Next Generation Broadcasting System for Arena Sports : A Football Stadium Scenario

by

Vamsidhar Reddy Gaddam

Doctoral Dissertation submitted to
the Faculty of Mathematics and Natural Sciences
at the University of Oslo
in partial fulfilment of the requirements for
the degree of Philosophiae Doctor

Abstract

Current multimedia applications are getting more user-centric by providing the user control over the content. Panorama video is becoming increasingly popular in that context, and we present an end-to-end real-time system called *Bagadus* to interactively zoom and pan into high-resolution panoramic videos. *Bagadus* integrates a sensor system, an event annotation system, and a video processing system using a video camera array. In this these, we focus mainly on the video capturing, user interaction and video delivery parts of *Bagadus*.

First, we look into the creation of the panorama video using multiple cameras, and we discuss several system-level details in implementing a real-time panorama pipeline. The developed pipeline creates a panorama of the size 4096×1680 pixels at 50 frames per second in real-time. Further, the panorama video is used to provide virtual pan-tilt-zoom services to the user, i.e., we present a design and implementation of a virtual viewer capable of providing interactive services in real-time.

Moreover, we argue that the energy spent in designing autonomous camera control systems is not spent in vain. We present a real-time virtual camera system that can create smooth camera motion. Similar systems are frequently benchmarked with the human operator as the best possible reference; however, we avoid a priori assumptions in our evaluations. Our main question is simply whether we can design algorithms to steer a virtual camera that can compete with the user experience for recordings from an expert operator with several years of experience? In this respect, we present two low-complexity servoing methods that are explored in two user studies. The results from the user studies give a promising answer.

We also discuss the challenges involved in delivering such interactive services to a large number of users. One of the major challenges involved in *Bagadus* is the overhead to transfer a full quality panorama to the client, where only a part of the panorama is used to extract a virtual view. Thus, a system should maximize the user experience and at the same time minimize the bandwidth required. In this regards, we apply tiling to deliver different qualities of different parts of the panorama. Tiling has traditionally been applied to delivery of very high-resolution content to clients, and here, we apply similar ideas in a real-time interactive panoramic video system. A major challenge is movement of such a virtual view, where clients' regions of interest change dynamically and independently from each other. We show that our algorithms, which progressively increases quality towards the point of the view, manage to (i) reduce the bandwidth requirement and (ii) provide a similar Quality of Experience (QoE) compared to a full panorama system.

Bagadus was originally designed and implemented with soccer as a use case. The importance of winning has increased the role of performance analysis in the sports industry, and this underscores how statistics and technology keep changing the way sports are played. Thus, this is a growing area of interest, both from a computer system point of view, in managing the technical challenges, and from a sport performance view, in aiding the development of athletes. The *Bagadus* prototype is currently installed and in use, at Alfheim Stadium and at Ullevaal stadium in Norway.

Acknowledgements

Several people deserve a thank you from me, for helping me out in the process of this thesis. A lot of them are not named here, so I would start by thanking them.

Today, I am what I am because of two important people: my mom and my dad. They taught me a lot of things and pushed me to reach my full potential at every wake of my life.

PhD is a hard process, during that I had several ups and downs. The media group has helped me to stay up and running. The person that I probably shared the most time and my office with is Preben. Michael, Ragnhild, Konstantin, Marcus and Viviane deserve a special mention for all the energy they spent on me. I would also like to thank Andreas, Kristian and Haakon for making my stay at Simula enjoyable.

My mentors, as well as supervisors, stayed patient with me all along. It is hard to differentiate between Prof. Pål and his clones some times, but I would thank them all for always being actively there for any supervision task - be it writing or discussing. Carsten provided valuable insights and profound conversations about anything under the sun.

My PhD time could have been worse if not for Johannes who constantly handled my BS and is always up for any activity. The tango group who managed to keep me sane during umpteen days of geek and craziness also deserves my gratitude.

Keeping the best for the last, she took care of me and she showed me a direction in life. It is only her love that changed me from "young, stupid and lost" to just "young and stupid". Thank you very much Lena.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> I just grew up watching a lot of movies. I'm attracted to this genre and that genre, this type of story, and that type of story. As I watch movies I make some version of it in my head that isn't quite what I'm seeing - taking the things I like and mixing them with stuff I've never seen before.
>
> *Quentin Tarantino*

The *1936 Summer Olympics* in Berlin mark the beginning of live broadcast of sports events. Ever since, the technologies in broadcast have seen a myriad of changes, improving the quality of broadcast several-fold. Figure 1.1 shows a typical broadcasting scenario. A typical workflow involves recording the action at an event from several viewpoints using several cameras. Then, a director takes control to create a unique stream, switching content from several sources, which later is broadcasted to the user receiving the stream on the other end. Over the time, we have seen better and faster cameras, better tools for directors, better networks to support high-quality content delivery and better screens to display the content. However, the power of a

user, for example *the young Tarantino*, to interact directly with the content has remained the same - still a user is a mere observer. Some experiments have tried to provide multiple contents between which the user can switch. But, the interaction is still limited, and the experiments were not successful.

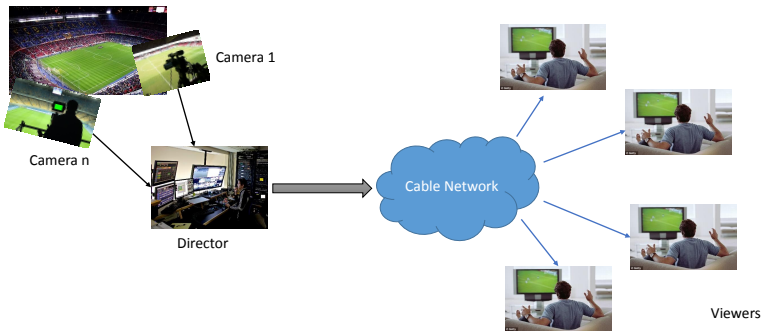
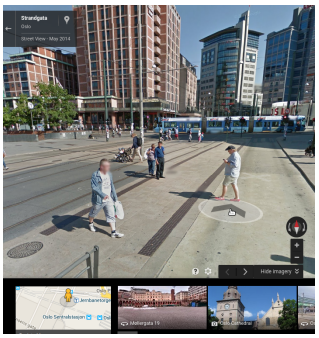


Figure 1.1: An overview of traditional broadcast.

On the other hand, the growing availability of high-speed Internet access and processing power on small devices has gone along with a growth in interactive and immersive multimedia applications. Providing users an opportunity to interact and immerse themselves in an environment enriches the user experience. Several applications have emerged where users can navigate virtual worlds. Often, the content for the virtual worlds is created by artists. The worlds are carefully created in a photorealistic manner to preserve the real world experience. Figure 1.2a demonstrates an example of such virtual environment, it can be seen that even though a lot of effort is placed in the creation of the environment, it nevertheless feels artificial.

In this regard, image-based rendering has helped interaction go a long way. For example, Google's *Streetview* is one of the well-known and frequently used applications that provides a nice interactive way to navigate through the streets. The content for *Streetview* is carefully captured and managed by Google. On the other hand, Microsoft's Photosynth provides users a web-platform to interact with user uploaded content. Figure 1.2 shows screenshots of Streetview and Photosynth. The idea is to provide the

(a) Virtual Environment.



(b) Google Streetview.

(c) Microsoft Photosynth.

Figure 1.2: User interaction with content in well-spread technologies.

ability to interact to the user using simple interfaces like a mouse or a keyboard or even using gyroscope on the cell phone. However, these services are limited to a single texture image that is loaded over time depending on the interaction.

Only recently, some people have ventured to replace the panoramic texture image with a video panorama in a similar interaction. An example from our system is presented in figure 1.3. Here, an example frame from a 50fps panorama video is presented along with the virtual camera that every user can control personally. However, the challenges in providing such a service with current technology are several. In this thesis, we add the component of *live real-time* to the interaction and explore the feasibility of such a service using soccer as a case study.

1.1 Problem Statement

Being able to provide tele-presence to individuals is a much desired technology. The dropping costs of different sensors and rising processing power make it easy to play with real-time (audio-visual-physical) data collected from a real scenario. However, there are several limitations in making the service available to a lot of users. Using a panorama video of a soccer stadium as a real-world scenario, our main question being pursued in this dissertation is as following:

Figure 1.3: The panorama video with the marked region of interest is shown together with the generated virtual camera, emphasizing that the extracted area is not a simple crop from the high-resolution panorama video. It is generated by a perspective reprojection and hence we cannot achieve it by a simple rectangular cropping.

What are the challenges involved in designing and developing live interactive video services to a large number of concurrent users?

We divided the main problem into the following research tasks that are tackled individually:

- Explore efficient and synchronized multi-camera capture, and panorama representations of video data from a soccer stadium to allow for live interaction.
- Explore the design of a basic interaction functionality and how that can be efficiently implemented on different devices.
- Explore efficient ways for a human to interact with the video streams on top of the basic interaction. Research higher level abstractions that

can be provided using a machine so that the interaction requires less work from the human-side.

- Explore the costs of providing this service, for both provider and the user, to several concurrent users. Explore ways to minimize the costs without altering the user experience.

1.2 Limitations

Even though the motivation of this thesis is to explore the services to a large number of concurrent users, due to several constraints like monetary cost and time, we have not experimented with this in the real world with the volume of users that is typical for a soccer game. We have limited our scaling solution to merely extending our framework improving already available large-scale solutions without large-scale testing.

We do not consider the audio component in this thesis. Even though the audio from a live sports event helps immensely to capture and carry over the emotions at the stadium, we have focussed only on the visual part of the broadcasting pipeline. This is largely due to the fact that the visual component plays a much bigger role in perception. Similar concepts, like capturing multiple signals and providing an interactive interface, can be extended to the audio component as well, but we did not venture into that domain.

1.3 Research Method

The final report [24] of the ACM Task Force on the Core of Computer Science gives a detailed description on how research in computer science can be organized. The main paradigms as discussed in the report are *Theory*, *Abstraction* and *Design*. Furthermore, the report identifies 9 subareas in the field of computer science. This thesis fits quite well into the Human-Computer Communication subarea, where the fundamental questions according to the report are [as quoted] :

> *"What are efficient methods of representing objects and automatically creating pictures for viewing? What are effective methods for receiving input or presenting output? How can the risk of misperception and subsequent human error be minimized? How can graphics and other tools be used to understand physical phenomena through information stored in data sets?"*

Even though the thesis works with practical system aspects making prototypes and running real-world experiments in soccer stadiums, thus trivially falling into the *Design* category, we still touch upon the *Abstraction* element in a fair amount of detail. The following defines the different research categories and provides how this thesis fits with all elements of Human-Computer Communication sub area:

Theory A research work that characterizes objects of study, finding patterns among the objects and verifying those using theoretical proof methods. In the *Theory* category, the report suggests touching upon elements like 2D, 3D geometries, color theory, linear algebra etc.

In virtual camera design and panorama creation, we explore the 3D geometry of a cylinder and perspective cameras. However, we do not cover any theoretical proofs in the thesis, and hence, we do not touch upon the *Theory* category.

Abstraction The hypothesis is verified using experiments and the data collected. However, the models are iteratively modified depending on the results from the experiments. In the *Abstraction* category, the report emphasizes on algorithms for various operations on images.

We explore ray-tracing in the creation of virtual camera. We perform several image processing operations both in the panorama creation and in virtual camera creation. Furthermore, during the user-interface stage, we study the effect of variables on the human experience. For achieving this, we developed multiple user studies to evaluate the system. These studies fall under the *Perceptual Psychology* domain. Even though there is a mention of

psychological studies in the report in regards to user-studies, it is unclear if perceptual psychology is covered in this topic of the report.

Design The engineering aspect of the research, that leads to design and development of system capable of realizing the theoretical ideas. Testing of the system also falls into this category. The *Design* category touches upon implementation of graphics algorithms, utilizing multiple architectures and use of displays.

All the theory and algorithms proposed in the dissertation are implemented into a real world system. The system uses multiple architectures and provide a real-world use case.

Apart from the above-mentioned elements, we also handle the idea of remote user - implying the addition of Computer Networks element to the Human-Computer Communication subarea.

1.4 Contributions

The single most important outcome of this dissertation is the working prototype of the live interactive video system running at Alfheim football stadium in Tromsø (used by Tromsø IL, elite club) and at Ullevaal stadium in Oslo (used by the Norwegian National team).

From the technical point of view, there are several contributions from the dissertation. The contributions are covered in detail later in the chapters, however, they are briefly listed as follows:

- The challenges involved in making a visually pleasing panorama automatically from a real-world outdoor stadium are several. We managed to generate a seamless panorama video from multiple cameras in real-time at 50fps including features like HDR, synchronized capture and dynamic seam to solve challenging light conditions. In the process, we have made several contributions towards real-time panorama creation on a distributed panorama video pipeline.

- We present a virtual Pan-Tilt-Zoom (PTZ) camera in order to extract an interactive personalized virtual view from a panorama video in real-time. We present different approaches for virtual camera extraction using heterogeneous architecture and speeding it up using a parallel architecture. Even on a commodity laptop hardware, we are able to achieve speeds up to 300 fps for extraction of the virtual view.

- We developed methods for automatically steering the PTZ camera to user's requirements. In this mode of interaction, the user can request to follow the ball or a player or a group of players and the system will automatically generate a virtual camera that smoothly pans/tilts/zooms according to the request.

- We also present valuable results from a subjective user study using the full pair-wise comparison tests to judge the performance of the automatic operation in comparison to an experienced human operator.

- A high resolution panorama video requires large bandwidth. We experimented with a DASH-based spatially segmented streaming approach to reduce the bandwidth required by splicing the panoramic video into tiles. We also demonstrate that, by fetching the appropriate quality tiles in relation to the user interaction, the quality of experience is not compromised, but the bandwidth consumption is greatly reduced.

In addition, the author has supervised several master students and published several papers (not all are included in the Appendix). The list of publications grouped with category is as following:

Journal Two papers (Appendix B [138] and Appendix D [40]) that are published in ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), and one paper (Appendix C [137]) published in International Journal of Semantic Computing (IJSC) contribute to this thesis. Apart from these, the work done during this dissertation has contributed to several other journal publications [84, 139], which are not included in the thesis.

Conference Three conference papers that were published during the course of the PhD contribute to this thesis. In chronological order, they are published at SPIE Electronic Imaging 2014 (Appendix F [41]), ACM NOSSDAV 2014 (Appendix G [42]) and Picture Coding Symposium 2015 (Appendix H [46]). Apart from these, the work done during this dissertation has contributed to other conference papers [143, 151], that are not included in this thesis.

Demo Three demos were presented at ACM Multimedia Systems Conference 2014 (Appendix I [45]), ACM Multimedia Conference 2014 (Appendix J [43]) and ACM Multimedia Systems Conference 2015 (Appendix K [44]).

Poster Two posters were presented at IEEE International Symposium on Multimedia 2014 [83] and at ACM Multimedia [76]. However, these are not included in this thesis.

Dataset We also provide the dataset recorded during the official game between Tromsø IL and Tottenham Hotspur for other researchers as a dataset paper [114] published in ACM Multimedia Systems 2014 . However, this is not included in this thesis.

Over the thesis, we are going to touch upon several related works and the state-of-the-art. Due to the number of fields a real-time live interactive video delivery system touches upon, there is a large amount of work published. However existing systems either (i) lack the real-time requirment, (ii) deal with only parts of such an interactive system, or (iii) do not elaborate on the system level details. Hence this thesis, detailing the challenges, research, development and evaluation of *Bagadus* system, extends the state-of-the-art by providing a comprehensive yet detailed journey into the world of real-time live interactive video delivery systems for sports. Finally, the entire code is made available as an open-source project [134]. This allows anyone to use the code and replicate our system for research purposes.

1.5 Organisation

Figure 1.4 presents the overview of the organisation of the thesis. In a nutshell, the thesis touches upon all aspects of the pipeline from the cameras at the stadium to the user's display devices.

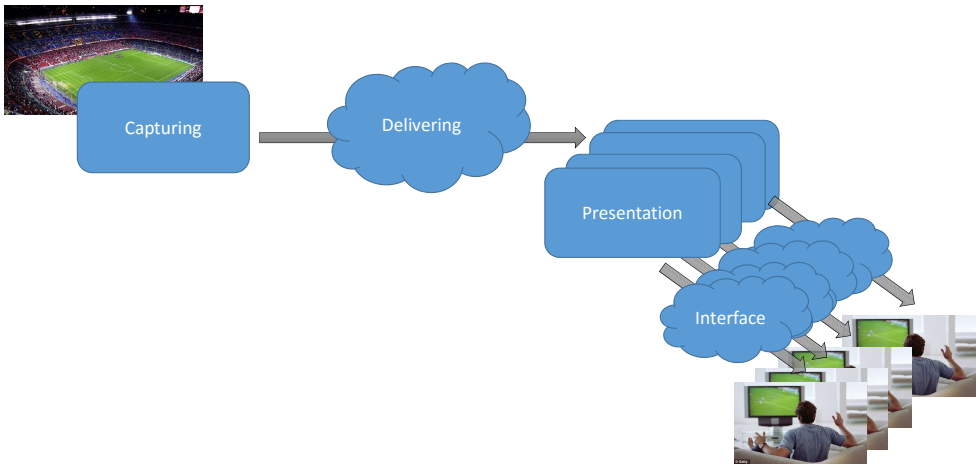


Figure 1.4: An overview of the organisation of the dissertation.

Chapter 2 initially provides the background and motivation for the work done during the dissertation. Then, we briefly present the related works in the field of interactive video services. It must be noted that each chapter individually delves into the state-of-the-work in the corresponding research areas.

The *Capturing* module of our pipeline is handled in Chapter 3. In this chapter, we present details about the panorama capture system. The chapter touches upon problems like synchronization in a multi-camera system, challenging light conditions in outdoor stadium, and how we met the real-time

processing requirements using a distributed pipeline.

Chapter 4 handles the details about *Presentation* module in the pipeline. In this chapter, we explore the details performing real-time projections from the captured panorama texture onto a cylinder on GPUs. This operation plays a crucial role in realizing the pan-tilt-zoom virtual camera.

The *Interface* module provides an abstraction layer to the user in order to manipulate the PTZ camera based on the preference. The user can request the system to provide a virtual view that follows the ball (or players) and the system automatically creates a smooth movement for the virtual camera. Chapter 5 presents details on how this interaction is achieved. In this chapter, we also perform user studies to understand the effect of various parameters on human interaction.

The *Delivering* module of our pipeline is responsible for delivering the interaction services to several concurrent users. Chapter 6 presents the main motivation of research in delivery stage i.e., why delivery can be challenging, and a few solutions to tackle the problem of high bandwidth usage.

Chapter 7 concludes the dissertation by summarizing the work . We also open up a discussion about open issues and further possible directions.

Chapter 2

Background

> My definition of an expert in any field is a person who knows enough about what is really going on to be scared.
>
> *PJ Plauger*

The interest in sports analysis systems has recently increased a lot, and it is predicted that sports analytics will be a real game-changer, i.e., "statistics keep changing the way sports are played — and changing minds in the industry" [31]. To the best of our knowledge, the coach of a sports team appreciates input from several sources that helps in the analysis of the performance of both the team and individual players. In the area of soccer, several systems enable trainers and coaches to analyze the game play in order to improve the performance. However, the analysis currently involves large components of manual work. Some systems, like the Cairo's VIS.TRACK [14] using global positioning and ZXY Sport Tracking [162] using radio frequency, help in capturing player performance measurements. These systems, using the collected data, can further present player statistics, including speed profiles, accumulated distances, fatigue, fitness graphs and coverage maps, in many different ways like charts, 3D graphics and animations.

2.0.1 Role of Video in Analysis

Traditionally, soccer games were recorded for viewing and archival purposes. However, the recent interest in sport analysis has put more use to the videos. The Norwegian national soccer team head coach said :

> *"Research has shown that immediate feedback in the form of images has proven to be most effective"* - translated from Norwegian
>
> *"Forskning har vist at umiddelbar feedback i form av bilder har vist seg å være mest effektiv"*
>
> - Per-Mathias Høgmo speaking about Bagadus [107].

In the recent times, videos are not only used to replay the instances for training purposes but also for both extracting game patterns from the video and embedding analysis meta-data along with the videos for further usage. For instance, in Interplay-sports [68], video-streams are manually analyzed and annotated using a soccer ontology classification scheme. ProZone [116] automates part of the annotation process using video-analysis software. In particular, it quantifies player movement patterns and characteristics like speed, velocity and position of the athletes, and is successfully used at several stadiums, for example, Old Trafford in Manchester and Reebook Stadium in Bolton [129]. Similarly, STATS SportVU Tracking Technology [135] uses video cameras to collect the positioning data of the players within the playing field in real-time. This is further compiled into player statistics and performance. Camargus [15] provides a very nice video technology infrastructure, but lacks other analytics tools.

The existing systems are offline when it comes to video consumption, and can deliver different types of reports some time after the game or training sessions. To improve game analytic systems, video that replays real game events immediately becomes increasingly important. However, the integration of the player statistics systems and video systems still requires a large amount of manual labor. For example, events tagged by coaches or other human expert annotators must be manually extracted from the videos, often requiring hours of work in front of the computer. Furthermore, connecting

the player statistics to the video also requires some manual work. One recent example is the Muithu system [73], which integrates coach annotations with related video sequences, but the video must be manually transferred and mapped to the game timeline.

As we have seen earlier, there exist several tools for soccer analysis. However, to the best of our knowledge, there does not exist a system that fully integrates all the features, like video data, event data and player data. In this regards, we have earlier presented [52] and demonstrated [128] a concept-system called Bagadus. This system integrates a camera array video capture system with the ZXY Sport Tracking system for player statistics and a system for human expert annotations. Bagadus allows the game analytics to automatically playback a tagged game event or extract a video of events extracted from the statistical player data, for example all sprints at a given speed. Using the exact player position provided by sensors, a trainer can also follow individuals or groups of players, where the videos are presented either using a stitched panorama view or by switching cameras. Our earlier work [52, 128] demonstrated the integrated concept, but did not have all operations in real-time, in particular the video operations.

2.1 Bagadus – The Basic Idea

Bagadus system is built in cooperation with the Tromsø IL soccer club and the ZXY sport tracking company. A brief overview of the architecture and interaction of the different components is given in figure 2.1. The Bagadus system is divided into three different subsystems.

2.1.1 The Subsystems

The *video* subsystem consists of multiple small shutter-synchronized cameras that record a high resolution video of the soccer field. They cover the full field with sufficient overlap to identify common features necessary for camera calibration and image stitching. Furthermore, the video subsystem supports two different playback options. The first allows playback of video

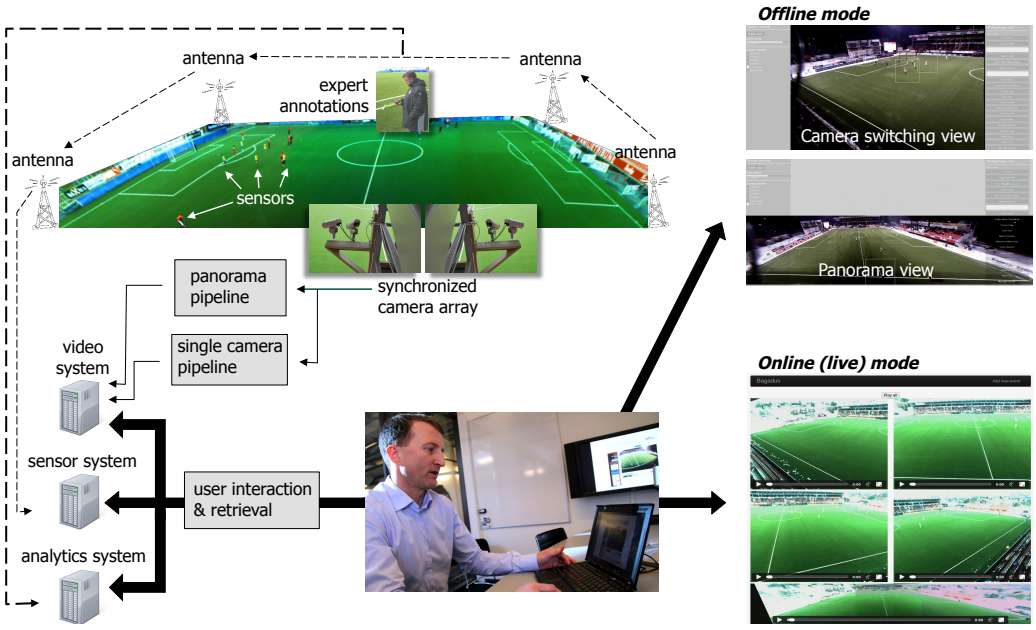


Figure 2.1: Overall Bagadus architecture.

that switches between streams delivered from the different cameras, either manually selecting a camera or automatically following players based on sensor information. The second option plays back a panorama video stitched from the different camera feeds. The cameras are calibrated in their fixed position, and the captured videos are each processed and stored using a capture–debarrel–rotate–stitch–encode–store pipeline.

To identify and follow players on the field, we use a *tracking* (sensor) subsystem. Tracking people visually by means of camera arrays has been an active research topics for several years [9, 25, 32, 62, 63, 89, 92, 96] . The accuracy of such systems has improved greatly, but the accuracy reported in the literature is usually around 60% for large arena sports. Therefore, for stadium sports, an interesting approach is to use sensors on players to capture the exact position. In this area, ZXY Sport Tracking [162] provides such a sensor-based solution that provides player position information.

The third component of Bagadus is an *analytics* subsystem. Coaches have for a long time analyzed games in order to improve their own team's game play and to understand their opponents. Traditionally, this has been

done by making notes using pen and paper, either during the game or by watching hours of video. Some clubs even hire one person per player to describe the player's performance. To reduce the manual labor, we have implemented a subsystem that equips members of the trainer team with a tablet (or even a mobile phone), where they can register predefined events quickly with the press of a few buttons or provide textual annotations. In Bagadus, the registered events are stored in a database, and can later be extracted automatically and shown along with a video of the event. In figure 2.2, the head coach can be seen using Bagadus app during a training session and the events are immediately played back on the big screen at the stadium.

(a) Bagadus on big screen. (b) App used by the head coach.

Figure 2.2: Bagadus System currently in use at the Ullevål stadium by the Norwegian National Team.

2.1.2 Interaction Between Subsystems

Bagadus implements and integrates the subsystems to support our arena sports analytics application scenario. The main novelty of our approach is the combination and integration of subsystems to enable automatic presentation of video events based on the sensor and analytics data that are synchronized with the video system. We can follow single players and groups of players in the video, and retrieve and playback the events annotated by expert users.

Thus, in the offline mode, Bagadus for example is able to automatically present a video clip of all the situations where a given player runs faster than 10 meters per second or when all the defenders were located in the

opponent's 18-yard box (penalty box). Thus, where people earlier used a huge amount of time for analyzing the game manually, Bagadus is an integrated system where the capturing of tracking data, retrieval of analytics data, and the synchronization with video are automatically managed. Bagadus can receive expert annotated events from the team analytics team and enable an immediate playback during a game or a practice session, however the video subsytem is not functional in real-time. In this thesis, we explore the challenges involved in making the video subsystem real-time and in addition improve the quality of the video significantly.

2.2 Related Interactive Systems in Research

Several research groups have experimented with creating interactive video systems in sports scenarios. In this section, we discuss about a few such systems with the emphasis on *video*, not analytics.

In [4], Ariki *et al.* presented a system that can be considered as a simpler case of the Bagadus system. They recorded the videos using a HD camera. They present Region-Of-Interest (ROI) by cropping a rectangular selection in the video. They perform processing of captured videos to estimate ball position and player positions. Then, they perform situation recognition based on the estimated positions using some simple rules. After this, they create a ROI path with a fixed set of rules. In addition to this, they also devised another set of rules for user preference.

Yokoi *et al.* [158] were active in virtual camera research a decade ago. They explore the simple scenario of a lecture hall. They used an HD video camera to capture the high resolution and crop the ROI from that video. In this work, they explored different approaches for operating the ROI that automatically follows the presenter in a lecture hall. They performed a small user study from 20 students and evaluated their algorithms.

Carr *et al.* [18] present a hybrid system using both a robotic PTZ camera and a virtual camera generated from panorama. They evaluated their system comparing it to a human operated one as benchmark. Their motivation was to get as close to the human operator as possible.

Chen and De Vleeschouwer [21] performed an automatic production planning over multiple cameras. They used several cameras located around a basketball stadium to capture the action from different points. Then, by analysing the game, they generate a single broadcast by combining different streams from different views along the time axis.

Mavlankar *et al.* [98] explored the field of interactive region of interest streaming from the network optimization side. Their work initially involved lecture videos and later extended to a soccer stadium. The panorama used in their work is a perspective panorama.

Ooi *et al.* [104] have explored two approaches for supporting ROI streaming from high resolution videos. In one approach, they split the video into pre-determined tiles and the other is where only the macro-blocks required for the ROI are transferred to the client. From this work, they have later provided several contributions regarding mobile devices [90], quality assessment [146], wireless networks [51] and crowdsourcing [16].

The research systems presented above provide proof-of-concept for different areas of live interactive video service. However, the need for immediate feedback and real-time performance have led us to develop the Bagadus system to a fully functional working prototype.

2.3 Summary

In this chapter, we briefly introduced the scope of Bagadus system and other similar existing research systems. However, related work is being touched upon in the chapters wherever necessary. The area of interactive video delivery is quite broad, and there are several works that focus and deal with parts of the challenge. We implement, study and evaluate different components together from systems point of view in this thesis. In the next chapters, we delve deeper into parts of the video system that can provide interactive visual experience in real-time, and we also consider the network performance.

Chapter 3

Panorama Video Capture

> There is nothing insignificant in the world. It all depends on the point of view.
>
> *Johann Wolfgang von Goethe*

In Bagadus, we aim to capture everything happening all the time from one viewpoint, i.e., we aim for a field of view of the entire soccer field. However, it can be challenging for one camera to capture such large spaces at a high resolution. A *panorama* can be used to capture a large *scene* with a wide field of view. Usually, panoramas are captured using multiple images that cover parts of the same scene. This can be primarily achieved by using the same camera that pans across the scene, or using multiple cameras pointed at different parts of the scene. In both cases, all the cameras are at the same point (*viewpoint*). The former case has the limitation of recording only a panorama image. In Bagadus, we would like to record a panorama video and so we use multiple cameras, each pointed towards different parts of the field. In total, they cover the entire field and have some overlap among the images of the adjacent cameras.

A lot of work has gone into automatic alignment of multiple cameras for panorama stitching. However, the Bagadus system has a static camera setup. This fact makes it possible to avoid recalibration of the camera setup and use only a static mapping from each camera space to a panorama space. How-

ever, apart from the alignment problem, a panorama capture poses several other problems like temporal synchronization between the cameras, exposure matching, parallax, dynamic range and high throughput. A further requirement for a real-time system is that all of the processing must happen within the time limits of the target framerate.

The research problem is to find an efficient way to solve the above challenges related to real-time panorama video capture. Using multiple cameras, heterogenous processing and efficient panorama representations, we aim to tackle the above presented challenges. We present and discuss theory and ideas, show the development of multiple versions and present respective experimental results and shortcomings. Finally, we present the system that is capable of recording and delivering a 4k panorama video at 50fps.

Most of the content of this chapter is extracted from papers [41] (Appendix F), [143], [138] (Appendix B), [114] and [76]. It is organized and modified to present the pipeline to reflect its development through multiple versions while gaining increasingly better insight.

3.1 Panorama Frame Generation

In Bagadus, we capture the field using multiple cameras. When a scene is captured using multiple cameras, we have some overlap between the cameras and the switch from the image of one camera to the next is not smooth. One can deliver the frames from individual cameras by switching the cameras. However, this kind of delivery provides a hard transition from one camera to the other one. So, we explored some options to map the camera frames into one single representation. In this regard, the obvious choice is to create a panorama.

3.1.1 Theory

In this section, we refer to the location of the camera setup as the *viewpoint* and the view of interest, football stadium in the case of Bagadus, as the *scene*. When using a single viewpoint, the scene that one observes can be

efficiently stored into a single representation. In this regards, there are two major possibilities, the complete and the partial panoramas. We provide a brief introduction to the theoretical concepts of panorama in this section. In all the cases described here, the panorama is rolled onto a plane and stored as a $2D$ image.

3.1.2 Full Panoramas

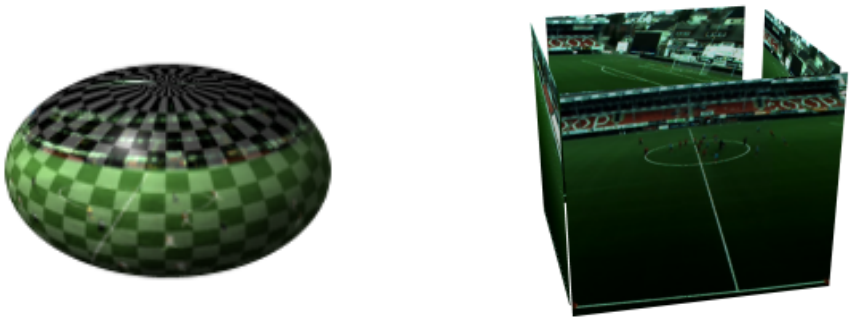
When the whole scene around one point needs to be captured, two common approaches are employed. One approach is to project the scene onto a *sphere* around the camera, and the other is to project the scene onto a *cube*. This imples that 360^o along both the horizontal and vertical directions are captured. Figure 3.1 demonstrates this using an example texture from the soccer field.

Spherical Panorama

A spherical projection provides the most compact form to capture the scene. This implies that the redundancy in the data is quite low, if not zero. However a spherical panorama introduces the non-linear projections and thus curving straight lines. Figure 3.1a shows an example sphere with a panorama texture.

Cube Panorama

A 6-faced cube representation is commonly used to represent a scene. This is a commonly used technique in games and other VR applications. The cube representation is a linear transformation per face from scene. This ensures that the straight lines remain straight in each face. However, the cube is not the most compact representation, which implies that for the same scene a cube representation will require more data than the spherical panorama. Figure 3.1b shows an example of cube representation. In the figure, the faces are made discontinous to demonstrate the advantage of cube representation: the texture can, intrisically, be stored as 6 rectangular images unlike the sphere case , where explicit unrolling must be done.

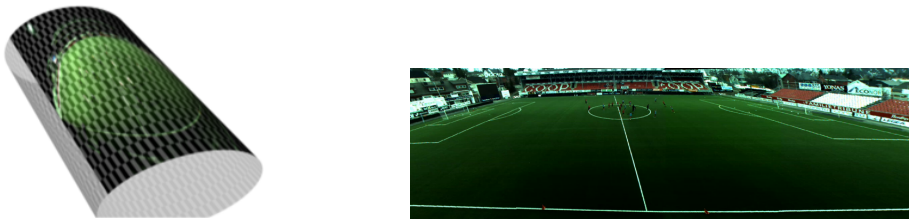


(a) Sperical Panorama. (b) Cube Panorama.

Figure 3.1: Spherical and Cube projections.

3.1.3 Partial Panorama

When the scene being captured does not pan 360^o horizontally and vertically, there are other ways to capture the panorama that are similar to the full panoramas. In this regard, there are two most common ways of capturing the scene. The first is a cylindrical panorama and the second is a perspective panorama. Figure 3.2 shows an example of partial panoramas.



(a) Cylindrical Panorama. (b) Perspective Panorama.

Figure 3.2: Cylindrical and Perspective projections.

Cylindrical Panorama

A cylindrical panorama is formed by projecting the scene onto a cylindrical surface. This kind of panorama is most ideal when the horizontal field of view is large and vertical field of view is limited. The projection onto a

cylinder ensures that the vertical lines remain straight and non-linearity is introduced only horizontally. Figure 3.2a shows an example of cylindrical panorama. It can be seen that the actual texture is partial, not covering the whole cylinder, because the football field only has 160^o field-of-view in our case.

Perspective/Rectilinear Panorama

A perspective panorama, also known as rectilinear or planar panorama, is simply the projection of the scene onto a pin-hole camera. Most images captured with a camera resemble a perspective panorama. However, a wide field-of-view perspective panorama is not trivially captured from a single camera. The advantage of using a perspective panorama is that straight lines remain straight. However, a perspective panorama has the problem of uneven sampling, the far-end of the field is compressed and the near-end is elongated. This can be observed in the figure 3.2b. This uneven sampling gets worse as the field-of-view increases.

Considering the figure 3.3, a slice along the XZ plane is presented for both the cylindrical and the perspective panoramas. One can observe that if the field-of-views, $\theta_1 = \theta_2 = \theta_3$, then it ensures that the arc lengths, $C_1 = C_2 = C_3$ on the cylinder. However, the line lengths, $P_1 = P_2 = P_3$ does not hold. This creates uneven sampling problems giving more samples for the same angle in the extremes and less in the middle.

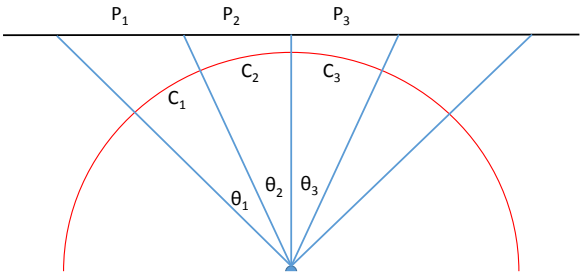


Figure 3.3: Uneven sampling in the case of a perspective panorama when compared to the cylindrical projection.

3.1.4 Related Work

The field of panorama images is a well established one. One of the oldest works in panorama can date back to 18th century. However, the use of machines to assemble photographs into a panorama image is only a couple of decades old. There is a lot of research that went into this in tackling different problems. For example, Szeliski *et al.* [141] present various ways to make panoramas. This work can be considered as an introductory tutorial for panorama creation.

Peleg *et al.* [113] discuss about stitching panoramas. Their premise is that of a moving single camera that is following some sort of panning motion. The two scenarios of *panning* are illustrated in figure 3.4. In the first case, the camera translates and in the second case, there is a rotation around the same point. However, they argue that this action cannot be perfect when performed mechanically. So, they propose a general manifold based stitching. In this work, instead of using a planar projection or a cylindrical projection, they make a projection depending on the camera. However, they do not go to the end of tracking the camera, instead they count on the errors in motion to be low and create the manifold by simple projections. After the alignment process, they stich the images together and blend them for photometric inconsistencies at the edges.

Xu *et al.* [157] proposed a panoramic video capture system using a few HD video recorders attached to a wooden planck. They move the capture system around thus capturing a pre-defined view-point change too. Their approach employs a color correction approach and motion blur detection. The recorded panorama videos are merely displayed on wide displays.

Brosz *et al.* [10] propose interesting ways to make a panorama defined by shapes. They argue that the parametric forms like cylinder, sphere or cube suffer from problems like irregular sampling when a panorama is changed from one format to the other. Instead, the authours want to make a general shaped panorama and sample only on the angles. They then create profiles for the surface, which are used to define the arbitrarily shaped surfaces. This method is interesting in computer generated panoramas and the work

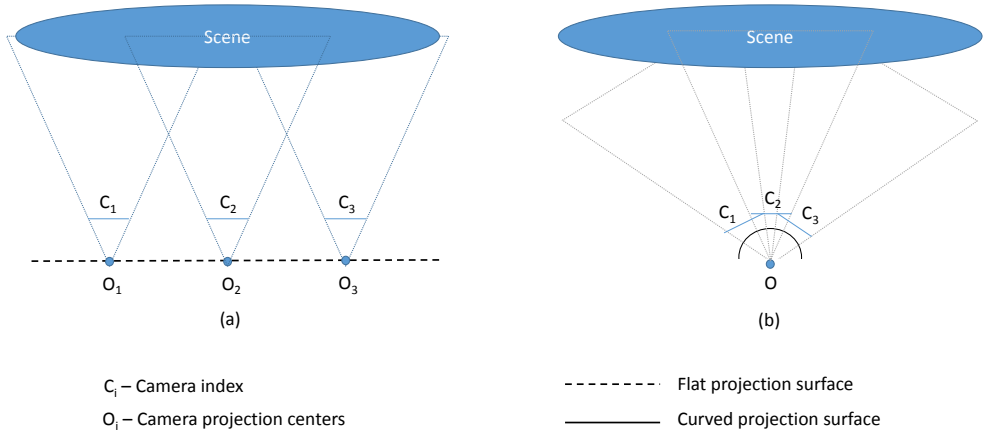


Figure 3.4: Two possible situations for recording a scene using multiple cameras. (a) represents a case where the cameras are translated, and (b) is the case where the cameras are rotated.

is constrained to those. However, they do not present any results from a real world scenario.

Bradley *et al.* [7] write about something that looks like Google Street View. They use a 6-camera Point Grey Ladybug system to capture the panoramas indoors. They capture panoramas and store them along with their location of capture. They provide an interface where the user can navigate an indoor space by moving around and looking around. The moving is performed by loading the new panoramas from the new location and looking around is simulated by 6-face cube projection. The approach is simple and uses images, not videos. They do not really talk about the system-level details like performance. Their example is from indoor data collection. All the challenges like moving subjects, occlusion and other panorama-related challenges are not discussed.

Jogan *et al.* [72] talk about an interesting method where they use panoramas for matching. They use a set of pre-captured panoramas and reduce the localization task to identifying the best match in the set. However, they do

not perform a simple correlation because that cannot be robust to noise and occlusion. They use eigen images to create a set of hypotheses as subsets of images points. The paper is interesting in terms of how panoramas can be used. Another work about using panoramas in an interesting way is presented by Hermans *et al.* [54]. They talk about showing objects on a static panorama scene and let users interact with them.

Wai-Kwan *et al.* [142] speak conceptually about a system that is capable of live capturing, rendering and presentation of the panorama. There is no discussion about the resolution of the panorama. Their major contribution is the *cockpit*, which is a curved projection system-based display. The work lacks discussion on details of the system. However, they mention a few system level aspects such as time taken by each of the modules.

Kimata *et al.* [77] set out a framework that is similar to Bagadus. However, they consider the challenge of virtual view extraction as simple crop. They create panoramic video from multiple cameras. They encode it using MVC for supporting tiles. Then, the corresponding tiles are fetched and displayed. However, the results demonstrated show that the setup is an experimental prototype rather than a working system.

Foote *et al.* [39] propose flycam, which is a cheap way of recording a panoramic video. They use a few cameras to capture the scene and stitch to a panorama. They talk about a virtual camera, but this is not perspective-corrected. They use simple cropping to get a region from the panorama. They also discuss controlling this *virtual camera*, by using motion analysis automatically. For the movement, they use a heuristic with an inertial momentum to smooth jitter and slowly pan the virtual camera.

Au *et al.* [5] present ztitch, an app that can make panorama images on a cell phone. They have an interesting approach, however their competitor, Photosphere has better version. This app runs on Windows Mobile unlike Photosphere. One can capture a few images using cell phone by pointing the camera in different directions. Then the images are stitched together using the input from the user. The user can manually drag the images around to help with the stitching process. After the alignment, the images are blended together to remove the seams.

Qi *et al.* [117] talk about a way to make seamless scenes with realistic conditions. Their main application scenario is teleconferencing. In teleconferencing, there can be two situations, as seen in figure 3.4 to use multiple cameras. The first scene would help to maintain a face-to-face angle with the camera if the cameras are placed infront of people. However, this would introduce large parallax as the cameras are away from each other. The second scene avoids parallax, however they argue that they suffer from lack of the direct face-to-face perspective affecting the quality of teleconferencing. They use slightly displaced cameras to capture multiple images. Then, they generate intermediate views from the captured images to help the process of mosaic stitching. Finally, they perform the panorama stitching with all the images as source. Figure 3.4 shows the different scenarios of recording a scene using multiple cameras.

Agarwala *et al.* [3] present an interesting approach to make long panoramas from a moving camera, where the view point is panned across the scene. The typical application is on streets. In short, they take a video travelling in a *straight line* of the street. This will give a lot of images panning the street. Their goal is to stitch these images into one seamless panorama that doesnt suffer from perspective distortions. The main constraint is that they want to keep an orthographic projection vertically and perspective projective horizontally, we refer the actual paper to the reader for more details on this constraint. They achieve this in several steps. The first is a preprocessing step, where they estimate the camera position for each image. Then, they compensate for exposure variations in the same step. The next task is to figure out a dominant scene plane. Figuring this out automatically is not trivial. For this, they make the 3D scene along with camera pose estimation and then do a Principal Component Analysis (PCA) on camera viewpoints. Here, they also provide for user input to correct the surface plane. Then the viewpoint selection is performed. The task is defined as finding which source image contributes to the current pixel on a panorama. They formulate this as a Markov Random Field (MRF) and solve it. There are three terms in the objective function, the first one is to make sure that the viewpoint is approximately infront of the object. The second term is to ensure smooth

transition between different regions of the panorama. The final term is to make sure that photometrically the panorama is close to the average image of the sequence. After the optimization,the user can provide input to the process where the solution can be refined for different constraints. The view selection is refined by picking which viewpoint should contribute to the region. Sometimes seams go through objects, creating unwanted artifacts, but these can be avoided by letting the user draw strokes to make sure that no seam passes through the stroke. This is not straight-forward, as a stroke from one viewpoint must be transformed to the other viewpoints. For this, they use homography from one image to the other to transform the stroke place on image to the other image. The final input mode is for inpainting, this is a common input for removing objects that lie off the dominant plane. Their results are impressive, but there are a lot of limitations to automatically achieving this task due to the involvement of the user in the process.

Carr *et al.* [17] present a portable system that is similar to Bagadus. However, their stitching happens on the client side. Also, they make it robust to movements between the cameras, which implies re-estimation of homography. The system they propose captures images from multiple cameras and stores them on the disk. They have an interesting approach for doing this. Their server figures out region of interest by analysing the image and then crops the image outside the region of interest. Then, a resampling is performed to match the resolution of the display device. Further, all these cropped regions are transmitted to client. On the client, the images are aligned and stitched together to form a mosaic. The alignment is performed by perspective transformation. The main limitation of their system is that they expect the user to provide point-correspondences between the views. This is a drawback, because if the user is expected to provide correspondence, the idea of automatic alignment is defeated.

Eden *et al.* [35] talk about making HDR panoramas, where their solution is quite elegant in terms of initial constraints. They start with the a really challenging problem of using images captured by amateur photographers. This implies that their solution should handle highly varying exposures between images and also that the images are not perfectly aligned. They pro-

pose to build a good panorama that has sufficiently high dynamic range and wide field-of-view. They work in *radiance space* to handle the varying dynamic range. In order to achieve this, they perform geometric alignment and radiometric alignment. Further, they do a two-pass selection process, where a reference panorama is created from the aligned inputs in the first pass. A cost function is defined so that it incorporates the data priors and smoothness priors, which is then minimized using graph-cut. After the two-pass process, they perform blending to create a smooth image. The output image is impressive, however, considering all the optimization processes, the computational cost of such an approach is quite high.

Apart from the above works, there are a few works that deal with small details of panorama stitching at a theoretical level. McMillan *et al.* [100] talk about plenotic modeling. Migliorati *et al.* [102] talk about interpolation adapted to the movement of the camera and the objects. Junhong *et al.* [48] mentions an interesting way to stitch panorama images using 2 homographies between the same images, instead of the classical way of using 1 homography. Lin *et al.* [88] propose an interesting approach where instead of using a single homography, they use a smooth affine to stitch the panorama. Ioana *et al.* [131] propose an interesting way of stitching two images with photometric inconsistencies into one good-looking panorama. From the works mentioned in this section, we use theoretical concepts from Szeliski *et al.* [141] regarding panorama stitching. However, when it comes to system-level aspects, there is a lack of detailed research in the field.

In summary, there exists a lot of related work, but none addresses the issues of a complete real-time panorama pipeline. Hence, we describe our work in the next sections.

3.2 Bagadus Setup

The video capture system consists of a few cameras to cover the entire field and processing machines to further process the captured data. The system is designed to record all parts of the field from one vantage point at all times. We initially built a system with 4 cameras which we upgraded to 5 cameras

with a better design. In this chapter, we use the term 'old system' to refer to the 4-camera setup and 'new system'/'system' to refer to the 5-camera setup. The new system is the system currently in use. However, a significant initial design and development took place with the old system. A major point to be noted in this chapter is that results may not be presented for both the setups in all experiments. This is due to the fact that some experiments were run only during the development of the framework and when the framework was adopted to the new system, these experiments were not re-run.

In order to realize the construction of a panorama video, we built a panorama pipeline. The pipeline consists of components performing primarily three operations - capturing, processing and storing.

3.3 Serial Panorama Pipeline Implementation - Version 1

The old system is shown in Figure 3.5a. The cameras are placed on one end of a recording stand. This stand is also used by the professional TV broadcasters to record the games. Hence this ensures that the recording happens from the same view-point as professional broadcaster's. The 4 cameras and lenses used in the setup are all the same and their details are presented in Table 3.1 and Table 3.2. The output from the cameras is presented in the Figure 3.5.

Manufacturer	Basler
Model	acA1300-30gc
Sensor format	1/3"
Pixels	1296 × 966
Frame Rate	30

Table 3.1: Basler acA1300 Camera.

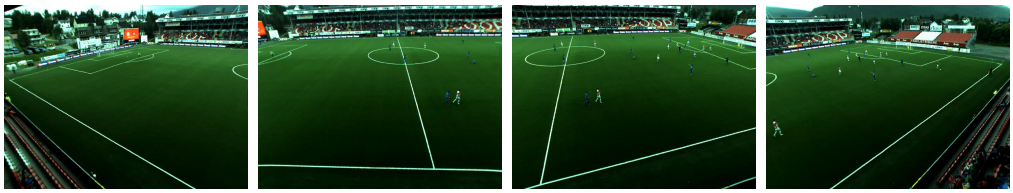
Manufacturer	Kowa
Model	LM4NCL
Focal Length	3.5 mm
Aperture	F1.4 - F32

Table 3.2: Kowa 3.5mm Lens.

3.3.1 Time Synchronization

When the camera frames are placed next to each other, like in the case of panorama, the temporal synchronization becomes a key issue. The hu-

(a) Old Camera System Setup.



(b) Camera 1. (c) Camera 2. (d) Camera 3. (e) Camera 4.

Figure 3.5: Views from each camera for a perfectly synchronized exposure after the debarrel step.

man eye can easily notice differences in frames captured at different time instances when placed next to each other. Initially, we have observed that just triggering cameras to operate at a specific frame-rate is prone to cause synchronization errors as there are several independent clocks and they are bound to drift with respect to each other. Hence, we decided to use hardware trigger, using which we inform all the cameras when to capture a frame continously. The cameras are equipped with such an interface. Figure 3.6 shows the 6-pin interface of the cameras used in the Bagadus system.

Table 3.3 gives an overview of the purpose of each pin. The pins 1, 3, 5 and 6 are used for power and ground purposes as can be seen in the table. Pin 2 takes input from an external trigger device depending on the mode set in the camera. Pin 4 gives out the status of the shutter(open/close) in case it needs to be used in other control mechanisms. The input pin can be used to convey mainly two things, when to make an exposure and also how long. However, we use the camera's internal metering abilities, meaning that we only trigger the camera when a frame needs to be captured.

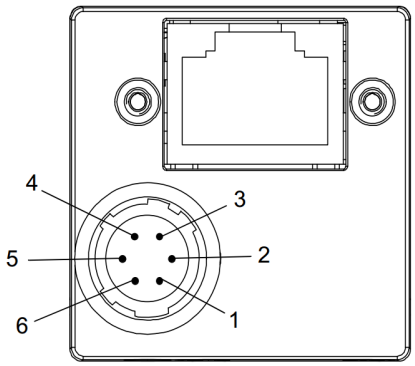


Figure 3.6: Camera Hardware *6-pin* interface.

Pin	Designation
1	+12V DC Camera Power
2	I/O Input 1
3	Not Connected
4	I/O Out 1
5	I/O Ground
6	DC Camera Power Ground

Table 3.3: Camera pin assignments.

To achieve a good time synchrony among multiple sub-systems in the Bagadus system, we use a Network Time Protocol (NTP) server at University of Tromsø. A small program delivers a single time-sync packet containing the frame-rate once every second to an Arduino that resides close to the camera setup. The Arduino delivers a trigger signal to all five cameras using its I/O pins i.e., it sends $\frac{1}{frameRate}$ Hz signals. The Arduino's timing is reset by every packet from the server hence keeping the drift to less than 1 second in the worst case. This leads to a synchronized trigger among multiple cameras down to a frame level and if there is time drift between the cameras, it is reset every second. The panorama-recording machines use the same NTP server and so, a time stamp on the final output panorama video file contains all the information required to synchronize with other sub-systems later on.

3.3.2 Components

Figure 3.7 presents the image with different processing modules of the pipeline. We have earlier discussed about synchronized capture from multiple cameras. Initially, the frames are captured in YUV format and then converted to RGB because of the limitation enforced by OpenCV. In Version 1, we relied on off-the-shelf video processing libraries. OpenCV [8] is one of the most used and well maintained video processing libraries available. However, several processing modules in OpenCV require the image to be in RGB format, and hence, we had to add the extra step. Then, the frames are corrected for barrel distortion, because of the wide angle lens used.

Figure 3.7: Simple serial process to build a panorama.

Once the frames are corrected for barrel distortion, they are sent into the warping module which uses pre-estimated homography between the panorama plane and each of the image planes to project the images onto the panorama plane. Once the warping is performed, the images are stitched together to form one complete panorama.

Once the stitching is complete, the panorama is converted back to YUV format so that it can be stored using H.264. We used x264 library to encode the panorama video in YUV420 format.

3.3.3 Evaluation & Results

An example panorama output from version-1 is shown in figure 3.8. Initially, we performed a few experiments to understand the costs of each module in the panorama pipeline. We performed all these operations on CPU in a serial fashion. Table 3.4 presents the time taken for each stage of the pipeline. In total, an average of 1.1 seconds is required per frame from the camera to the encoded panorama file. Obviously, this is much longer than the real-time

requirement of 33 ms for a 30 fps video. Furthermore, even if the components would run in parallel, we can see that the *stitching* module takes the most time. Hence, in the next implementation (version 2), we decided to parallelize the pipeline to achieve real-time performance.

Figure 3.8: Result of Version-1 pipeline.

Process	Time (ms)
RGB to YUV	4.9
Debarrel	17.1
Stitch	974.4
YUV to RGB	28.0
Write	84.3
Total	1109.0

Table 3.4: Profiling of the stitching pipelines (in ms per frame).

3.4 Parallel Real-Time Pipeline - Version 2

The experiments for version-1 displayed some severe processing overheads with respect to generating a 30 fps panorama video in real-time. In this section, we address this by implementing the modules in a parallel pipeline. There are many ways to parallelize such a pipeline on both CPUs and other offloading devices. Other research [136] shows great potential of GPUs, and next, we therefore offload compute-intensive parts of the pipeline to a modern GPU. The pipeline is illustrated in figure 3.9.

3.4.1 Components

As shown in figure 3.9, the pipeline has been greatly modified compared to version-1 in figure 3.7, to both improve the picture quality and the processing

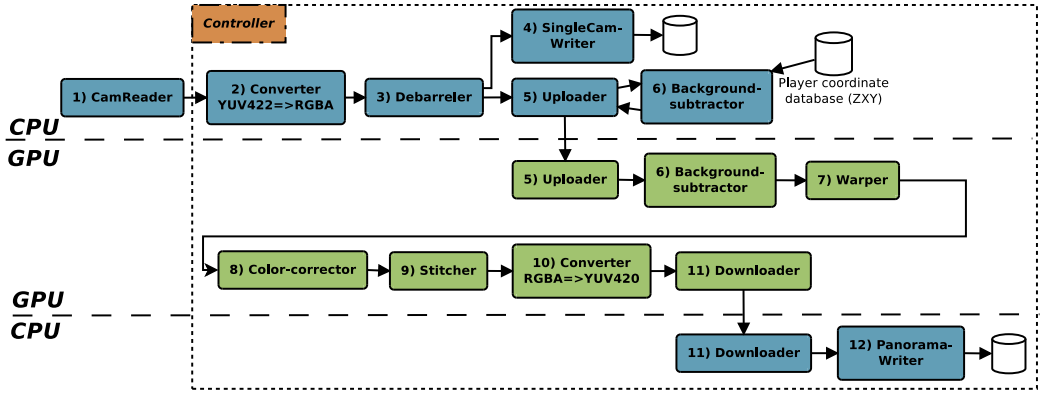


Figure 3.9: The parallel and distributed processing implementation of the stitching pipeline.

speed. The parallel pipeline is separated into two main parts, one part running on the CPU and the other part running on a GPU. Several of the CPU modules in the pipeline is the same as in the non-real-time loop. The *CamReader*, *Converter*, *Debarreler*, *SingleCamWriter* and *PanoramaWriters* are based on the same design as version-1, but are running in their own threads and with an updated version of the x264 encoder in version-2. Compared to version-1, the *Controller module* is new and is responsible for initializing the pipeline, synchronizing the different modules, handling global errors, frame drops and transferring data or data pointers between the different modules. The Controller also checks the execution speed. If an earlier step in the pipeline runs too slow, and one or more frames have been lost from the cameras, the controller tells the modules in the pipeline to skip the delayed or dropped frame, and reuse the previous frame.

The *Debarreler* module is the same as the serial implementation and we have opted to use the OpenCV code for it, as the component is already capable of performing in real-time. However, by using this component, we again are restricted to the usage of RGB color space in the pipeline. Hence, we require the color space converters at both the ends.

A *Background subtractor* module is running both on the CPU and on the GPU. This module is new in the pipeline and is responsible for determining which pixels of a video belong to the foreground and which pixels belong

to the background. The background subtractor can also get input from the ZXY sensor tracking sub-system (Section 2.1.1) to improve the performance and precision. Even though we have enhanced the background subtraction with sensor data input, there are several implementation alternatives. When determining which algorithm to implement, we evaluated two different alternatives, i.e., Zivkovic [159, 160] and Kaewrakulpong [74]. Both algorithms use a Gaussian Mixture Model (GMM), are implemented in OpenCV and have shown promising results in other surveys [13]. In the end, Zivkovic provided the best accuracy which is important for our scenario and it was therefore selected.

There are also several modules that are running primarily on the GPU. The *Uploader* and *Downloader* are managing the data flow to and from the GPU. The Uploader transfers RGB frames and the background subtraction player pixel maps from the CPU to the GPU for further processing. The downloader transfers back the stitched video in YUV 4:2:0 format for encoding. Both modules use double buffering and asynchronous transfers.

The main parts of the panorama creation are performed by the *Warper*, *Color-corrector* and *Stitcher* modules running on the GPU. The warper module warps the camera frames and the foreground masks from the background subtractor module to fit the common panorama plane. Here, we used the Nvidia Performance Primitives library (NPP) for an optimized implementation. The Color-corrector is, in this implementation, added to the pipeline because we experimented with the exposure synchronization (section 3.4.3) at the same time as building this pipeline. During the synchronization experiments, to generate a best possible panorama video, we correct the colors of all the frames to remove color disparities. This operation is performed after the images are warped. The reason for this is that locating the overlapping regions is easier with aligned images, and the overlap is also needed when stitching the images together. The implementation is based on the algorithm presented in [155], which has been optimized to run in real-time with CUDA.

The stitcher module is similar to the homography stitcher in the loop implementation, where a seam is created between the overlapping camera frames. Our previous approach uses static cuts for seams, which means that a

fixed rectangular area from each frame is copied directly to the output frame. Static cut panoramas are very fast, but can introduce graphical errors in the seam area, especially when there is movement in the scene as illustrated in figure 3.10a. Thus, to make a better visual result, a dynamic cut stitcher is introduced. This module now creates seams by first creating a rectangle of adjustable width over the static seam area. Then, it treats all pixels within the seam area as graph nodes. Each of these edges' weights are calculated by using a custom function that compares the absolute color difference between the corresponding pixel in each of the two frames we are trying to stitch. The weight function also checks the foreground masks from the Background subtractor to see if any player is in the pixel, and if so it adds a large weight to the node. We then run a simplified version of the Dijkstra graph algorithm (only going up in the image) on the graph to create a minimal cost route from the bottom of the image to the end at the top. An illustration of how the final seam looks can be seen in figure 3.10b, where the seams without and with color correction are shown in figures 3.10c and 3.10d respectively.

3.4.2 Evaluation & Results

To evaluate the processing performance of the parallel pipeline implementation, we used a single computer with an Intel Server Adapter i350-T4 for connecting the four cameras with Gigabit ethernet, an Intel Core i7-3930K six core processor with 32GB RAM and a single Nvidia GeForce GTX Titan graphics processor.

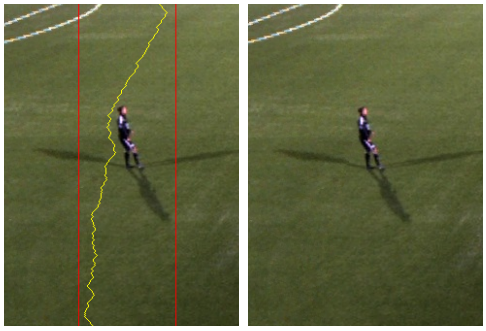
The overall performance of the parallel pipeline is shown in figure 3.11a. The CPU modules are marked in blue, and the GPU modules are marked in green. The Uploader and Downloader module run both on the CPU and the GPU, but we have chosen to mark them as CPU modules, since they both are controlled by the CPU.

Images from all four cameras are asynchronously transfered to the GPU as soon as the debarreling is complete. The number of threads and blocks on the GPU is automatically adjusted by the number of cores available on the GPU. The modules executing on the GPU are synchronized with barriers; when one

(a) The original fixed cut stitch with a straight vertical seam.

(b) The new dynamic stitch with color correction.

(c) Dynamic stitch with **no** color correction. In the left image, one can see the seam search area between the red lines, and the seam in yellow. In the right image, one clearly sees the seam, going outside the player, but there are still color differences.

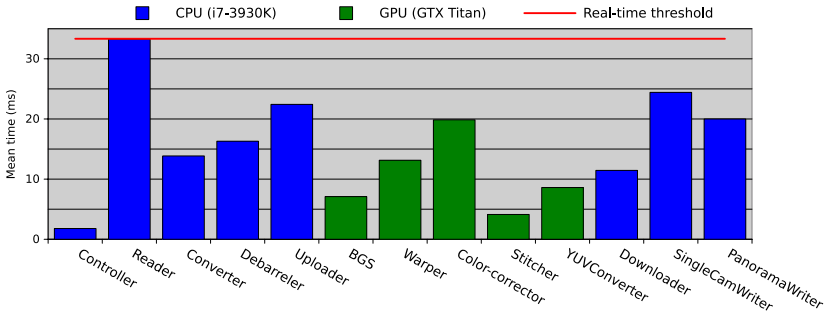


(d) Dynamic stitch **with** color correction. In the left image, one can see the seam search area between the red lines, and the seam in yellow. In the right image, one cannot see the seam, and there are no color differences making it seemless.

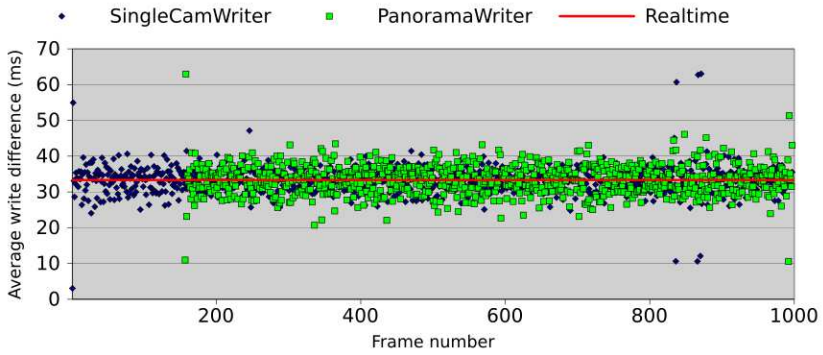
Figure 3.10: Stitcher comparison - improving the visual quality with dynamic seams and color correction.

module finishes, the next is started. Data is stored in global memory, and pointers to the data are transfered between the different modules. When processing is finished on the GPU, data is asynchronously transfered back to the CPU for encoding and writing to disk.

In figure 3.11a, we can see that when executing the whole pipeline, all modules perform well below the real-time threshold running concurrently on the same machine. Note that the reader module is limited by the cameras



(a) The processing performance of the different pipeline modules running concurrently on the same machine.



(b) Pipeline write differences (showing times for 1000 frames). Note that the delayed start of panorama writes is caused by the frame delay buffer implemented in the uploader module.

Figure 3.11: The processing performance of the parallel and distributed processingpipeline.

which produce a new frame every 33 ms. Also, the modules run in a pipelined fashion. Thus, since all modules perform better than the 33 ms threshold while sharing the resources, we are able to deliver panorama frames in real-time. It must be noted that the real-time execution of the entire pipeline is only guaranteed if all the modules running concurrently perform with in 33 ms. It can be possible for each of the modules running independently using all the resources of a machine to perform in real-time, however in such a case, the pipeline is not guaranteed to execute in real-time. This is further demonstrated by measuring the differences between the single camera writes and the difference between the panorama writes. In figure 3.11b, we present the write differences between the frames, and we observe that a new frame is

output every 33 ms, i.e., equal to the input rate of the cameras. These results show that our parallel and distributed processing implementation executes in real-time on a single off-the-shelf computer.

3.4.3 Exposure Synchronization

The ideal solution to ensure the photometric continuity would be to use the same physical camera and the same exposure settings throughout the panorama capture. However, when capturing panoramic videos, it is only possible to use the same physical camera if one uses a reflective sphere, but this approach results in reduced resolution because of limitations of a single video camera. When an array of multiple cameras produces images that are not captured using similar exposure parameters, there will be visual differences between adjacent camera images. Often this is overcome by color correction approaches, which handle the images post-recording.

Xu *et al.* [156] provide a good performance evaluation of several color correction approaches. Color correction approaches can be divided into two broad categories - parametric and non-parametric approaches. Any of the approaches can be further classified into global and local. A global approach is where the color transfer is assumed to be from an entire source image to an entire destination image, whereas, the local approaches apply the color transfer locally to parts of the image. The parametric approaches assume that there is a *color transfer model* and attempt to estimate the model parameters that are later used to transfer the color. The non-parametric approaches mostly rely on Look-Up Tables (LUT) between the color values.

Ibrahim *et al.* [66] provide an interesting approach for selecting the reference for color correction. They propose an automatic reference for color correction in panoramic video stitching. They design a cost function as sum of normalized transformed grayscale values over all channels in each image. Then the image with the least cost is selected. By doing this, the gain and offset in the color values are minimized. For the color correction itself, they employ a simple parameterized transform. However, there is no discussion about how flicker is avoided in the case of a video. There is a discussion

on performance. However the reported results are from MATLAB and only numbers for estimating the reference image are presented - not the total time taken for color correction.

Xiong *et al.* [155] proposed an elegant color correction approach which applies a color transform that is optimized over all the images to minimize drastic changes per image. The advantage of the approach is its low computational complexity. They employ their approach for mobile devices, however low computation complexity is also useful in case of real-time requirement in a panoramic video stitching.

Nevertheless, even though color correction approaches can provide good results in panorama images, they can introduce artifacts like flicker and unnatural colors when it comes to the stitched videos. Figure 3.12 shows one such effect. Here it can be seen that the color correction is trying to achieve a seamless field but instead introduces an unnatural green in the whole of the right part of the panorama. In a constrained space like a sports stadium, this problem can be handled even before the videos are captured.

Figure 3.12: Artifacts observed due to color correction occassionally.

Another challenge to handle the changing light conditions is to understand the light that is coming from the scene. In this respect, we use the internal metering mechanism to estimate the exposure parameters. The region of interest that is considered for metering can be modified for each camera. We make use of this functionality in the three exposure setting approaches described here. It must also be noted that, as far as we know, there exists no publications where the exposures are synchronized before recording the

videos. A common approach is to fix the exposures, but such an approach would fail when the light conditions change.

Approaches

Independent Metering This is the most trivial approach for an automatic exposure system. In this approach, we use the fact that the football field provides a nice surface for metering. Since our target space is confined to a football stadium, we can use the green surface of the football field to evaluate the exposure parameters. Initially, a manual selection of metering region is selected per camera, and the cameras are driven to make an automatic exposure. The internal mechanism decides on a specific exposure value and gain to achieve a pre-defined gray value for the average of all the pixels from the metering region. An upper limit can be imposed on the exposure time to force the camera to use a higher gain in case of low light, instead of increasing the exposure time.

Pairs Metering This approach can be considered as a special case of the Independent metering presented above. In this approach, we exploit the fact that the adjacent cameras have an overlapping region. Therefore, camera pairs are formed that have defined regions of interest pointing to the same physical space on the field. The selection of the regions of interest is performed manually to minimize the effect of the players. Then, the cameras are operated independently to perform automatic exposure, but the metering is based on the selected patches that are overlapped. Since the camera pairs are physically close to each other, the directional reflections have minimum effect on the exposure. However, the first camera pair and the second pair are at a distance of 4m from each other.

Pilot Camera Approach In this approach, there is a pilot camera that functions in auto-exposure mode, and the pilot camera's exposure parameters are transfered to the other cameras. Let the m cameras be named C_j where $j \in [1, m]$, and C_p be the pilot camera. Let e_j and g_j be the exposure time and gain of camera C_j.

Then, given e_p and g_p from the pilot camera, which operates in auto exposure mode, we need to compute e_j and g_j for the rest of the cameras. Furthermore, let T_j be the transformation function from the pilot camera to camera C_j. Then,

$$(e_j, g_j) = T_j(e_p, g_p). \tag{3.1}$$

The transformation function depends on the relation of camera C_j to the camera C_p. In an ideal situation, where the cameras are all identical and have exactly the same settings for aperture and focal length, T_j is an identity function. However, this is not the case because physically different cameras do not have identical spectral response curves thus leading to difference in exposures. Other factors that can cause differences are the imperfections in adjustment of the aperture size, because they are physically adjusted by rotating the aperture rings on the camera in the Bagadus system. The cameras need a prior calibration step to estimate the corresponding transformation functions.

The general processing flow is presented in figure 3.13. There are two types of threads that are running concurrently: one is for controlling and communicating with the pilot camera, and the other type is for the rest of the cameras. All these threads have a synchronization barrier at the end of every frame. Periodically, the pilot camera thread sends a trigger to the pilot camera to make an auto exposure and lock the exposure settings until the next trigger. In figure 3.13, this can be seen before acquisition of frame n. After the exposure, the exposure parameters e_p and g_p are transferred back to the controlling machine. These parameters are communicated to other threads which in turn transfer these individually to the other cameras applying the appropriate transformation before setting the exposure on the corresponding camera.

It can be observed that the frames n of the other cameras are not synchronized in exposure with the pilot camera, but we have observed empirically that the light conditions change slowly over the period of the exposure updating trigger. One more important detail is that the frame rate sets a hard upper bound on the execution time and thus on exposure time too. The for-

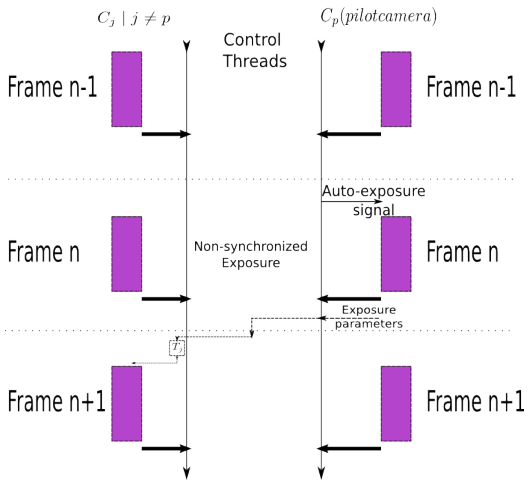


Figure 3.13: Pilot camera approach.

mulation of the transformation function cannot guarantee this because one of the transformations can demand a higher exposure time than the upper limit. This can occur especially, when the rest of the cameras have lower response to light than the pilot camera. This problem can be handled in two ways: one way is to embed this property into the transformation function by placing an upper bound, the other way is to handle it in the driver before setting the camera parameters. We found that the driver solution is safer and more robust to further changes in the algorithm.

Results

In this section, we present some experimental results showing the visual differences between the approaches and the importance of a synchronized exposure when generating panorama images or video frames. The panorama images presented here for each approach are to emphasize the different lighting conditions. First, we present images recorded during different lighting conditions that emphasize the differences in the approaches in figure 3.14. Then, we also show the results using the three approaches from the same match in the same lighting condition for a fair comparison in figure 3.15.

Figure 3.14a shows a scenario where there is snow around the football field. The metering system has to compensate for this and make a good choice

(a) Independent metering approach in a snow condition.

(b) Pairs metering approach under a partially cloudy sky.

(c) Pilot camera approach under an overcast sky.

Figure 3.14: Panorama generated using different approaches under different light conditions

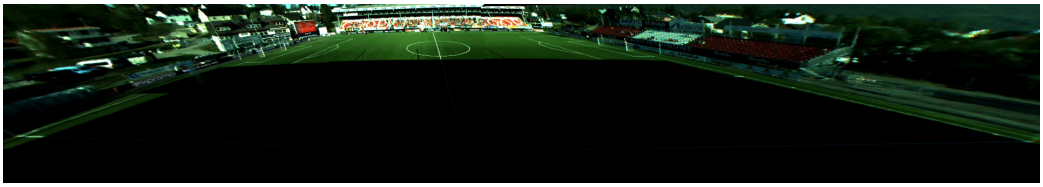
of exposure values. The influence of snow can be observed in the independent metering approach. The problem is that the exposures are different in each of the cameras, even though each of the images is well exposed, they are not synchronized introducing large visual differences in the generated panorama image.

Figure 3.14b shows the pair metering approach in one of the possible light conditions, i.e., when the sky is partially cloudy. In this approach, a clear difference can be seen at the center of the field due to the pairwise exposure settings. However, the left and the right camera-pairs (using the same region of interest for metering) of the panorama are perfectly seamless.

Figure 3.14c shows the pilot camera approach using another lighting condition where there is an overcast sky. Here, it can be observed from the figure that the exposure in the whole of the panorama is perfectly synchronized as there are no visual differences between the different parts in the stitched image. There is no specific color-correction applied when stitching the panorama.

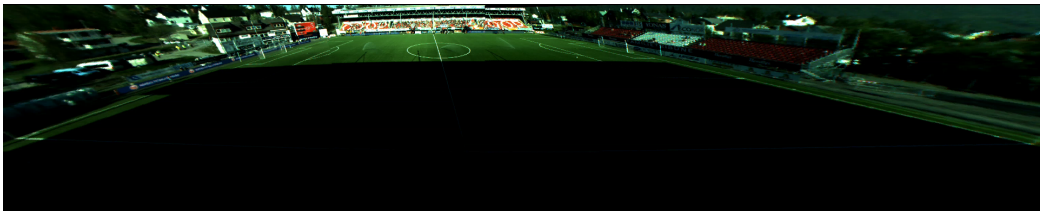
In the next experiment, we present frames using the three approaches dur-

ing a similar time period for comparison (figure 3.15). This is also one of the hardest light conditions to handle, when there is direct sun on the stadium and the roof of the stands cast a sharp shadow in the middle of the field. In such a case, the camera's dynamic range is insufficient to capture variation in the light and dark areas. It can be observed that the first and second (independent and pair) approaches provide rather similar result whereas the third (pilot camera) approach provides a seamless result. This similarity between the first two approaches has been observed in different light conditions as well. These results therefore confirm the previous results showing the importance of a synchronized exposure for multi-camera generated panorama images.



(a) The independent metering approach.

(b) The pairs metering approach.



(c) The pilot camera approach.

Figure 3.15: Generated panoramas under equal ligthing conditions

3.5 Parallel Real-Time Pipeline with Upgraded Setup - Version 3

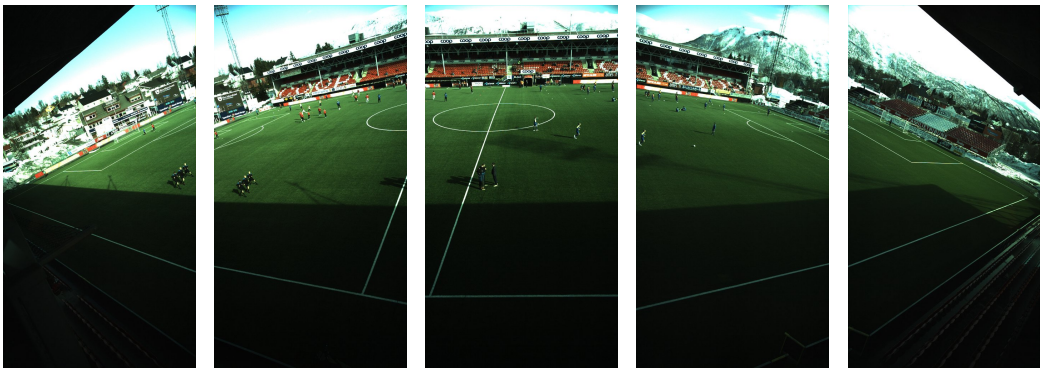
The previous section discusses the real-time panorama video creation pipeline for the old camera setup. The changes introduced in the new camera setup (from section 3.5.1) affect the pipeline. In this section, we discuss the changes made to the pipeline in terms of added and removed components from the old pipeline. The implementation details are not highlighted, as the new pipeline retains much of the old pipeline in terms of implementation. Only the key differences are highlighted in this section.

3.5.1 Upgraded Setup

The distance between the optical centers of the old camera setup can cause huge ghosting problems when players run from one camera view to the next one. Moreover the resolution of the panorama in the old setup is much lower than the sum of number of pixels from 4 cameras because of the large overlap among the cameras due to use of wide-angle lenses. Hence we focussed on these issues and decided to use better cameras (2046×1086 pixels) and narrower lens (8mm). Table 3.5 and Table 3.6 provide the details about the new camera and lens respectively. To further improve the vertical resolution of the panorama, the cameras are rotated by 90 degrees because the cameras have approximately twice as many pixels horizontally as that are vertically. We tackled the problem of ghosting by bringing the cameras close to each other and arranging them in a fashion that their optical centers coincide as shown in figure 3.16a. The output from the new cameras is presented in the Figure 3.16.

One other advantage of the new cameras is that they can capture at 50fps. However, the Gigabit interface is not capable of transferring the full resolution at 50 fps when sent in multiple color channels. The cameras capture in a Bayer pattern which, is a single channel from which multiple color channels are interpolated on the camera. We instead perform this interpolation outside the camera, so that we reduce the bandwidth required on the Gigabit

(a) New Camera System Setup.



(b) Camera 1. (c) Camera 2. (d) Camera 3. (e) Camera 4. (f) Camera 5.

Figure 3.16: Views from each camera for a perfectly synchronized exposure.

interface. However, we faced new challenges because of double the amount of data to be processed and adding further processing steps to the pipeline. So, we decided to distribute the pipeline on several machines.

Manufacturer	Basler
Model	acA2000-50gc
Sensor formate	2/3”
Pixels	2046×1086
Frame Rate	50

Table 3.5: Basler acA2000 Camera.

Manufacturer	Azure
Model	0814M5M
Focal Length	8 mm
Aperture	F1.4 - F32

Table 3.6: Azure 8mm Lens.

3.5.2 Key Differences to Old parallel pipeline

The new camera and lens system has practically no barrel distortion in the image. A line of length 2048 pixels deviates by 1 pixel atmost from the

straight line. This enables us to remove the debarreling step from the old pipeline. Moreover, the exposure synchronization is also constructed into the pipeline, and hence, the color correction component, which was optional in the parallel pipeline version-2, is now completely removed from the new pipeline.

Furthermore, the new cameras enable capturing video at 50 fps at full resolution. However, the Gigabit interface limits the possibility to only a single 8-bit channel. Hence, we use the Bayer format to retrieve color data using only the bandwidth used for a single 8-bit channel but at full resolution and at 50fps. This introduces a need for *debayering* filter in the pipeline. In addition, we added a *High Dynamic Range (HDR)* module to the pipeline to handle challenging light conditions.

Another key difference to the earlier pipeline is that the new pipeline runs in a distributed fashion as seen in figure 3.17. We use different machines to capture the frames from the cameras, the frames are then transferred to a processing machine and then the processing machine stitches the panorama. However, transferring raw frames from one machine to other machine requires high bandwidth and low latency to perform in real-time. To enable this, we used hardware from *Dolphin Interconnect* [33]. The adapters enable communication between multiple machines using PCI Express.

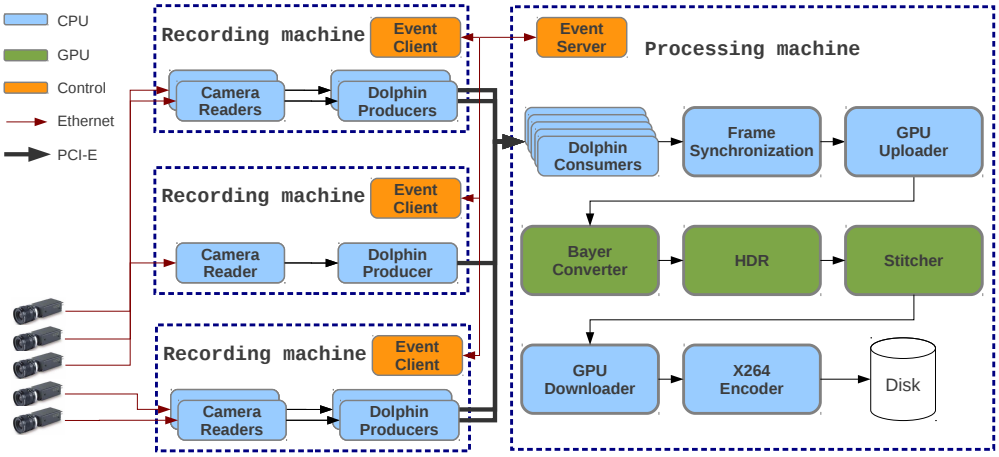


Figure 3.17: Real-Time Pipeline with Upgraded Setup

3.5.3 Debayering

When an image is captured in color on a sensor, it is challenging to build a sensor that is capable of recording all color values in a single pixel. Therefore, most modern color sensors use a simple intensity sensor along with a Color Filter Array (CFA) to capture a color image. The most common pattern for a CFA is Bayer. An example of a Bayer pattern is presented in figure 3.18. Here one can observe that, at each pixel position, only one of the three colors is recorded. The pattern is then used to interpolate all the three color channels at every pixel. This process is called *debayering*.

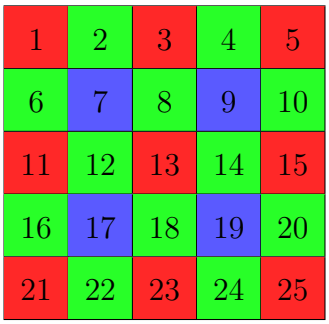


Figure 3.18: A bayer pattern

Several debayering algorithms exist in the literature, e.g., [2, 20, 56, 79, 91, 94, 101], and below we have selected algorithms that we deemed most promising considering our real-time requirements. We will be referring to figure 3.18 in example equations, identifying each pixel with a number and each color value with R, G or B, for example, $R1$ is the top-left corner pixel.

Bilinear Interpolation This uses the average value of the two or four nearest neighbour pixels of the specific color, e.g.,

$$B8 = \frac{B7 + B9}{2} \qquad B13 = \frac{B7 + B9 + B17 + B19}{4} \tag{3.2}$$

It is generally considered the cheapest of the acceptable algorithms, often used in real-time video systems due to its low complexity. Therefore, we have included this as our baseline algorithm.

Smooth Hue Transition This is a two pass algorithm [23] that first uses bilinear interpolation to reconstruct the green channel. The second pass uses the relation between the green channel and the red/blue channel within a pixel to reconstruct the remaining channels, e.g.,

$$B13 = \frac{G13}{4}\left(\frac{B7}{G7} + \frac{B9}{G9} + \frac{B17}{G17} + \frac{B19}{G19}\right) \tag{3.3}$$

This utilizes the principle that the difference between two channels within a pixel only changes gradually and rapid transitions cause visual artifacts.

High-Quality Linear Interpolation This is a single pass algorithm [94] that performs bilinear interpolation, but uses the color information already present in the pixel to correct the result, e.g.,

$$\Delta r = R13 - \frac{R3 + R11 + R15 + R23}{4} \tag{3.4}$$

$$G13 = \frac{G8 + G12 + G14 + G18}{4} + \frac{\Delta r}{2} \tag{3.5}$$

If the interpolated red value differs significantly from the real red value, there is likely a significant change in luminosity in this pixel.

Edge Directed Interpolation This is a two pass algorithm [2] that tries to avoid interpolating across edges, averaging two widely different values. It uses the laplacian, i.e., the divergence of the gradient between enclosing pixels, of the green channel and the gradient of the red or blue channel to determine the presence of an edge when reconstructing the green channel. The horizontal gradient is determined by

$$Grad13_H = |G12 - G14| + |2R13 - R11 - R15| \tag{3.6}$$

and the vertical similarly. The algorithm performs linear interpolation of either the two enclosing vertical samples, or horizontal, depending on the smallest gradient. When interpolating the red and blue channel, it performs linear interpolation with the correction from [94].

Homogeneous Edge Directed Interpolation This is a three pass algorithm, designed as a simplification of the adaptive homogeneity directed demosaicking algorithm [56]. When interpolating in only one direction it may be visually apparent if single pixels choose a different direction compared to neighbouring pixels. This algorithm therefore computes the directional gradients in the first pass, before selecting the direction based on the local directional preference in a second pass. The final pass for interpolating the red and blue channel is equal to that of the edge directed.

Weighted Directional Gradients This is a two pass algorithm [91] that uses a weighted average of pixels in four directions in the initial green interpolation, weighted based on the inverse gradient in its direction. The algorithm determines the value contribution G of each direction right/left-/up/down, and its weight α. For example, the right direction of pixel 7 is determined by

$$\begin{aligned} G_r &= G8 + \frac{B7 - B9}{2} \\ \alpha_r &= \frac{1}{|G6 - G8| + |G8 - G10| + |B7 - B9| + \frac{|G2-G4|+|G12-G14|}{2}} \end{aligned} \tag{3.7}$$

similarly for each direction. The final green value can be computed by

$$G7 = \frac{\alpha_l G_l + \alpha_r G_r + \alpha_u G_u + \alpha_d G_d}{\alpha_l + \alpha_r + \alpha_u + \alpha_d} \tag{3.8}$$

This is performed similarly when interpolating the red and blue channel, while then also taking advantage of having the full green channel. It performs a similar directional weighted average independently for each channel.

Implementations

To improve performance, the algorithms above have been implemented on CUDA and optimized for the Kepler architecture. We have set focus on optimizing the algorithms for execution speed, not memory requirement.

The algorithms are all implemented using the same base principles, as they are primarily differentiated by the number of required passes and the number of pixel lookups per pass. Every kernel is executed with 128 threads

per CUDA block, the minimum required to allow maximum occupancy on the Kepler architecture. Every active kernel is also able to achieve more than 95% occupancy. The initial Bayer image is bound to a two-dimensional texture, giving us the benefit of two-dimensional caching when performing multiple texture lookups per pixel. The use of textures is essential, as many of the algorithms would be difficult to implement with good memory coalescing using the GPU's global memory.

In most kernels, we tried to perform as few texture lookups as possible and to rely on temporary storage when using the same pixel multiple times. However, with the original weighted directions, this increased the local register requirements for each thread, reducing the number of concurrent threads that could execute. Instead, we observed better results when performing duplicate texture lookups.

Most of the algorithms utilize multiple passes, most commonly, an initial green pass followed by one red and blue pass. These are implemented in nearly the same way, using a temporary texture with two bytes per pixel, for saving the green value and either a red, blue or empty value. Using a single texture for this provides much better data locality and cache efficiency, increasing performance significantly over using two separate textures. In order to write the temporary values, we utilize surface memory in CUDA.

The homogeneous edge directed algorithm uses two passes to interpolate the green channel. In the first pass, the green value is computed both based on the horizontal and the vertical interpolation method. Additionally, we calculate the directional preference. These values, along with the original red/blue value is written to surface memory with 4 bytes per pixel. It proved faster to keep this data localized in one array, despite having to perform nine texture lookups when we determine the localized directional homogeneity.

The original weighted directional gradients approach uses two passes to interpolate the red and blue channels. The second pass fully interpolates the red and blue pixels, leaving the green pixels untouched. This data is then used in the third pass to complete the remaining red and blue values. This implementation uses a full four bytes per pixel to ensure data locality for the final pass, but this may not be ideal. It is generally considered more

efficient to use four bytes per pixel instead of three, due to memory alignment, but in our case, we have only half the pixels carrying three values and the other half (green pixels) carrying a single value. We opted to implement two variations of this algorithm, the original and a modified version that borrows the constant hue correction-based approach of the edge directed algorithms.

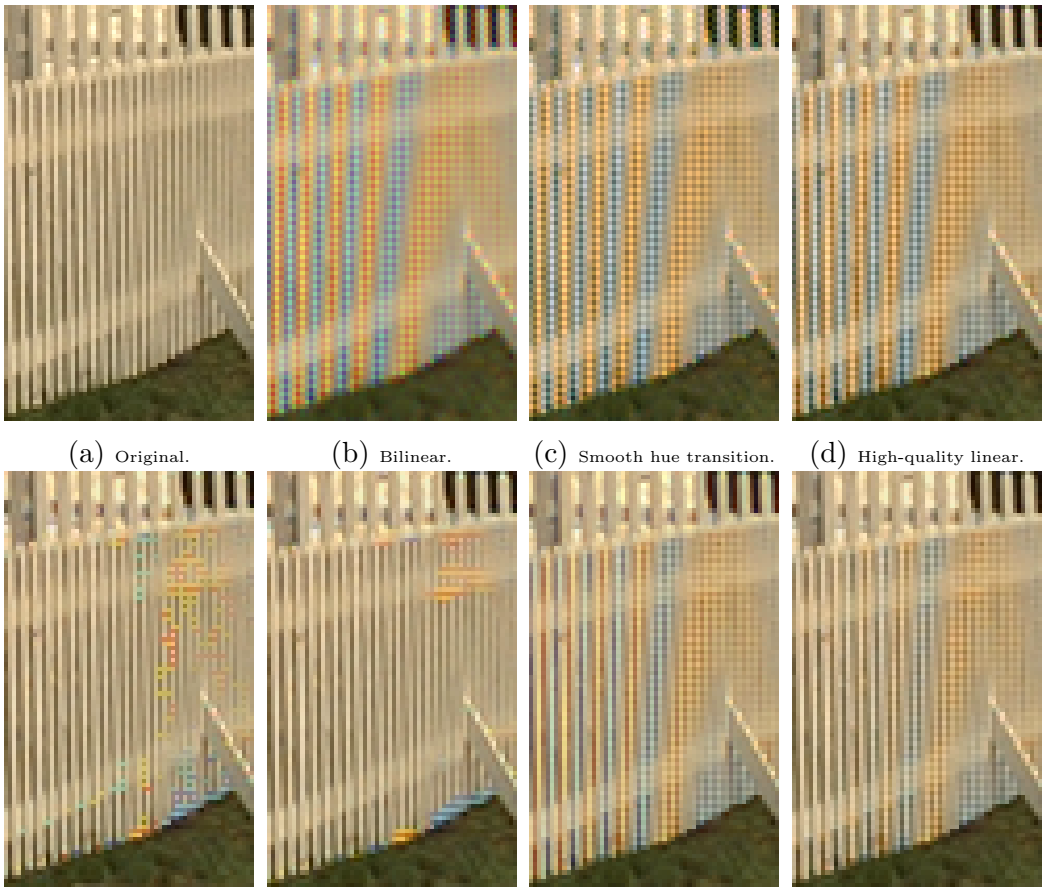
When implementing the kernels it was essential to avoid branching code, based on the color of each pixel. A naive approach would run the kernel on each pixel and perform one of four branches, depending on the color of that pixel. Because each branching operation within a single thread warp must be executed by all threads in that warp, it would be guaranteed that at least half of the executing threads would idle due to branching. Instead, our kernels process 2×2 pixels in each iteration. This introduces zero branching as a result of selecting pixels. These four pixels also need to access a lot of the same pixels, so we load these at once for local computations.

Experimental Results

To compare the different algorithms and implementations, we performed a number of experiments that we will present in this section.

Visual Quality We evaluated the visual quality of each algorithm by subsampling existing images, imposing the bayer pattern, and see how accurately the image can be reconstructed. When reconstructing the images, we typically see interpolation artifacts, primarily in the form of false colors and zippering along edges. The zippering stems primarily from the green interpolation, while false colors typically stem from the red and blue interpolation. However, since most algorithms use the green channel when interpolating red and blue, incorrect green interpolation is normally the prime cause of false colors for these algorithms. Figure 3.19 shows how each algorithm handles a region prone to false colors.

Peak signal-to-noise ratio (PSNR) is a simple metric for evaluating image reconstruction. We computed the PSNR of each of the reconstructed images, filtering away homogeneous areas that rarely produce visible errors with an edge detection filter. Although PSNR can yield inconsistent results



(a) Original. (b) Bilinear. (c) Smooth hue transition. (d) High-quality linear.

(e) Edge directed. (f) Homogeneous edge directed. (g) Weighted directions original. (h) Weighted directions modified.

Figure 3.19: Visual assessment of zippering and false colors.

with many image transformations, we saw a very strong correlation between the PSNR and the visual result. High PSNR in the green channel was common in those algorithms that avoided zippering artifacts and maintained the greatest level of detail. Low PSNR in the red and blue channel normally meant a high level of false colors. In table 3.7, we observe a summary of the visual assessment of the implemented algorithms. We see that the best performing algorithms all use the same, simple final pass for interpolating the blue and red channels. This shows that if the green channel is accurately reconstructed, we can use the concept of constant hue to reconstruct the more sparsely sampled channels.

Algorithm	PSNR		Zippering	False colors	
	Green	Red/blue		Frequency	Intensity
Bilinear	28.43	23.51	Very strong	Very high	Very strong
Smooth hue transition	28.43	27.07	Very strong	High	Strong
High-quality linear	34.44	29.67	Strong	High	Strong
Edge directed	35.61	34.62	None	Very low	Strong
Homogeneous edge directed	36.22	34.89	None	Very low	Strong
Weighted directions original	37.97	31.02	Very weak	Medium	Medium
Weighted directions modified	37.97	36.25	Very weak	Low	Weak

Table 3.7: Summary of each algorithms visual performance. We rank the selected algorithms based on the best and worst within each category.

Performance The primary requirement in our real-time panorama system is the overall execution time of the algorithm. The algorithms presented have quite a varying degree of computational complexity, but this is not necessarily the only requirement for performance efficiency. Table 3.8 shows the mean execution time of each algorithm for two different GPUs.

We see that most algorithms are nearly equally fast, and all algorithms are within our 20 ms real-time limit on a GTX680. The original weighted directions proved extremely inefficient, due to its slow red and blue channel interpolation. However, we saw that our modified algorithm produced better visual results at a lower processing cost. The execution time seems to be primarily determined by the number of texture lookups required, with an added penalty for each pass. An exception appears to be the second pass of the smooth hue transition, which is slowed down by having to perform four divisions per pixel.

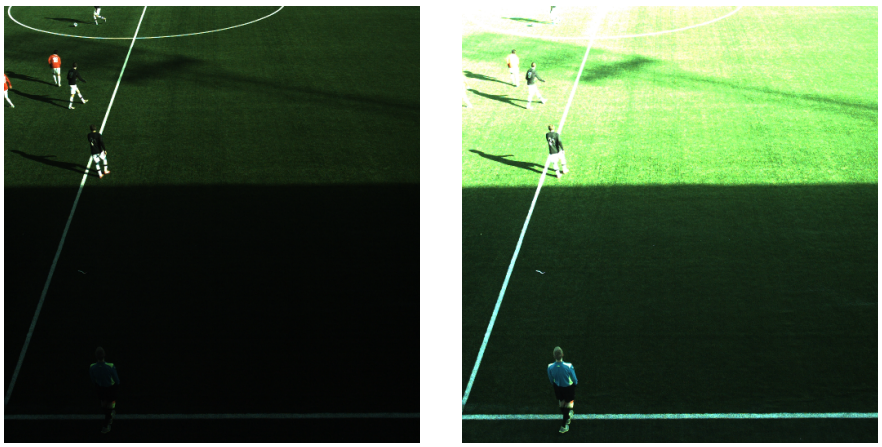
3.5.4 High Dynamic Range (HDR)

An example of challenging light conditions can be seen in figure 3.20. Here, we can see that the harsh sun-lit and the shadow regions provide a lot of contrast that it becomes impossible to capture details in both regions using the same exposure. This is mainly caused by the dynamic range of an image. We consider these images as Low Dynamic Range (LDR) input images and we desire to fuse them into a High Dynamic Range (HDR) image where all

Algorithm	Quadro K2000 (μs)	GTX680 (μs)
Bilinear	5516	929
Smooth hue transition	10183	1979
High-quality linear	6370	1073
Edge directed	10941	2025
Homogeneous edge directed	20029	3184
Weighted directions original	49212	9061
Weighted directions modified	19052	3094

Table 3.8: Summary of each algorithms performance. Execution time is measured with a 2040×5400 resolution image.

the details are preserved in a single image.



(a) Low exposure. (b) High exposure.

Figure 3.20: Input images for the HDR processing.

An HDR image can be created by a process called radiance mapping, where multiple LDR images of the same scene are fused together. The fused image as such cannot be displayed using the conventional devices, so a compression of the dynamic range is performed where we try to preserve details in all regions and to maintain a local contrast, i.e., giving a visually pleasing result. This process is called tone mapping. For the rest of the paper, we refer to the final output of tone mapping as the HDR image. We explored three approaches for radiance mapping, and three for tone mapping. We chose radiance mappers based on the criterion that they must merge multiple exposures. For the tone mappers, we picked a representative subset: one

very simple to serve as easy comparison to others, one global and one local tone mapper. After extensive material research, we picked algorithms that provided the most pleasing results in a small user study.

Radiance Mappers

Debevec One of the most cited approaches is presented by Debevec *et al.* [30], where the authors try to recover HDR radiance maps from photographs. It performs an estimation of camera response function and then a weighted selection process where the information is extracted from the mid-tone regions of different exposures. We made a GPU implementation for the second step alone, since the first step needs to be performed once in a lifetime for a camera.

Robertson Similar to *Debevec*, Robertson *et al.* [126] proposed a two-step mapping with similar modules. However, they introduced a more extensive approach for recovering the camera response function. Thus, the significant changes are only to the offline step of the algorithm.

Tocci As opposed to the previous approaches, Tocci *et al.* [144] proposed a solution that consists of a single online step. The main assumption is that unless close to saturation in intensity, the preferred output pixels are those from the high exposures. Therefore, they introduce an approach that takes saturated pixels from the neighbourhood into consideration. Since this approach requires fetching the same pixels by different threads with 2D locality, we use the GPU's texture memory to exploit the spatial caching feature.

Tone Mappers

Ward Ward *et al.* [149] proposed an approach where a global scale factor is applied to each pixel, which is dependent on, among other parameters, the average brightness of the input image. A parallel reduction approach [53] for

calculating the average over an entire image was implemented as part of our GPU implementation.

Larson This algorithm proposed by Larson *et al.* [85] performs tonal compression by creating a look-up table per frame to represent a desired histogram. Unlike a simple histogram equalization, the target histogram is computed taking human contrast sensitivity into account.

Reinhard Reinhard *et al.* [121] try to emulate a technique called "dodging & burning" [1]. This approach relies on the information from local neighbourhood for tonal compression. Adaptive Gaussian kernels are employed along different dimensions to average the exposure value, the adaptive nature is that the size of these kernels depends on the local contrast changes.

Experimental Results

The computer used for the experiments has a six-core Intel Core i7-3930K at 3.2 GHz with 32 GB quad-channel DDR3 memory and an Nvidia GeForce GTX 680 graphics processing unit with 3GB memory based on the Kepler architecture. The HDR module can be seen as part of the processing pipeline in figure 3.17. We show some experiments regarding quality and performance of the HDR module in the pipeline.

Visual Quality In this section, we provide a subjective assessment of the different configurations of our implemented radiance and tone mappers. The input images displayed in figure 3.20 show a small part of our panorama frames to highlight details. Then, the output of all possible combinations can be seen in figure 3.22.

We performed a limited user study to assess the quality of the different configurations of algorithms. Here, we asked 12 participants to rate videos obtained by combining the different algorithms. Several emerging patterns can be observed from the results shown in Figure 3.21. The first concerns the choice of tone-mappers. The average score achieved by *Robertson's* is

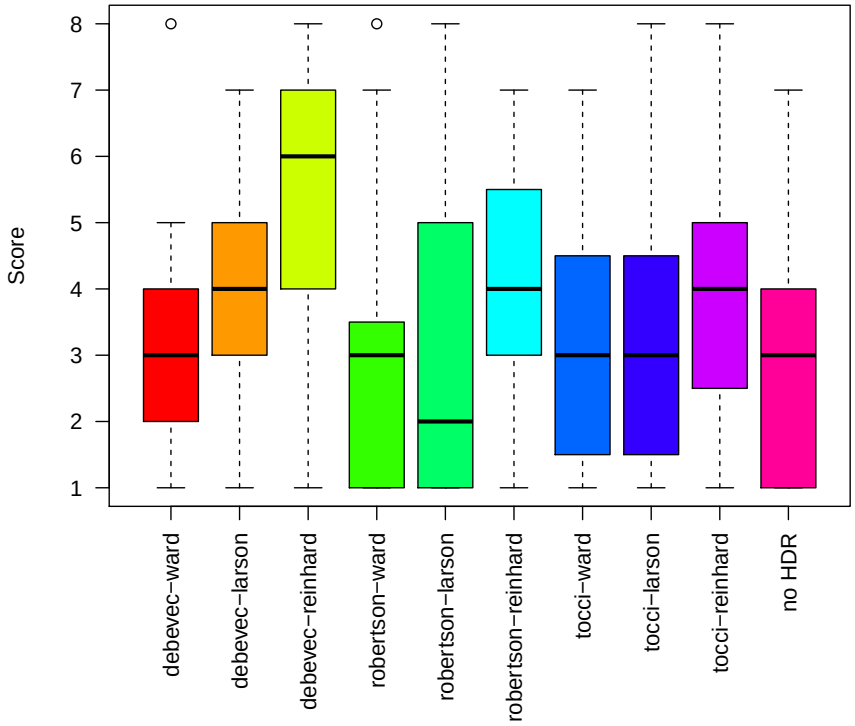
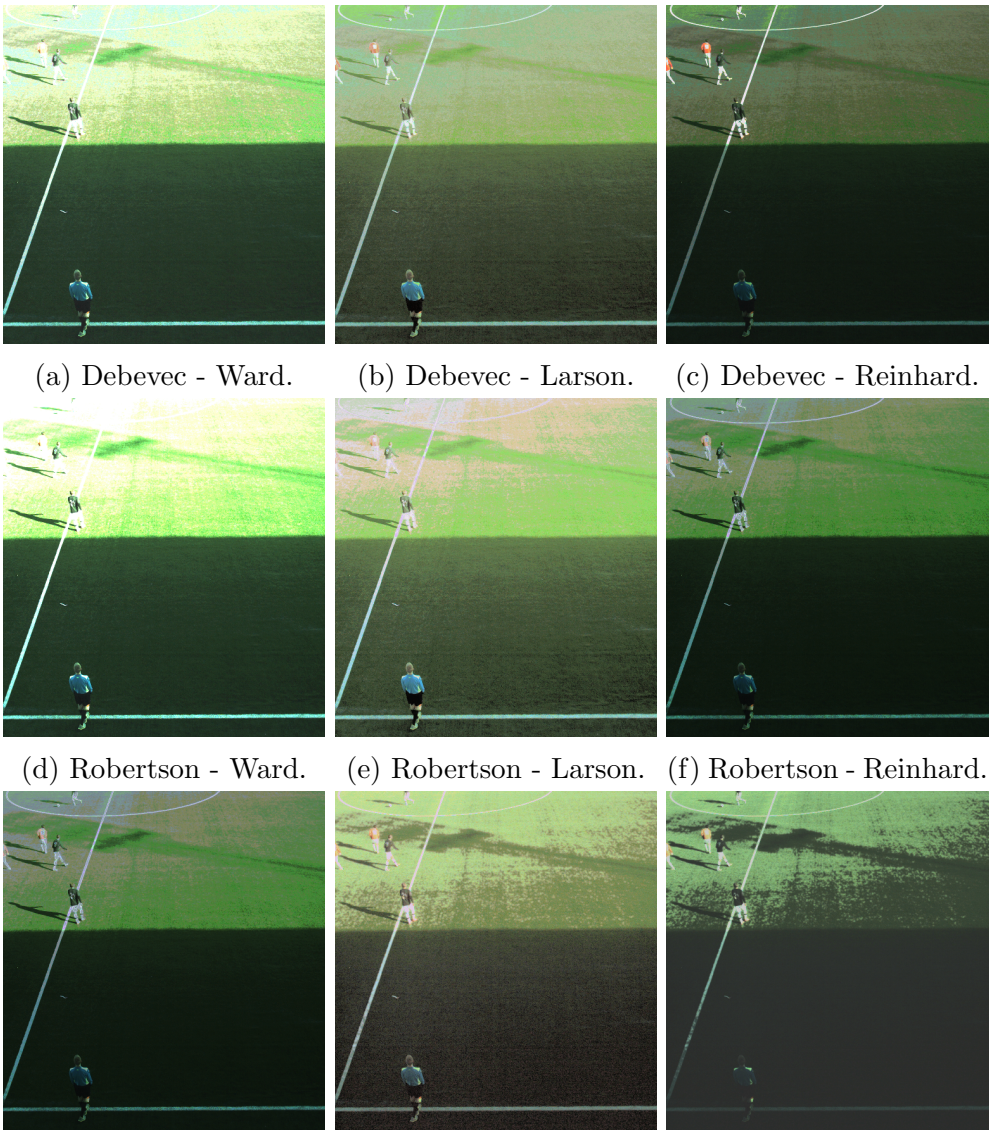


Figure 3.21: Results of the user study

higher than for the other tone-mappers in every combination with radiance-mappers. A second, less clear trend can be observed for the radiance-mappers. *Debevec's* always scores similar or higher than the other proposed radiance-mappers when combined with different tone-mappers.

Execution Time We accomplished real-time performance on all algorithms except with *Reinhard's* tone mapper. The multiple FFTs for applying the Gaussian kernels in this algorithm are performed per frame. Although we use the highly optimized CUFFT library provided by Nvidia [109], the multiple FFTs still turned out to be a bottleneck. Furthermore, *Reinhard* is consuming a lot of memory. For each stored Gauss-kernel, approximately 126 MB have to be allocated. Furthermore, the combination *Tocci* - *Larson* tends to miss the real-time dead-line occasionally.

A detailed listing of execution times can be found in Figure 3.23. We also executed only the HDR modules without interference from other modules. Those execution times can be seen in Figure 3.24.



(a) Debevec - Ward. (b) Debevec - Larson. (c) Debevec - Reinhard.

(d) Robertson - Ward. (e) Robertson - Larson. (f) Robertson - Reinhard.

(g) Tocci - Ward. (h) Tocci - Larson. (i) Tocci - Reinhard.

Figure 3.22: Visual quality after HDR for the different algorithms using the input images in figure 3.20.

Each of the modules contains several kernels. The scheduling of these kernels on the GPU is managed by CUDA. So, it must be noted that the execution times include the overheads created by scheduling. It can be seen that the execution times of other modules fluctuate when different HDR algorithms are employed. This is from the fact that CUDA schedules the

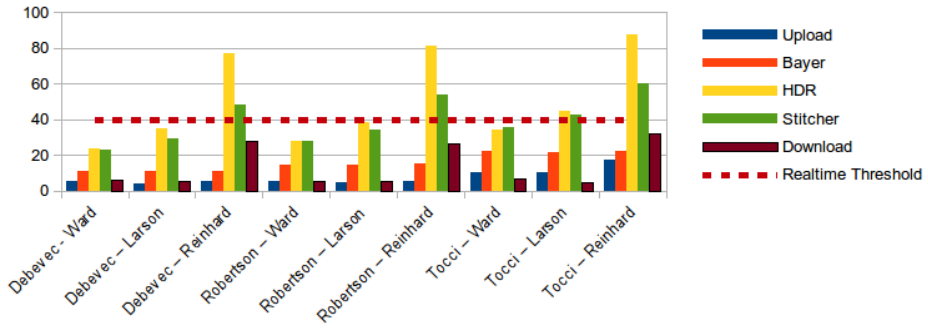


Figure 3.23: Execution times (ms) of the various modules in different configurations.

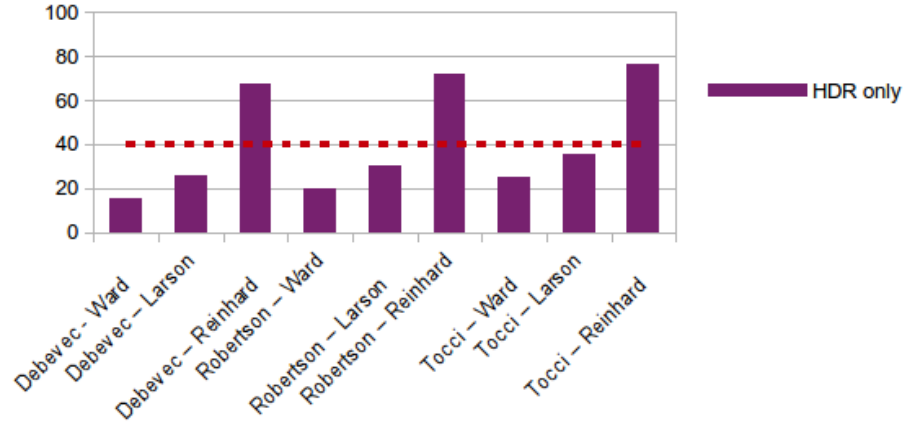


Figure 3.24: Execution times (ms) of the HDR module in different configurations.

kernels from these modules in different ways depending on the requirements of different modules. The GPU used in our experiments are capable of executing up to 16 kernels concurrently.

Summary

We believe that using *Debevec's* radiance mapper paired with *Larson's* tone mapper is the best approach taking the visual quality and execution overhead into account. Although *Reinhard's* tone mapper shows promising results, we could not make it pass the real-time requirements, and it also consumed a lot of memory on the GPU thus affecting other components of the pipeline. *Tocci's* radiance mapper proved to be too complex to be parallelized effi-

ciently using CUDA. Also, in our case, it produced undesirable artifacts. Lastly, *Robertson's* radiance mapper also proved to be very good, but the higher dynamic range it produced caused worse results when tone-mapped. The *HDR* module provides useful results when the light conditions are so severe that the normal video is rendered useless.

3.6 Discussion

In this chapter, we have explored our system used for real-time panorama video recording. Initially, we describe the evolution of our camera setup and how the upgraded camera setup helped to increase the visual quality of the panorama output.

Later, we provide an extensive discussion about the panorama creation itself. Our initial experiments suggested that real-time performance by processing on CPU alone might not be achievable. Then, we described our first real-time pipeline with the initial camera setup. In this pipeline, we addressed the performance and the quality of the output panorama.

When the camera setup is upgraded, we introduced new challenges in the processing. We used distributed processing as a way of solving the processing challenges. Furthermore, we introduced *debayering* and HDR modules to the new pipeline. These are well studied to provide a good quality panorama and also the real-time performance.

In the Bagadus system, we use the tracking information for two operations. One is during the creation of the panorama to assist the background detection. The location of players can help the background detection process to reduce the noise by providing regions of interest around the players. The second place where the tracking data is used, is during the retrieval. Chapter 5 details on using the tracking data to automatically create videos for the user based on their preference of player.

It must be noted that, we attempted to gain the best performance and high resolution panorama output using the cameras that were available during the development of the project. However, the progress in camera technology is quite rapid. Our research results can easily be extended to the new

higher resolution cameras. Moreover, we already introduced the concept for distributed processing, which can be used to extend the number of cameras as well.

3.7 Summary

Even with the new camera technologies, capturing a high quality panorama video at high framerate and high resolution, and making it available to be consumed in real-time with constant delay is still a demanding task. We have developed an inexpensive system using multiple cameras and commodity hardware for processing. In this chapter, we have gone through the development of the panorama capture system in detail. We also briefly mentioned the tracking sub-system used to capture player tracking data. In the next chapters, we elaborate on ways to use the captured panorama data to provide live interactive video services to a user. The next chapter focuses on creation of virtual PTZ cameras on a client device without much processing power using the high resolution panorama created using the pipeline version-3 in this chapter.

Chapter 4

Virtual View

> Virtual reality is a self-created
> form of chosen reality.
> Therefore, it exists.
>
> *Joan Lowery Nixon*

Exploring virtual environments has been a well-established topic in the graphics community. Some often used applications include games, and virtual tours of museums, stadiums and other building interiors. Until recently, most of these scenes were created by designers manually. However, using real-world captured scenes to build the virtual world becomes more popular, and in this regard, the most common applications now are using panoramic images as a scene. We would like to explicitly mention that we are not creating any 3D information from the football field.

In this chapter, the problem is to find efficient ways of extracting a personalized view of the soccer field by controlling their own virtual camera Using the cylindrical panorama from chapter 3 as an intermediate representation, our system aims to generate an arbitrary virtual camera view from the position of the camera array. As shown in figure 4.1, the virtual camera view is corrected to a perspective view very similar to that of a physical camera. The user has the full freedom to pan, tilt and zoom.

This chapter is an extension of [42] (Appendix G) and [45] (Appendix I).

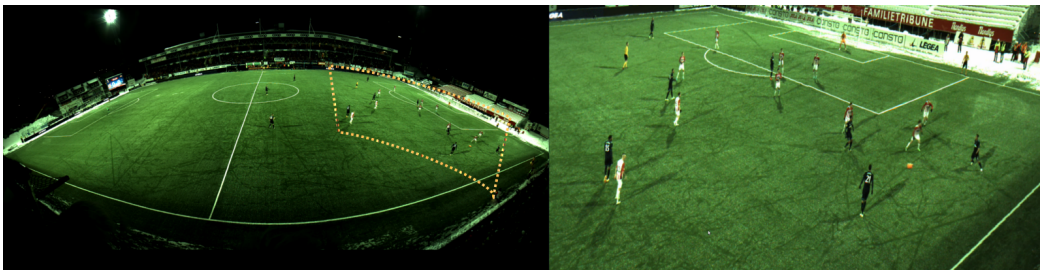


Figure 4.1: Panorama video with labeled ROI (left) and the virtual camera generated (right). It can be observed that it is not a simple crop from the bigger video.

4.1 Pan-Tilt-Zoom (PTZ) Camera

With the advent of technology advancements in control technology and the low costs of camera units, a type of cameras have emerged called Pan-Tilt-Zoom (PTZ) cameras. The cameras have exactly the functionalities mentioned in the name. They are fixed at one point called *viewpoint*, and it remains the same during the entire operation. However, they can be remotely operated to follow and focus on interesting parts of the scene. They provide high resolution images of the target while keeping the entire space reachable. These devices are widely used in surveillance. Figure 4.2 presents an example of a PTZ camera along with the possible space that can be captured. It must be noted that the PTZ cameras usually have freedom to capture any limited part of that space, but not the entire space.

Inspired from the physically moving PTZ cameras, a virtual PTZ camera can be realized. The functionality of a virtual PTZ camera remains similar to that of the actual PTZ camera. However, the operations are performed virtually on an ultra high-resolution image. Often, these are used to reduce the bandwidth of transmitted footage by cropping only the interesting parts of the image.

In the previous chapter, we discussed various possibilities to capture a wide field-of-view image and we ended up with a distributed video processing pipeline running in real-time generating a cylindrical panorama video. All methods introduce some kind of distortion to the captured high resolution

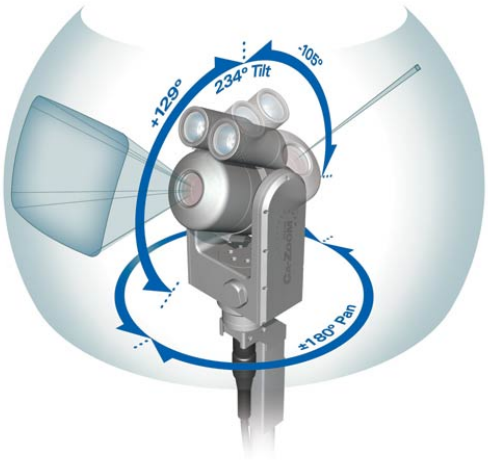


Figure 4.2: A physical PTZ camera and its capabilities [67]. The sphere around the camera shows the range of the *pan* and *tilt* angles. The size of the projection determines the field of view and thus the *zoom*.

image. A physically moving PTZ camera does not introduce any distortion to the image. Hence, a certain amount of processing is required to replicate the behaviour of the actual PTZ camera but using the wide field-of-view frames. This chapter describes our realization of the virtual PTZ camera, and the next section touches upon some previous work in that regards.

4.2 Related Work

The process of using one video and applying geometric transformations on the video to produce another video is a well studied field. The geometrical transformations are applied in a way that the pixel grid is modified, not just the intensity of each pixel.

One of the most common places where this is required is *video retargeting*, trying to adjust the video to displays having other aspect ratios. However, fitting a video to arbitrary sized screen is prone to introduction of distortions. Wolf *et al.* [153] discuss interesting approaches for retargeting. They perform the retargeting in two steps. The first step is to assign saliency to each pixel in the frame, and then a remapping is performed where the importance of the

pixels was taken into consideration. They include face detection and motion analysis into their saliency calculcations.

When wide field-of-view images are recorded, they provide the opportunity to record a large part of the scene into one flat image. However, there are distortions introduced into the image, and when they are displayed on a common rectangular display, the distortions look unrealistic with straight lines curved. This is a common problem that several wide-angle presentations suffer from. Carroll *et al.* [19] investigate approaches for reprojecting the wide-angle images into images that look good with perspective view. The straightforward perspective tranformation introduces skewed angles when the image is wider than 100 degrees in field-of-view. They propose a semi-automatic process where the user can specify the straight lines that need to remain straight in the final output. This input is used along with the image content to create a smooth mapping. They present interesting results which take a wide-angle image as an input and generate a natural looking perspective image.

Zorin *et al.* [161] show an intersting approach to correct perceptual distortions caused by the perspective in the pictures. They propose to decompose the viewing transformation and then optimize the transforms by using the image content. They demonstrate impressive results using their framework. However, the proposed framework is applied only for pictures, and it cannot be trivially extended to videos.

Xinding *et al.* [140] present a similar system as Bagadus, which they use for lectures. However, the Region Of Interest (ROI) is simply a part of a high resolution rectangular video. This kind of ROI requires nothing but a simple cropping operation. The work mainly focuses on the tracking of the target (the speaker in a lecture) using a Kalman filter to steer the ROI automatically. In chapter 3, we saw that using a perspective panorama, like Xinding *et al.* do in [140], introduces uneven sampling distortion to the video.

Grunheit *et al.* [50] talk about a remote streaming system with panoramic view where not all of the panorama information is fetched. They propose an MPEG-4 based system where they fetch only the parts of the panorama

that are used, along with some extra information to support free navigation. Their work does not provide any consideration for the non-linearities that most panoramas provide. Heymann *et al.* [55] extend the previous system and introduce the idea of using a head-mounted dispaly. However, much of the technical aspects are similar to the previous system. They add interaction of 3D objects to the panoramic scene. The work still restricts to intra-frame coding only in support of random access to the frames. However, the loss in compression efficiency is quite high with such a constraint, which will be explored in greater detail in chapter 6.

Several other works exist that try to handle the problem of creating a perspectively correct image from a distorted one as a reprojection problem. Hughes *et al.* [64] discuss about various distortion correction algorithms from fish-eye wide angle lens. However, our case is not that of presenting the whole wide-angle image onto the screen, but rather an extracted view controlled by the user.

Some works [4, 18, 38, 98, 140] exist where video textures are used. However, the resolution is limited, and the experience is still experimental. Research topics have evolved in order to solve problems that stem from an attempt to have a high resolution panorama video as the texture for virtual environments. Different works have focussed on different problems arising from implementing a real-time interactive virtual viewer system. However, no work has focussed on the system aspects of the virtual viewer i.e., real-time performance, processing on different architectures, scaling to several users etc. In this chapter, we therefore discuss how we realized the virtual viewer in *Bagadus*, the challenges we faced in realizing it and then provide some experimental results.

4.3 Theory for Virtual View

When it comes to creating the virtual view, the main goal is to assume that there is a 3D cylinder where the panorama is rolled as texture and then operate a pin-hole camera at the origin of the cylinder as shown in figure 4.3. There are two common ways of realizing this operation. One is to project the

3D cylinder onto a projection plane and display the result. The second one is to trace the ray passing through each pixel on the virtual view to extract the texture from the location of the intersection of the ray with the cylinder. Next, we describe both approaches and evaluate them.

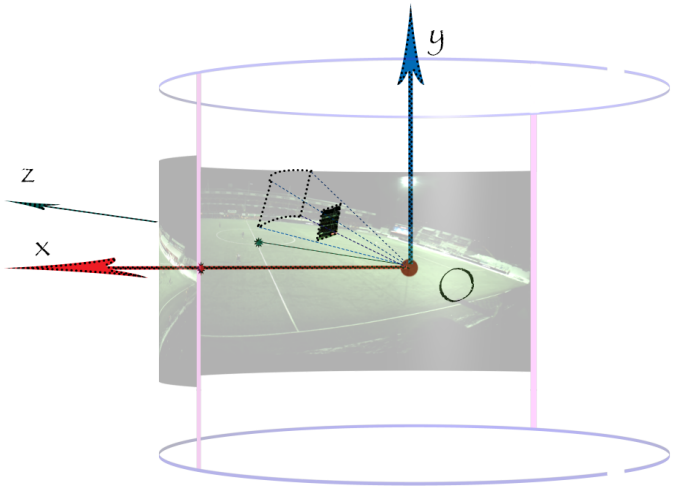


Figure 4.3: The intersection of the ray from the virtual view with the unit cylinder.

4.3.1 Projective Camera

The first approach is to model a cylinder in 3D space and then perform a pin-hole camera projection. The output of the projection is then displayed on the screen. The pin-hole camera is placed at the origin of the cylinder, and then panning and tilting operations are performed as rotation of the camera around different axes. The zoom is defined by the field-of-view, which is decided by the surface area on the cylinder spanned by the image plane (dotted plane) as seen in figure 4.3.

To implement such a projective camera, we can use OpenGL, which is the standard library for high performance graphics applications. OpenGL

provides an efficient way to use the GPU to render scenes by allowing us to define the graphics pipeline. Many GPUs support OpenGL, thus leading to portable graphics implementations.

OpenGL allows us to define objects in 3D space and extract the projections onto different camera models. The OpenGL Utility Library (GLU) is a library built on top of OpenGL and provides several drawing primitives to the user. We used the `gluCylinder` and `gluQuadric` to draw the cylindrical texture in 3D space. Then, a viewport for the virtual view of the size (w, h) is set. A perspective view is then extracted from the viewport which is then presented on the screen. The implementation for virtual viewer using OpenGL is illustrated in figure 4.4

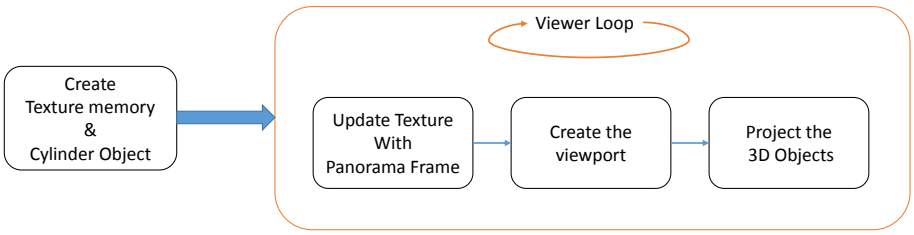


Figure 4.4: The pipeline using opengl when the panorama frame is available.

Internally, `gluCylinder` works by slicing the cylinder into vertical and horizontal stripes and considering each stripe as a rectangle. Figure 4.5 shows an example image of the cylinder object. In the figure, the cylinder is made up of 10 horizontal and 10 vertical slices to demonstrate how it is being constructed. However, in a real world scenario, the cylinder is built using one horizontal slice and about 10000 vertical slices. The reason for keeping only one horizontal slice is that, the image is simply split into vertical lines for each column in the image. In our prototype, we experimented using different number of slices, and table 4.1 shows the time taken for rendering the view by varying the number of slices. We emphasize that, to our limited knowledge, `gluCylinder` only allows for 360^o cylinder even though the football field only occupies approximately 160^o. Due to this high overhead and waste of resources using a projective camera, we evaluate an alternative approach

using ray-tracing.

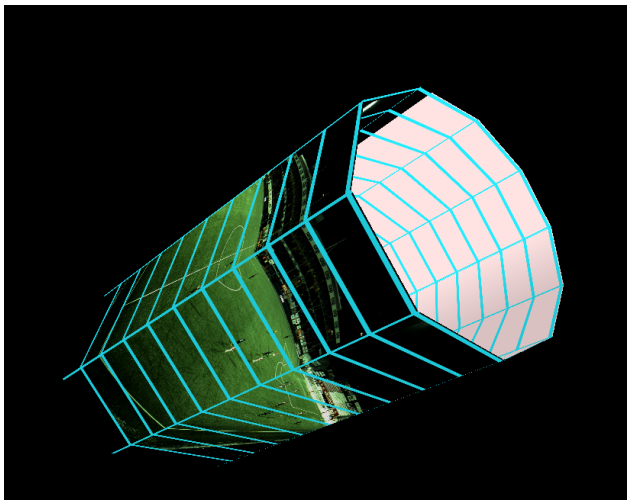


Figure 4.5: Cylinder using the gluCylinder primitive of OpenGL Utility Library. The cylinder is sliced only to 10 slices vertically and 10 slices horizontally to emphasize the way it is constructed.

Type	Horizontal	Time(μs)
1	10	2892
2	50	3026
3	400	3012
4	1000	2968
5	5000	2906
6	10000	2993

Table 4.1: Time taken for rendering a viewport with varying the number of stripes on the cylinder.

4.3.2 Ray Tracing

Ray tracing is a fairly common technique of tracing the path of light through the pixels on the image plane. In our case, the operation is to fetch the pixels of the image formed on the camera from the cylindrical texture. This can be better seen in figure 4.6.

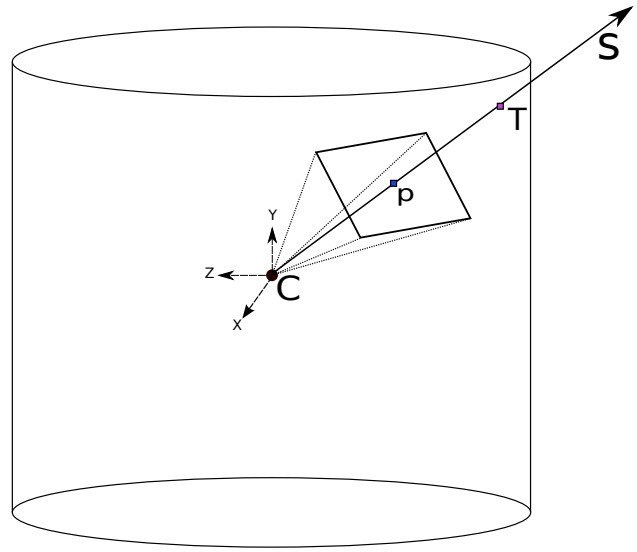


Figure 4.6: The intersection of the ray from the virtual view with the unit cylinder.

A pin-hole camera for a point projection from a 3D point P to image point q can be written in the following manner:

$$\lambda q = [K|0_3] \begin{bmatrix} R & 0 \\ 0_3 & 1 \end{bmatrix} \begin{bmatrix} 0_3^T & -C \\ 0 & 1 \end{bmatrix} P \tag{4.1}$$

where R is the general (3×3) 3D rotation matrix as a function of θ_x, θ_y and θ_z, the rotation angles around the x, y and z axes, respectively. C represents the center of the camera, which is located at the center of the cylinder in our case as shown in figure 4.6. K is the camera *intrinsic* matrix built with focal length(f), scaling factor(s), width of virtual camera(w) and height of virtual camera(h) as in equation 4.2.

$$K = \begin{bmatrix} -f & 0 & w/2 \\ 0 & -sf & h/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

R is the 3D rotation matrix defined as product of the rotation matrices for each axis like in equation 4.3. R_x, R_y and R_z are the three rotation matrices and they are defined in equations 4.4, 4.5 and 4.6.

$$R = R_x R_y R_z \tag{4.3}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & sin(\theta_x) \\ 0 & -sin(\theta_x) & cos(\theta_x) \end{bmatrix} \tag{4.4}$$

$$R_y = \begin{bmatrix} cos(\theta_y) & 0 & sin(\theta_y) \\ 0 & 1 & 0 \\ -sin(\theta_y) & 0 & cos(\theta_y) \end{bmatrix} \tag{4.5}$$

$$R_z = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

$$C = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.7}$$

Let p be the current pixel. Then, we need to find the ray that passes from the camera center C to the pixel p. The ray can be represented by:

$$s = \lambda R^{-1} K^{-1} p \tag{4.8}$$

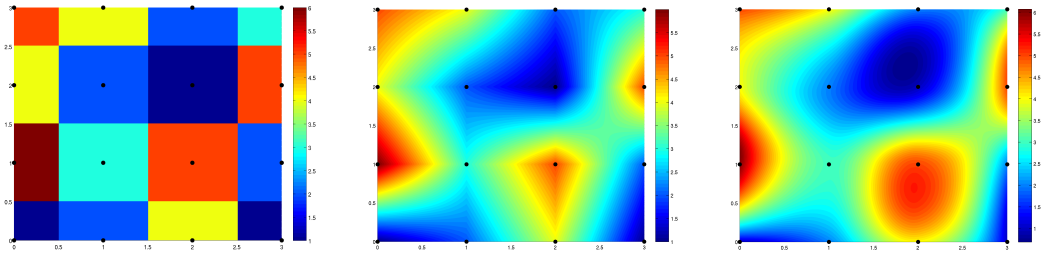
The intersection of this ray with the unit cylinder gives us the exact position on the cylindrical texture. The intersection point can be found as follows:

$$T_x = \left(\frac{W_p}{FOV}\right) \left\{ arctan\left(\frac{-s(1)}{s(3)}\right) \right\} + \frac{W_p}{2} \tag{4.9}$$

$$T_y = \left(\frac{1}{2} - \frac{s(2)}{\sqrt{s(1)^2 + s(3)^2}}\right) H_p \tag{4.10}$$

where W_p, H_p and FOV are the width, height and the field of view of the panoramic texture, respectively. (T_x, T_y) are the coordinates on the unrolled

cylindrical texture.



(a) Nearest neighbour. (b) Bilinear. (c) Bicubic.

Figure 4.7: Examples of outputs using different interpolation algorithms.

When the computation of the ray intersections is performed with floating point accuracy, the intersections on the panorama image cannot be gauranteed to be at integer points, which is the default sampling due to finite pixel size. However, this is a really common problem in inverse mapping and has been solved in image processing. We explored three solutions to this. Figure 4.7 shows the effect of the three interpolations assuming we only have the intensity values at the 16 black points at the center of each square in figure 4.7a.

Nearest Neighbor

The most trivial solution is to round off the floating point values to the nearest integers and extract the pixels from the corresponding locations. However, this method introduces strong aliasing/blocky artifact. Figure 4.7a shows an emphasized effect of using the nearest neighbor.

Bilinear

Bilinear interpolation uses the four nearest neighbours to the point and then uses linear interpolation among the points to estimate the intensity value at the floating point position. This reduces the blocky artifact significantly. However, it assumes a strong linearity, and this can easily be seen in figure 4.7b. It is natively supported on most GPUs.

Bicubic

The bicubic interpolation works by estimating the intensity on a cubic spline that is passing through 16 points around the floating point. This interpolation provides much smoother output compared to bilinear or nearest neighbour as can be seen in figure 4.7c. However, it comes with an added computation cost as it is not supported on GPUs natively.

Performance

Interpolation trades computing time for smoothness and sharpness. Figure 4.8 presents a highly zoomed frame using the three different interpolation modes with the kernel execution time for each of these approaches.

When it comes to visual quality, the nearest neighbour interpolation performs the worst and causes strong aliasing effect leading to the blocky nature of the output image as can be seen in figure 4.8a. A certain amount of blockiness can also be observed in the linear interpolation output. However, the overall image is quite smooth as can be seen in figure 4.8b. The bicubic interpolation outputs the smoothest image output and the least blocky output as seen in figure 4.8c. Thus, as it can be seen in the figure, bicubic interpolation seems to provide the best visual quality at the cost of a higher execution time. However, we pay less than 3.5ms, which is far below the real-time threshold and therefore, choose the higher image quality. Next, we investigate using the ray-tracing approach in the actual system implementation.

4.4 Implementation

In order to implement all the theory presented above, we built a system in several steps. The basic building blocks remain the same from a theoretical point of view. However, there are big differences in the implementations. Figure 4.9 shows the basic steps in the virtual viewer system. The first task is to fetch the video files from a network. Then, the files must be decoded to extract individual panorama frames, which are then used for rendering the virtual views. The user controls are captured from an input device that

(a) Nearest neighbour (2916 us).

(b) Bilinear (2840 us).

(c) Bicubic (3242 us).

Figure 4.8: Frame quality and execution time for the interpolation algorithms.

determines the virtual view parameters : pan, tilt and zoom.

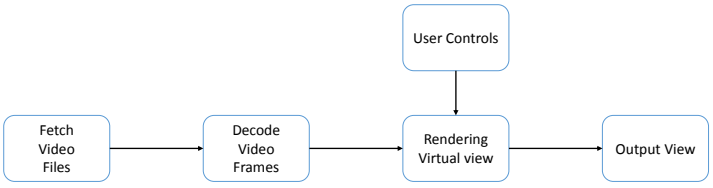


Figure 4.9: The basic building blocks of the virtual viewer system.

In this section, we will briefly go through the chronological development of the system from the basic serial implementation to the parallel pipelined implementation. We use `libcurl` for fetching the video files from the server. For decoding the video, we use `libav`. We use `openGL` for the rendering of the image onto the screen. For the GPU processing itself, we use `CUDA`.

4.4.1 Video Handling

For streaming, we use HTTP segment streaming (with plans for adaptive HTTP streaming, see chapter 6). The segments of the panoramic videos are served by an Apache server along with a manifest file. The manifest file is used to inform the clients when the next file is ready for download. The viewer checks the manifest file periodically and downloads the next segment when it is ready.

As soon as the panoramic video segment is transferred, it is kept ready for processing. This process runs in the background without blocking either the display thread or the user input thread.

4.4.2 Version CPU: Serial Implementation

As the name says, this implementation is a completely serial one. Every step of the figure 4.9 happens in a single thread. The decoding, may use multiple threads, but this is managed internally by `libav`. A straightforward implementation of the renderer for CPUs is to loop through all the pixels in the virtual view and find the positions where the rays land on the panoramic texture. The heavy operations include an inverse tangent and a square root in every pixel calculation. Since the operations are well suited for parallelization, we could have used the CPU's vector instructions like MMX, SSE and AVX, but the modern GPUs have even greater potential, so next, we have ported the program to a GPU.

4.4.3 Version GPU1: Parallel Real-Time Implementation

In figure 4.9, all blocks can function in a pipelined manner and in this section, we present a first version of a parallel multi-threaded implementation achieving real-time performance on the GPU. In this implementation, we use one thread to perform the fetching operations because network operations can have latency and it can affect the playback of the virtual view if the files are not available in time. The user controls also are run in a seperate thread.

The main thread still performs the decoding, rendering of the virtual view and displaying on the screen.

A simple port of the rendering module (version GPU1) using CUDA performs the calculation of the ray intersection and *fetching* of the corresponding pixel from the panorama on the GPU. So, the videos are decoded on the CPU, the frames are transferred to the GPU, and calculations and fetching operations are performed on the GPU. Since it is possible to render OpenGL textures written by an NVidia CUDA kernel directly from the GPU to the screen using `CUDA/OpenGL interoperability`, this implementation uses that feature. An OpenGL texture is defined in advance and bound to the screen buffer. When the ray calculation and pixel fetching operations are complete, the output is not transferred to the host, but written to the bound OpenGL texture buffer on the GPU. Then, this texture is displayed directly on the screen, saving the transfer overhead from device to the host. In addition, most CUDA enabled devices support hardware accelerated linear interpolaton from CUDA *textures*. We use these *textures* instead of global memory on the GPU to further increase the performance.

4.4.4 Version GPU2: Parallel Pipelined Implementation

On top of the improvements in version GPU1 implementation, we made a completely pipelined implementation (version GPU2). Figure 4.10 presents all the stages of the pipeline. Each of these modules is designed using a producer/consumer pattern. They each run in their own thread and communicate with each other using data queues. The key idea is that all the modules perform at their peak performance when there is a task and push the output to the queue connecting the next module.

One can observe that even the GPU part of the pipeline is split into multiple threads. The reason for this is that the several operations on GPU can be performed concurrently, and `CUDA` allows for that using `streams`. A `Stream` is defined as a sequence of operations that execute in issue-order on the GPU. Hence, assigning operations to different `streams` can enable

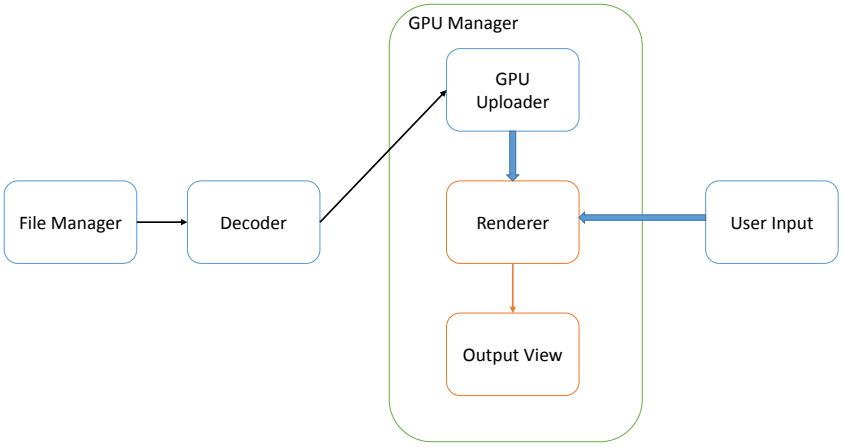


Figure 4.10: The fully pipelined parallel system. Each module runs in it's own thread with producer/consumer pattern.

concurrency in operations. In our case, the next decoded frame is transfered to GPU while the GPU extracts the virtual view from the previous panorama frame.

4.4.5 Performance

The two reasons that we preferred the ray-tracing approach to the `openGL` are performance and control of the pipeline. The main tasks, once the panorama frame is decoded, are transfer of the frame to GPU and rendering of the virtual view based on the input pan-tilt-zoom parameters.

From table 4.1 presented earlier, we can see that the rendering time for `openGL` is about $3ms$, and from figure 4.8, the ray-tracing approach also takes around $3ms$ irrespective of the interpolation performed. However, the transfer of panoramic image texture varies significantly between the `openGL` and `CUDA` approaches because of the size of the texture. As previously mentioned, the panorama frames are encoded using H.264 using YUV 422 planar packing. When working with CUDA, we just need to transfer the YUV 422 texture to the GPU and internally work with YUV 422. However, the `opengGL` requires RGB textures. This has two disadvantages, i.e, one is the overhead of conversion from YUV to RGB and the other is that RGB data

requires 1.5 times larger memory than YUV 422 data.

In our experiments, we found that the time taken for transferring a texture to GPU using `openGL` is approximately $26ms$, whereas, the time taken for transferring YUV 422 data to GPU using `CUDA` is approximately $3.6ms$. Moreover, the `CUDA` transfers can execute concurrently with computation on the device, thus providing more window for the real-time execution. Due to this, we based the implementation on ray-tracing.

Property	Desktop	Laptop
CPU	i7-2600	i7-2620M
CPU Cores	8	4
CPU Clock	1600 MHz	800 MHz
CPU Memory	8G	4G
GPU	GeForce GTX 460	NVS 42000M
GPU Cores	336	48
GPU Memory	1G	1G
GPU Clock	1300 MHz	1480 MHz
CUDA version	5.0	5.5

Table 4.2: Configuration for Desktop and Laptop hardware used.

Furthermore, we experimented with two machines using the virtual viewer. One machine was a desktop and the other was a laptop. Table 4.2 presents the configuration of the machines. Figure 4.11 shows the performance of each module on the two devices. It can be seen that even on a small laptop, the performance is better than real-time. The most time consuming task is decoding. We can also see that the performance of the decoding module can be improved by using multiple threads. However, even then the decoding module becomes the bottleneck for the pipeline. Assigning more threads to a single module will decrease the performance of the pipeline because, when the threads exceed the number of cores, there is a lot of context switching overhead.

Most of the performance measures provided in this section are for a Full HD resolution, but the resolution of the virtual camera varies with the viewing device. Figure 4.12 therefore demonstrates the effect of the size on the kernel execution time of the final generation of the zoomed image (note

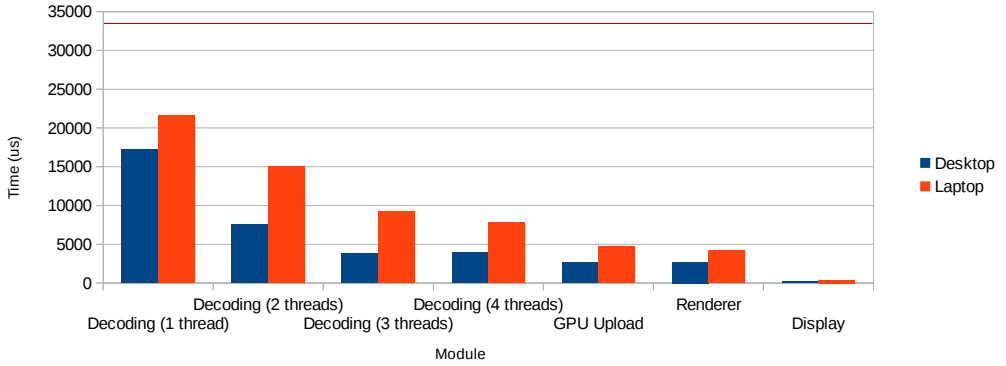


Figure 4.11: Performance of each module in the pipeline on a desktop and a laptop. The red line on the top shows the deadline for real-time performance at 30 fps.

the *micro*second scale). It can be seen that the time taken by the kernel drops significantly as the size of the virtual view decreases. This suggests that there is potential to perform in real-time using even less powerful GPUs that are common in the recent mobile devices.

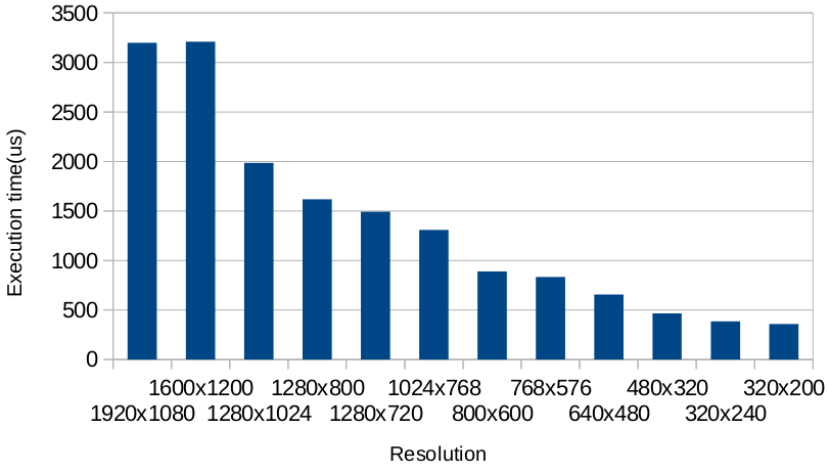


Figure 4.12: Execution times for various sizes of virtual camera.

4.5 Summary

In this chapter, we have presented a system for real-time interactive zoom and panning of panorama viewo used in a soccer stadium scenario. Based on

video streams from five stationary 2K cameras, processed and stitched into high resolution panorama video, we are able to support any free view angle from the position of the camera array, i.e., an interactive virtual camera able to present an extracted, perspective corrected personalized view to the users.

We have presented two possibilties in terms of realizing the virtual camera given the pan-tilt-zoom parameters. One way is by modelling a 3D cylinder in space and projecting it onto a virtual camera. The first approach is realized using the off-the-shelf graphics library `OpenGL`. The second way is to perform ray tracing from the virtual camera and find intersection on the panorama texture rolled onto a 3D cylinder. This approach is realized by programming the GPU using `CUDA`. By experiments, we concluded that the `CUDA` approach provides us with better performance and even more opportunities for tuning.

We performed experiments on a commodity desktop hardware and a laptop hardware. The results show that real-time performance is only constrained by the video decoding time, but on the tested hardware, real-time performance was achieved for all configureations and tested resolutions (up to 1080p HD).

In this chapter, the operation of the virtual camera is restricted to manual control. However, during an actual 90 minute football match that can be tiring. Hence, the next chapter (chapter 5) covers different approaches on how the virtual camera can be operated automatically to follow interesting *features* like a ball or a specific player. Moreover, at the moment, the system demands a large network bandwidth due to the full resolution panorama video. Even though the panoramic video is compressed using H.264, saving quite a lot of space, the network overhead is still high. Hence, we explore a few approaches of saving the bandwidth in chapter 6.

Chapter 5

Visual Servoing

In the previous chapter, we have described how a virtual camera can be generated in real-time for every individual client. The previous chapter focused on the system aspects of the virtual camera, and the operation of virtual camera is limited to manual control. In a real-world scenario, like that of a football game, even though manual camera control is a desireable and exciting feature for a short period, it is not practical during an actual 90 minute game. Normally, a user expects to follow the game rather than use her attention to steer the virtual camera during the entire game.

In that regard, it can be beneficial to additionally provide a higher level abstraction for the user interaction instead of a manual operation. Figure 5.1 illustrates two scenarios. In the manual operation scenario, the user can steer the virtual camera. In the guided scenario, a user merely requests the client device to operate the virtual camera automatically depending on the user's interests. For example, a user can request the device to automatically steer the virtual camera to follow the ball, a particular player or even a group of players. Such a request can be quite useful for coaches to observe the performance of players. The task of the device then becomes to provide a smooth operation that is similar to that of being provided by a human camera operator with several years of experience.

In lose terms, such automatic camera operations can be considered as a problem of *visual servoing*. Visual servoing is a technique using feedback

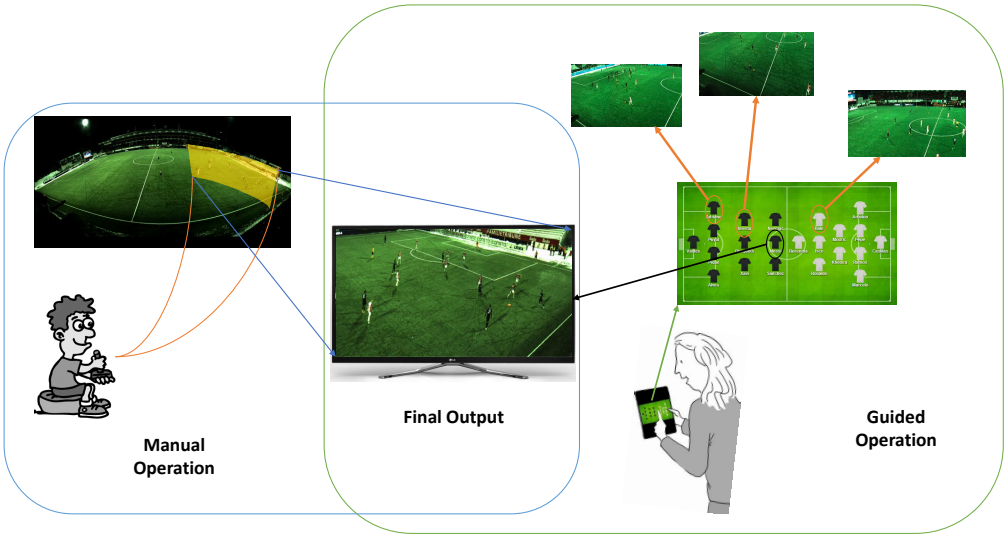


Figure 5.1: Two modes of new interaction that can be provided to user can be seen here. The manual mode provides user the option to steer the camera manually and the guided operation provides a set of features to chose from, for example a ball or a player. Once a feature is selected, the system automatically creates the smooth motion following the feature.

information from multiple sensors to control the physical motion of a robot with a possibility of the camera on the robot [93]. However, in our case, it is not a real physical camera that is being operated. This problem is also different from the *control theory* of the camera based tracking, because often such systems feed the final output to a machine. In such a scenario, the most efficient form of tracking can be the one where the target is at the center of the image frame at all times. However, in our system, the final viewer is a human and placing the feature, for example the ball, in the center of the image is not the most ideal form of presenting football videos.

This chapter presents a few approaches to achieve this along with a subjective study to compare the machine to a human operator. This chapter can be considered as an extension of [40] (Appendix D). In addition to the football scenario, we experimented with these approaches for surveillance applications. In this regards, we presented a demo at ACM Multimedia [43] (Appendix J).

5.1 Related Work

Some previous works have focussed on virtual servoing in sports scenarios, and here we provide some examples.

In [4], Ariki *et al.* present a simple system that is capable of automatically providing ROI from a high resolution overview video. They use a Schmitt window based steering. They perform a small-scale user study based on 15 users.

In [18], Carr *et al.* present a hybrid system using both a robotic PTZ camera and a virtual camera generated from a panorama. They evaluate their system comparing it to a human-operated one as benchmark. Their motivation is to get as close to the human operator as possible. Even though this work was a really thorough work dealing with automatic virtual cameras, they fix the focal length and the tilt angle limiting the movement to only panning of the virtual camera.

Daigo *et al.* [27] present a system for automatic panning based on audience face direction. Their main hypothesis is that the direction of the scorers in the game has a great impact on understanding where the action happens on the court. The face detection is performed by template matching. As an idea it is interesting, however the complexity of their approach makes it hard for real-time operations.

Dearden *et al.* [29] present a system that learns from the movement of a trained camera operator. They propose a workflow for learning the camera movement from the videos recorded during a professional football game. In their approach, they initially extract the pitch region information using histogram based pitch identification. Then, the player positions are estimated using simple morphological operations and particle filters. Further, they use the Kanade-Lucas-Tomasi (KLT) optical flow to estimate the camera movement from the video. Then, a model is learnt roughly based on k-Nearest Neighbour (kNN). The authors demonstrate the camera movement model only in simulations. Moreover, the player position estimation is also reported to be inaccurate, estimating only some players on the field not all of them.

It can be seen that the virtual camera control in most cases is handled as a

reduced problem limiting it to only panning motion. However, it is common to see pan, tilt and zoom operations in a football broadcast. In the next section, we provide details of our approaches on automatically controlling the virtual camera.

5.2 Approaches for Automatic Camera Control

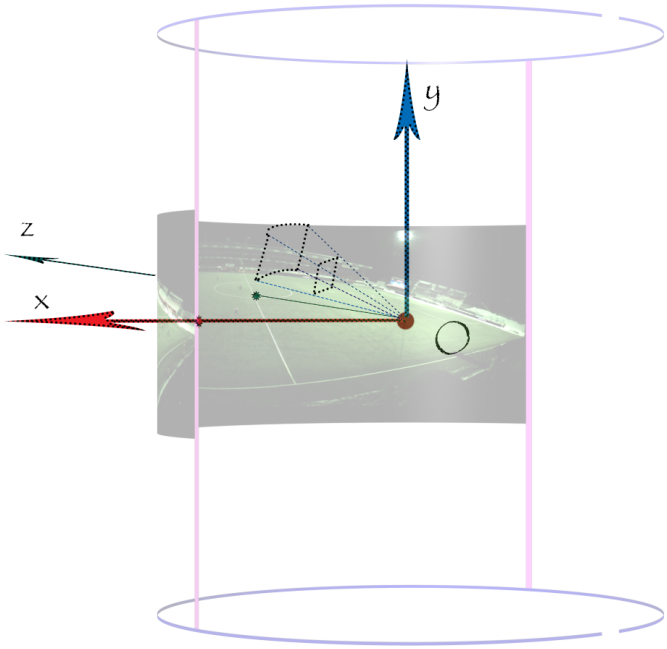


Figure 5.2: The virtual camera operation mainly involves in deciding the angles around y and x axes. The field of view is then determined by the focal length, which decides how zoom of the output view. The interesection of x and z axes with the panorama texture are marked with stars. The projection details from the panorama are presented in figure 4.3.

To operate the virtual camera, we are operating in 3D space with the origin on the axis of the cylinder for all the movements. Here, we let θ_x be the angle along the pan direction (horizontal), θ_y be the angle in the tilt direction

(vertical), and f be the focal length, i.e., these three variables are used and changed to control the virtual camera. A ray pointing at $(\theta_x, \theta_y) = (0, 0)$ meets the panorama image at the center. Figure 5.2 presents a cylindrical texture rolled on a 3D cylinder and the corresponding pin-hole camera. It can be easily observed that any rotation around the y-axis contributes to panning and the x-axis contributes to tilting. The rotation around the z-axis corresponds to action of rolling. However, we do not use this action in a football scenario.

Furthermore, let the feature point on the panorama be $s_p = (\theta_x^p, \theta_y^p)$, and let the current state at of the camera be $c^i = (\theta_x^i, \theta_y^i, f^i)$ where previous states are denoted $c^{i-1}, c^{i-2}, ...$ Here it must be noted that translation is also possible, however, we fix the camera viewpoint at one location and only allow for pan, tilt and zoom operations. Then, the problem of operating the virtual camera can be formulated as:

$$c^i = F(s_p^{i+l}, s_p^{i+l-1}, s_p^{i+l-2}, ..., c^{i-1}, c^{i-2}, ...), \tag{5.1}$$

where l is the future data fetched by simply delaying l units of time. The models that we developed for controlling the virtual camera handle the state variables independently. There are two models for controlling the angles, and the focal length is controlled depending on the current position of the center of the virtual camera on the panorama, where we also investigate two models.

5.2.1 Models for Pan and Tilt

Figure 5.3 shows the position of the ball across 500 frames in the panorama panning space (θ_x). Assigning θ_x to the the ball position would lead the virtual camera to be centered on the ball in each frame. We can observe that the signal varies in a non-smooth fashion. It is not a pleasant experience to watch such a video where the camera is centered on the ball every frame. The main constraint that we have while selecting the models for smoothing the input signal, is that the computational complexity should be really low. This ensures that there is no significant additional overhead on the client

side. Even though Extended Kalman Filters and other non-linear regression methods provide superior smoothing abilities, they come with a big cost of high computational complexity. Hence, we have used two very simple models for the pan/tilt operations, i.e., a Schmitt trigger and an Adaptive trigger. The pan and tilt angle movements are computed independently in our calculations. However, the changes in tilt angles are penalized more than the pan angles because panning is usually more natural than tilting a camera in wide field of view situations. The analysis in this chapter is performed on pan angles even though the tilt angle is also modified for the camera operation. An example for changes in tilt angle can be seen in figure 5.6.

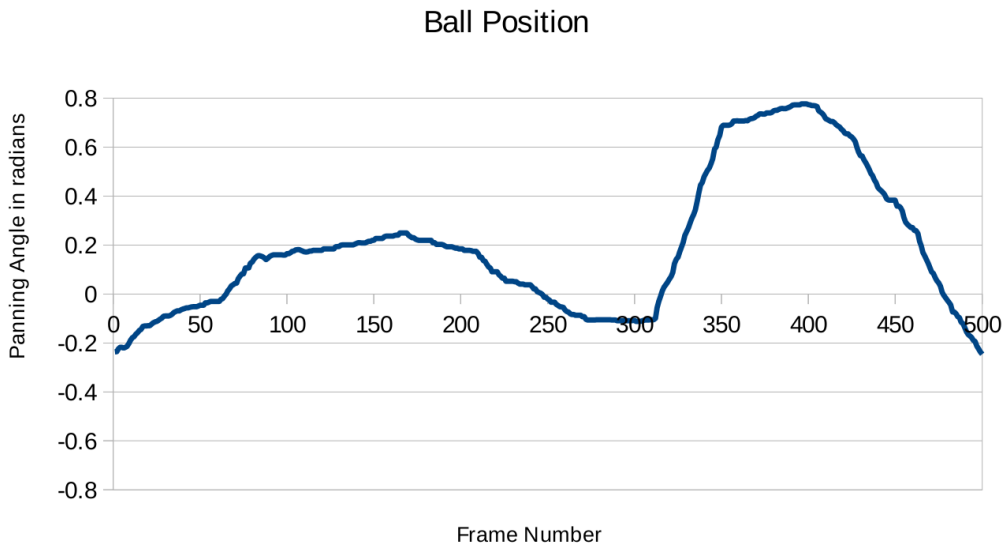


Figure 5.3: Ball position plotted against frame number for one of the 500-frame segment from a match. It can be observed that using exactly the data, one will get a noisy output.

Schmitt Trigger

Schmitt Trigger [130] is a traditional signal stabilizing filter that has been used in Electrical Engineering and other fields for several years. It offers a simple and reliable way to stabilize an input signal thus allowing us not to spend more computational power in operating the virtual camera. We

modified Schmitt Trigger so as to provide a smoother movement by adding an acceleration α to it. Algorithm 1 presents this approach. For the Schmitt trigger to function, we define an imaginary window (characterized by θ^t) inside the virtual view. When the target point is inside the window, the virtual view is quickly brought, yet smoothly to avoid abrupt movements, to a stop by using a deceleration α_{stop}. Once the target point goes outside the window, we provide an acceleration α, to the virtual view so that we reach the target. The sign of α depends on the current velocity of the feature point and the virtual camera. The acceleration is added only when the velocity is less than the maximum velocity $\delta\theta_{max}$.The velocity and acceleration of a variable θ are written as $\delta\theta$ and $\delta^2\theta$, respectively.

Algorithm 1 Schmitt Trigger

1: $(\theta_x^0, \theta_y^0) \leftarrow (\theta_x^p, \theta_y^p)$
2: **while** running **do**
3: **if** θ^p is outside θ^t **then**
4: **if** $\delta\theta^p > \delta\theta^{i-1}$ **then**
5: $\delta^2\theta \leftarrow \alpha$ //Accelerate in the positive direction
6: **else**
7: $\delta^2\theta \leftarrow -\alpha$ //Accelerate in the negative direction
8: **end if**
9: **else**
10: $\delta^2\theta \leftarrow \alpha_{stop}$ //Bring the movement to stop but smoothly
11: **end if**
12: **end while**

Adaptive Trigger

The adaptive trigger is designed to adaptively estimate the required velocity of the virtual camera. We compute the movement of the camera in a two step smoothing process. We use a running weighted mean smoothing at both steps. Another key difference, compared to the Schmitt trigger, in this model is the use of future data. By delaying the system by 1 second, we have "future data" for about 1 second. The windows for the regression are smaller than the fetched future data because of the second level smoothing.

For a given variable x, let $S(x)$ be the smoothed value. Algorithm 2 describes this approach. When computing the target velocities, the gradient is taken over smoothed positions because the noise gets amplified with a gradient. τ is a threshold for removing small variations in position that are caused by small jerky motions. These jerky motions create a small average velocity over multiple frames. This is similar to Schmitt Trigger, except that the τ is applied to the speed instead of position. We preferred to keep the camera static rather than subjecting it to a really slow movement.

Algorithm 2 Adaptive Trigger

$(\theta_x^0, \theta_y^0) \leftarrow (\theta_x^p, \theta_y^p)$
while running **do**
    $\delta\theta^s = \delta(S(\theta))$ //Initial velocity calculation
    if $\delta\theta^s > \tau$ **then**
        $\delta\theta^{st} = \delta\theta^s$ //Thresholding
    else
        $\delta\theta^{st} = 0$
    end if
    $\delta\theta = S(\delta\theta^{st})$ // Final velocity estimation
end while

5.2.2 Models for Zoom

The zoom is controlled by modifying the focal length (f), the virtual view is zoomed in by increasing f. In the current system, we developed two models to change f, smooth zoom and toggle zoom.

Smooth Zoom

The smooth zoom imitates the nature of the physical zoom that is obtained by smoothly controlling the zoom ring on the recording camera. We modelled a quadratic function in the current camera position coordinates such that f increases when the position approaches the goal posts or the other end of the field from the camera setup:

$$f^i = \lambda_0 + \lambda_1(\theta_x^i - \theta_{x0})^2 + \lambda_2(\theta_y^i - \theta_{y0})^2 \tag{5.2}$$

where λ_1 and λ_2 are the parameters that control the effect of pan and tilt angles, respectively. θ_{y0} is used to offset the curve so that the function increases over all the tilt angles. θ_{x0} is set to 0, because the function should be increasing from the center of the field as we move towards the goals. Because, it is usually preferred to have a zoomed in view closer to the goals and an overview close to the midfield line. λ_0 is the zero order offset. The function for focal length is smooth, hence, there are no abrupt shifts between zoom levels. All the parameters are empirically selected. However, selecting these parameters varies for different installations and experiments needed to be performed to obtain visually pleasing results.

Toggle Zoom

The toggle zoom mode was developed to imitate the immediate switch in zoom levels, similar to directors cutting over between different cameras in the stadium. We picked a rather simple model for creating this effect. The panorama is partitioned into several zones and a focal length is assigned per zone. The zones can be seen in figure 5.4. They are selected based on two major factors, proximity to goal and proximity to the camera. It can be assumed that close to the goal posts, the virtual view is preferred to be zoomed in. However, in the middle of the field, an overview is better because one can observe the game better. Since, it is not possible to have the same size along the entire center line, because the camera is located at one end of the center line. The zoom factor is selected based on how far the view is from the camera position. If the view is too close to the camera position, then the view should be zoomed out and vice verca. The view simply changes from one zoom level to the other one without smooth transition.

5.3 Objective Analysis

In the models that we presented in the previous section, there are several parameters influencing the behaviour of the automatic pan, tilt and zoom operations. In this section, we present a brief analysis on how those pa-

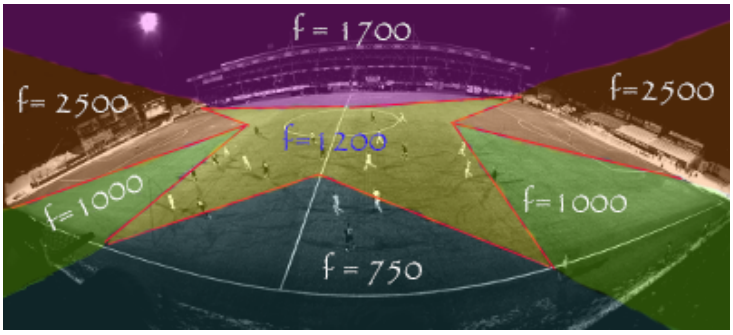


Figure 5.4: Toggle zoom assignment.

rameters affect the results. In order to compare with a human operator, we requested two individuals to perform the camera steering using a joystick. One individual is an expert camera-man with several years of recording experience who got used to operating the virtual camera using a joystick by doing several training attempts. In the experimental results presented below, the results from that individual are labelled "expert". The second individual, termed "novice", has a lot of experience to use the joystick from gaming. However, he lacks experience with broadcast video recording.

First, we analyse the effect of two models used for panning and tilting on the angles of the virtual camera. Then, we analyse the zoom models comparing them to the human operators.

5.3.1 Execution Time

The average execution times per frame for the Schmitt trigger and adaptive trigger to find the virtual camera positions are around $2\mu s$ and $30\mu s$, respectively. Even though they differ significantly, the absolute values are still negligible, compared to other parts of the pipeline and far below the 20 ms (50 fps) real-time threshold.

5.3.2 Pan/Tilt Models

Figure 5.5 illustrates the panning angles for a 300 frames segment for camera movements that try to follow the ball and ball position, where we see the pan angle (in radians) of the virtual view generated by both machine and human

operations. In other words, if the curves are close, they capture more or less the same view. The *lagging* nature of the Schmitt trigger and the human operators can be observed in the figure owing to the fact that, they recieve the ball position as it happens and the next camera position is computed based on the previous ball position. On the other hand, the adaptive model has access to "future data". Figure 5.6 shows a similar image but for the tilt angle (in radians). We can see that the variation in panning angle is much higher due to the nature of the football game. So in the following, we perform objective analysis only on the panning angle, but the same observations are valid for tilt angle as well.

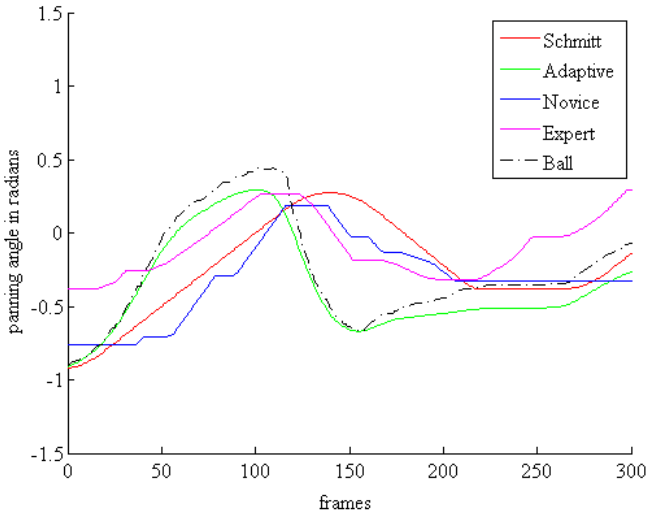


Figure 5.5: Schmitt trigger and adaptive trigger plots for 300 frame segment along with the plots from human operated camera for the panning angle.

Schmitt Trigger - Analysis

There are three control parameters in the Schmitt trigger case, the acceleration, maximum velocity and the deceleration. Figure 5.7 displays the angles for different accelerations over 300 frames. It can be observed that higher acceleration tends to get the camera center closer to ball position quickly, but a problem is that it can introduce uneasiness in watching because the camera speeds rather quickly and unnaturally.

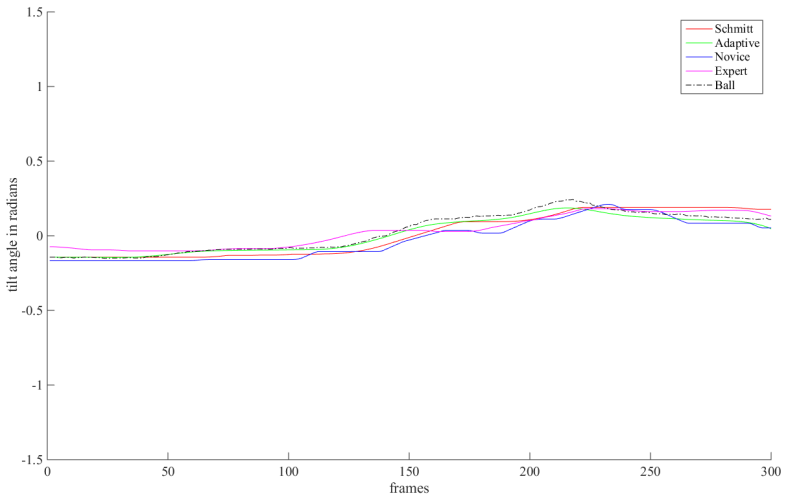


Figure 5.6: Schmitt trigger and adaptive trigger plots for 300 frame segment along with the plots from human operated camera for the tilt angle.

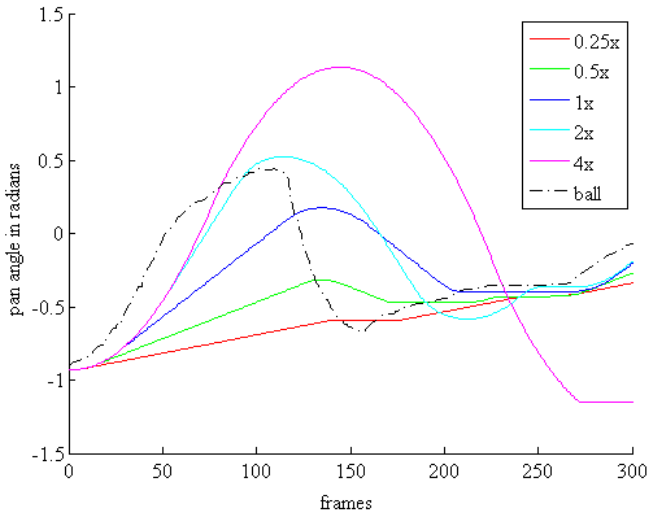


Figure 5.7: The calculated trajectories for various acceleration values in the Schmitt trigger case. Here, x is 0.0001 $\frac{rad}{s^2}$.

Figure 5.8 demonstrates the effect of varying the maximum velocity over 300 frames. When the ball moves really quickly, the curves in the plot show that the higher the maximum velocity, the closer they get to the slope required. However, this creates an undesired effect of overshooting irrespective of the quick deceleration. For example, consider the case of $0.25x$ and $4x$

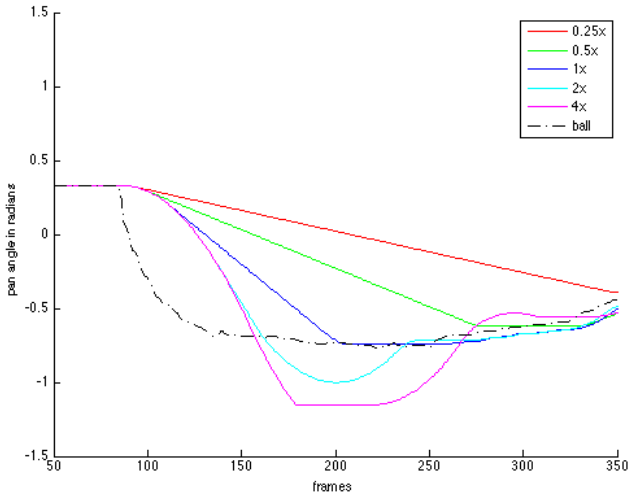


Figure 5.8: The trajectories for various max velocities in the Schmitt trigger case. Here, x is 0.01 $\frac{rad}{s}$.

which are two extremes. Between 0 and 150 frames, one can observe that the $0.25x$ is far from the original ball position compared to the $4x$. However, once the ball changes the direction, the $4x$ trajectory shoots off, and it takes time before the virtual camera position comes close to the ball position. This happens because of the constant acceleration. Considering this tradeoff, we can observe that $2x$ trajectory performs the best in this scenario.

Moreover, figure 5.9 demonstrates the effect of the deceleration on the virtual camera movement. The trade-off here is between an appearance of a mechanical stop to a swinging effect. Both the velocity and acceleration effects can be seen in the plots.

Adaptive Trigger - Analysis

In the adaptive trigger, we have two control parameters. One is the window size and the other is thresholding for clipping. The thresholding for clipping only eliminates small jerky movements, it is empirically chosen and it's plots do not provide great variation. Figure 5.10 demonstrates the effect of the window size on the panning variable. The window size is varied between 5, 10, 15 and 20 frames. A scene of 300 frames, where there are atleast a few changes in the ball direction, is picked. The exact field of view depends on

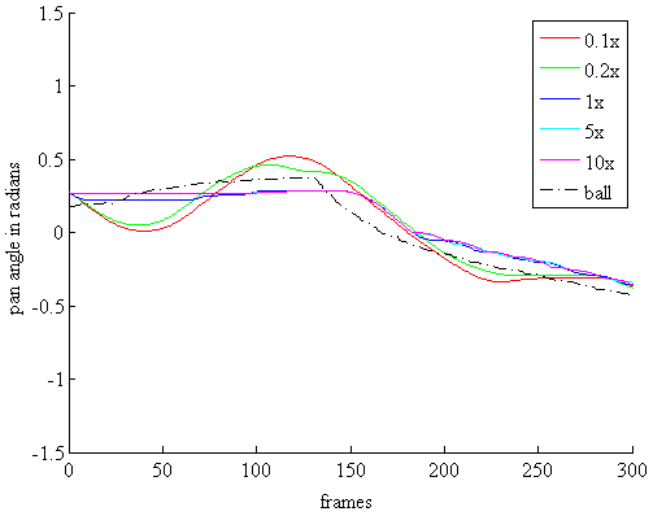


Figure 5.9: Effect of varying stop-acceleration on the trajectories in Schmitt trigger case. Here, x is 0.001 $\frac{rad}{s^2}$.

the current focal length. However, as a rule of thumb, anything inside 0.2-0.5 radians from the center of the virtual camera can be assumed to be inside the field of view.

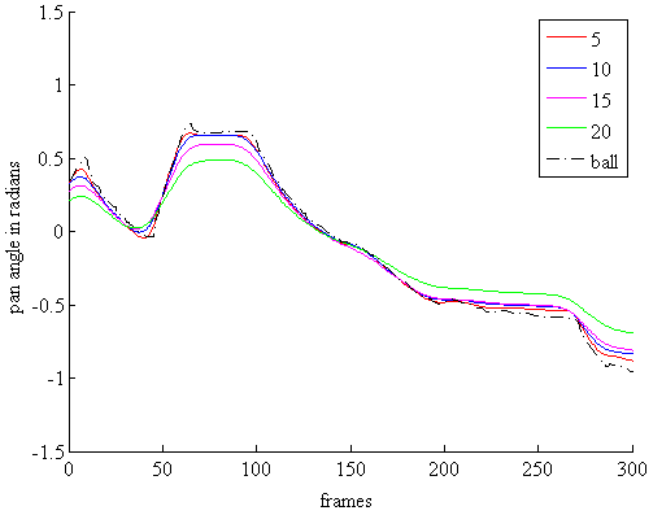


Figure 5.10: Effect of window size selected for smoothing on the trajectories using the adaptive trigger. The window size is in number of frames.

5.3.3 Zoom Models

Since the calculation of zoom is a closed form expression over the current viewing position, the execution time is really low. Figure 5.11 provides plots from the different zoom models and the human operators over a 300 frames segment. Since the position is dependent on the pan/tilt model chosen, both curves are calculated using the adaptive trigger. It can be observed that the machine-generated zoom curves show noticeable similarity to the expert controlled camera, irrespective of the simplicity in the models.

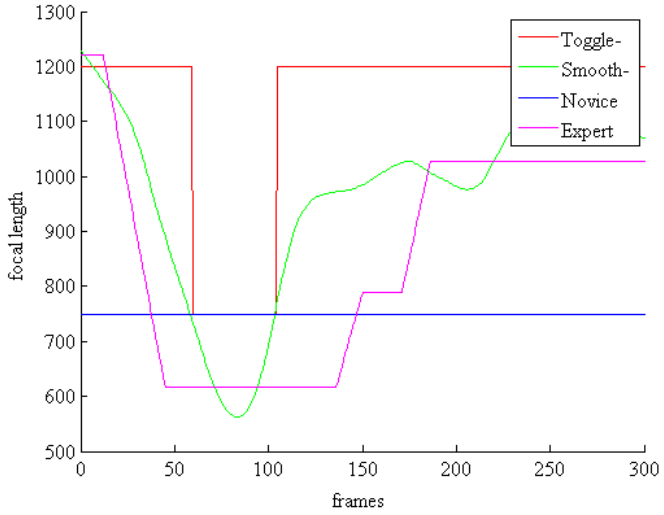


Figure 5.11: Smooth zoom and toggle zoom plots along with the plots from human operated camera for 300 frame segment.

5.3.4 Conclusion

The objective analysis shed some light upon how different parameters affect the trajectory of the virtual camera. This provides us a basis for tuning the parameters for best visual appearance within each approach. However, the most important factor in deciding the quality of an appraoch is the viewer. The next section, therefore describes an experiment and results taking the user into consideration.

5.4 Subjective Analysis

In the development of a user-centered system, subjective feedback from a representative group is essential for a successful outcome. However, users are not always able to express what they want and need. Fortunately, useful experimental approaches have been adapted and extended by researchers in the field of multimedia; a comprehensive overview of these methods is provided by the ITU [69, 70]. For instance, Quality of Experience (QoE) studies aim at assessing cognitive, emotional and/or behavioural responses to different aspects of multimedia systems [154]. Particular attention has been devoted to the perception of video quality and the detection of visual artifacts [36, 49, 105]. QoE research is also bringing behavioural experiments out of the laboratory [28, 105]. While controlled experimental settings are necessary to isolate factors of interest from the influence of external variables, this type of control becomes less pertinent when considering the great variations inherent in multimedia content. Furthermore, by presenting multimedia sequences on the platforms where they are normally enjoyed, the results can be generalised to real-life scenarios with less restrictions.

An additional benefit of removing laboratory settings is access to a larger pool of participants. Running laboratory experiments is time-consuming for researchers and participants alike, hence bringing the experiment to the participant could increase the likelihood that an individual would consent to participate. Using mobile devices, multimedia researchers can collect results that represent the platform the content was designed for [28,105], while at the same time allowing recruitment of potential volunteers in central locations. Moreover, the use of online surveys and tests is also becoming more common [22]. The use of crowdsourcing and other online recruitment schemes offers even more flexibility than mobile test devices, but this flexibility comes at a cost. With participants left unobserved, the experimenter has no guarantee that the task is completed diligently. Consequently, online testing may add concerns to the validity of results [132]. However, larger numbers of participants and stimulus repetitions, combined with the exclusion of outlying scorers, may alleviate some of these concerns.

Since the user experience with the system is dependent on many factors outside video quality, the task of evaluating the system could quickly become a daunting one. In the current context, camera zoom, pan and tilt movements must be evaluated alongside the action of the game itself, the panaroma view, and the visual quality artifacts that occur sporadically. To ease the burden of separating the factors of interest from these and other distractions, we decided to perform a pairwise comparison test [70] to contrast the different combinations of camera movements, two by two. When asked to select one of two versions of the same sequence, participants are presented with a task that is comparatively simpler than subjective ratings of sequences. Seeing how pairwise comparisons only require decisions on one's preference, this test is a good alternative when exposing participants to unfamiliar stimuli and situations [86].

5.4.1 Pairwise Comparison Method

User preference for transient variables, such as camera movement and zoom, is deemed to be highly subjective and to depend on the presented sequence. To avoid subjective ratings that may vary more between presentations than between our experimental variables, we decided to use pairwise comparisons, as recommended by the ITU [70]. Hence, each sequence was presented twice in a row, with only our variables of interest changing between presentations. In the first study, we evaluate the automatic approaches against each other to figure out the preference of the viewer among the automatic approaches. Then we evaluate the best of the automatic approaches against the human operated virtual camera.

5.4.2 Evaluation Metric

Each user is shown the same pair in different orders 4 times. The user decides which video provides a more pleasing experience. In order to figure out whether the users' preferences are consistent with their own choices across experiments and whether some approaches are consistenly performing better than others in several users' point of view, we performed several standard

tests in statistics. Due to the uncontrolled nature in our data collection, we constrained ourselves to analyse the data only using non-parametric tests. The three notable ones are as following:

Parametric Statistics This test simply reports mean and standard deviation regarding each option. Further inferences can be made from the assumption of a normal distribution. However, we restricted ourselves to simply reporting the metrics and did not infer using these statistics.

Friedman Rank Test This is a commonly used non-parametric statistical test to detect inconsistencies among repeated experiments and corresponding choices. The test computes ranks for each stimulus and depending on how well spaced the ranks are, one can study the consistency in the choices. Here, the significance is tested by comparing to the $(\chi^2(k))$ distribution for k degrees of freedom. In our case, high values indicate high level of signifance.

Wilcoxon Signed-Rank Test This is another commonly used non-parametric statistical test to asses whether the mean ranks of the stimuli differ. In addition this test also allows for computation of the effect size which is essentially the rank correlation.

5.4.3 Study 1: Camera Controls

Because the system aims to provide users with the best possible experience, we need user feedback in order to establish the most preferable parameters for camera movements and zooming. We therefore conducted a user study to compare center trigger and adaptive camera movements, as well as toggle and smooth camera zooms.

Participants A total of 49 users, 42 men and 7 women, participated in the first study. They were aged between 20 and 40 years, with an average of 27 years. Participants were presented with the opportunity to enter a lottery for a chance to win a small prize.

Stimuli and procedure All soccer sequences were derived from the same international league match, recorded in 2013. While the ITU [70] recommends a duration of approximately 10 seconds for pairwise comparisons of video presentations, we placed higher priority in ensuring that the soccer sequences contained more than one example of pan, zoom and tilt movements. Due to this, we extended the set sequence duration to 15 seconds. Automated camera movements were implemented subsequently, making sure that each movement and zoom contrast was presented four times. Each soccer sequence was therefore presented twice, separated by a two-second interval showing a fixation point on a black background. Stimuli contrasts were paired up so that either the camera movement or the camera zoom approach differed between the first and the second presentation. Although each paired contrast was presented four times, new soccer sequences were included for every pairwise comparison. Thus, participants watched 16 unique sequences, selected as the most suitable excerpts, where there are atleast a few passes, from the entire soccer match.

As stated above, we conducted the study using an online web-form so participants could complete it at their convenience. The paired video presentations were grouped in two stimuli blocks, with every contrast repeated twice within a block. Stimuli were counterbalanced with reverse-order for half of the contrasts, before they were randomised within each block. We created four randomised versions of the study, so that the random order varied between participant groups. In order to control whether subjective preferences depended on soccer viewing experience, we introduced the study with two questions to assess soccer interest and dedication; we also collected details on age and gender. Participants received no information on the camera implementations, instead they received instructions to select the version they preferred. Following the questionnaire and instructions, we included two practice trials to get participants acquainted with the task, which were succeeded by the 16 pairwise comparisons.

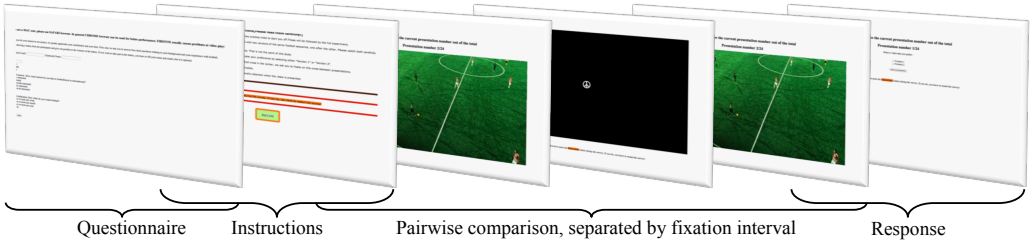


Figure 5.12: Visual outline of the steps presented in the user study. Participants started with the questionnaire and instructions, before moving on to the soccer sequences. These were introduced by two practice trials, followed by the full study. Each pairwise comparison was separated by a 2-second fixation interval and terminated in a response session.

Results

With every contrast repeated four times, the preference scores for the different conditions were added up for every participant. This resulted in individual counts the four combinations of camera movements and camera zooms, ranging from 0 to 4. In order to identify and weed out outlying preference counts, we also calculated the difference in scores between paired stimuli. This resulted in four mean differences, and we used the average of these to identify any scores that fell more than two standard deviations from the mean. Accordingly, we identified and excluded data from two participants, whose mean difference scores of zero indicated that they were unable to distinguish between stimuli. For the main analysis, we collapsed preference scores across stimulus combinations to obtain the overall number of times each camera mode was preferred by an individual. With two contrasts repeated four times for every camera mode, the highest possible preference count comes to 8. The degrees of freedom for the distribution are found according to the number of choices available. A Friedman rank test was used to analyse the preference counts from the remaining our 47 participants, revealing a significant effect of our camera implementations ($\chi^2(3) = 72.73$). To further explore the difference between stimulus combinations, we also ran three Wilcoxon signed-rank tests and calculated effect sizes from these. Results from the analyses are presented in table 5.1. Furthermore, we also explored the individual contrasts with a Friedman rank test, again revealing

a significant overall effect ($\chi^2(7) = 162.33$). These results are illustrated in figure 5.13, listed according to their Friedman rank scores.

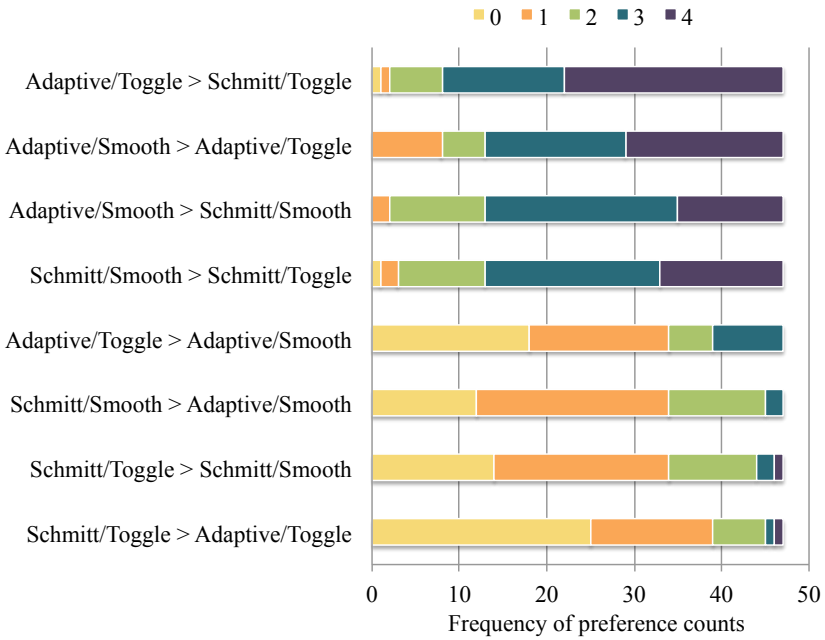


Figure 5.13: Frequency distribution portraying the number of times one stimulus was preferred over its contrast, accumulated across users. For example, in the first line, we see that 25 persons have preferred the Adaptive/Toggle over Schmitt/Toggle in all four repetitions. The maximum count of 4 corresponds to the number of repetitions for each pair of videos. Stimulus contrasts are sorted according to Friedman rank scores and plotted symmetrically.

From the collapsed preference counts and the ranking scores presented in the first part of table 5.1, the adaptive trigger movement combined with the smooth focal zoom emerges as the preferred camera implementation. Although not significantly different from the third rank, the adaptive trigger remained the preferred choice over the Schmitt trigger, ranking second when combined with the toggle focal zoom. These trends are also evident when looking at the ranked individual contrasts in figure 5.13. The adaptive trigger movement is preferred over Schmitt alternative for the vast majority of presentations, just as the smooth focal is the predominantly preferred zoom option over the toggle focal. In short, the opinions of 47 users clearly

Results from Study 1

Stimulus combination	Friedman rank score	Wilcoxon signed-rank test	Effect size
Schmitt/Toggle	1.32	-	-
Schmitt/Smooth	2.47	<.001	- 0.51
Adaptive/ Toggle	2.74	ns	- 0.14
Adaptive/Smooth	3.47	<.001	- 0.39

Results from Study 2

Stimulus combination	Friedman rank score	Wilcoxon signed-rank test	Effect size
Novice	1.31	-	-
Expert	2.24	<.001	- 0.52
Adaptive/ Toggle	2.99	<.001	- 0.41
Adaptive/Smooth	3.46	<.021	- 0.28

Table 5.1: Non-parametric statistics for the number of times a stimulus combination was preferred over its contrasts, averaged across participants and sorted according to the Friedman rank score. Wilcoxon signed-rank test indicates statistically significant differences between stimuli, these are reported in relation to the lower ranked stimulus (the row above). Non-significant contrasts are labelled *ns*, while non-applicable comparisons are marked with a hyphen (-).

demonstrate the preference for the adaptive trigger and smooth focal camera implementation.

5.4.4 Study 2: Man vs. Machine

Following the results from Study 1, we established that users prefer the camera movement combination with adaptive trigger pan and smooth focal zoom. However, an important challenge for such an automated system is to provide a viewing experience that can compete with a soccer match filmed by a manually operated camera. Hence, the second user study compares user preferences for the two highest ranked automated camera implementations with that of two human operators.

Participants With 14 females and 23 males, we collected data from 37 participants, none of whom had taken part in Study 1. Their ages spanned from 21 to 71 years, with an average of 29 years. Every participant was provided with the opportunity to sign up for a lottery that offered small prizes to be won.

Stimuli and procedure To compare automated camera movements with manual camera operations, we selected the two best-preferred stimulus combinations from Study 1. In so doing, we re-used half of the stimuli from the first user study and compared these to sequences with recorded camera movements. To record the camera movements, we invited an expert and a novice camera operator to watch the same soccer match. The expert was an experienced camera operator from a Scandinavian broadcaster, whereas the novice had experience with camera-view operations within games. After receiving instructions on how to move and zoom with the virtual camera using a joystick, the operators embarked upon the task of following the match by keeping the ball and action in focus. From their recordings, we selected 20 expert and 20 novice 15-second excerpts to contrast with the automated sequences. For further verification of the preference ratings from Study 1, we also contrasted the automated sequences with each other. Moreover, we contrasted the expert and novice recordings to see whether preferences differed between the two.

Study 2 proceeded in the same manner as Study 1, described in section 5.4.3. The only procedural distinction between the two studies is the inclusion of more stimuli, resulting in 24 pairwise comparisons.

Results

Response data from Study 2 are re-structured and analysed the same way as described for Study 1 in section 5.4.3, again with 2 outliers detected and excluded. For this analysis, we collapsed preference scores across stimulus combinations to obtain the overall number of times each camera mode was preferred by an individual. With two contrasts repeated six times for every camera mode, the highest possible preference count comes to 12. With the Friedman rank test indicating significant differences between the collapsed preference counts ($\chi^2(3) = 56.73$), we again followed up with Wilcoxon signed-rank tests. Results from these analyses are included in table 5.1. A second Friedman rank test revealed significant differences also between the individual contrasts ($\chi^2(11) = 177.15$), the ranked preference counts for these

are portrayed in figure 5.14.

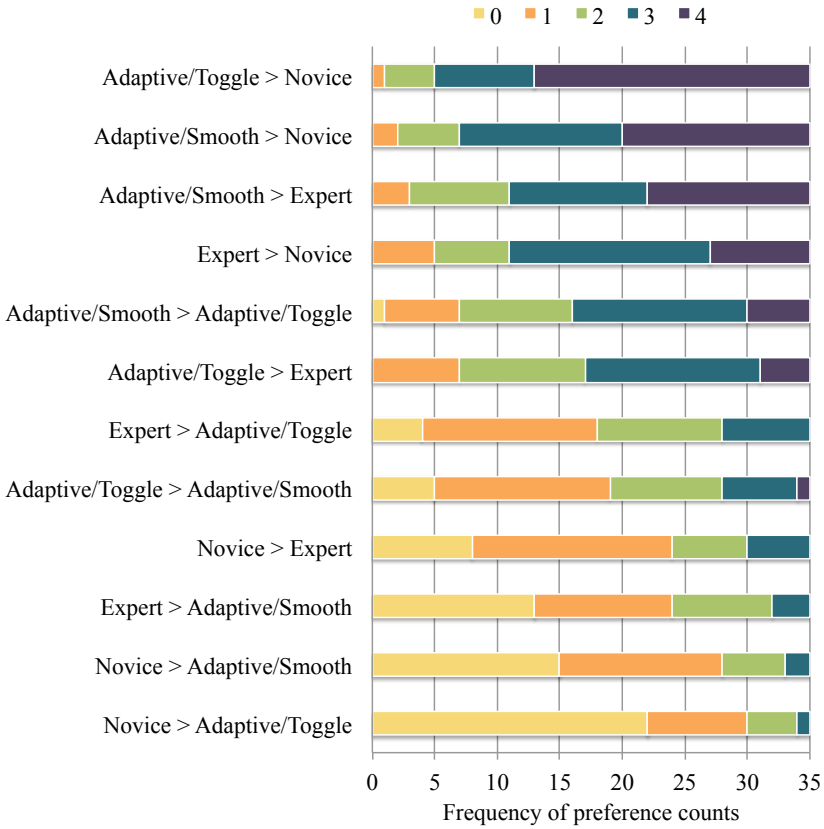


Figure 5.14: Frequency distribution portraying the number of times one stimulus was preferred over its contrast, accumulated across users. The maximum count of 4 corresponds to the number of repetitions. Stimulus contrasts are sorted according to Friedman rank scores and plotted symmetrically.

5.4.5 Conclusion

First and foremost, the results from Study 2 suggest that users tend to prefer automated over manual camera movements. Of course, the quality of manual controls is only as good as the operator and the joystick based operation. It must also be noted that the quality of the manual operation might vary in the real physical camera motion scenario. We considered this possible limitation and took precautions by including two camera operators, one expert and one novice. The higher ranking of the expert over the novice operator exempli-

fies the importance of the camera man's expertise. Despite our precautions, we cannot ascertain that users will prefer the automatic camera operations over any camera operator. However, considering the significant differences and the magnitudes of effect sizes for the presented conditions, the results show that our system is capable of outperforming the two human operators. Specifically, a consistent trend can be observed for both the collapsed preference counts (table 5.1) and the individual contrasts (figure 5.14), where the automated camera movements are chosen over the manual operations in the majority of presentations. Furthermore, the higher rank for the adaptive/smooth over the adaptive/toggle combination reflects the results from Study 1. As a final note to the subjective studies, we would like to emphasize that the experiments were conducted online. This implies a rather uncontrolled experiment setup including factors like video playout quality, user display quality and the general settings in which the experiment is performed. Hence, we would like to emphasize that the subjective studies merely suggest feasibility of creating an automatic control and we do not draw any significant conclusions from the collected data.

Results from Study 1

Stimulus	**Parametric statistics**		**Percentiles**		
combination	**Mean**	**Std. dev.**	**25th**	**50th**	**75th**
Schmitt/Toggle	1.77	1.40	1	2	3
Schmitt/Smooth	4.00	1.25	3	4	5
Adaptive/ Toggle	4.36	1.10	3	4	5
Adaptive/Smooth	5.87	1.56	5	6	7

Results from Study 2

Stimulus	**Parametric statistics**		**Percentiles**		
combination	**Mean**	**Std. dev.**	**25th**	**50th**	**75th**
Novice	2.60	2.06	1	2	4
Expert	5.37	1.68	4	5	7
Adaptive/ Toggle	7.43	1.67	6	7	8
Adaptive/Smooth	8.60	2.09	7	9	10

Table 5.2: Additional statistics to provide further insight into the distribution of the data collected from two user studies.

In addition to the non-parametric tests, means and standard deviations are presented in table 5.2 to provide further insight into the distribution of scores.

5.5 Discussion

Bagadus system aims to provide live interactive video services to football viewers. In cases where manual control of the virtual camera is desired, the system simplifies significantly. On the other hand, a viewer following a game might be interested in interaction but at a higher level. The viewer might place a request to the client to follow the ball/a single player or a collection of players. In such a case, the client has to provide an aesthetically pleasing virtual camera based on the position data from the ball and players. Even a coach is greatly advantaged by such a system, he/she can instantly request multiple virtual cameras focussing on different features. For example, one for the ball, one for a recently injured player, one for a recently exchanged player and one for the defense. So, building the entire system and a subjective evaluation of the results provides better overview of the challenges and possibilities in comparison to a theoretical evaluation.

In the two user studies, we have explored and analysed user preferences for automated and manual camera movements. The first study established that the average user prefers the adaptive trigger movement over the Schmitt trigger and the smooth focal zoom over the toggle; implications of these findings are discussed below. From the second user study, we found that the average user maintains the same preference for the adaptive trigger and the smooth focal zoom when compared to a human-operated camera. While this finding is specific to the current context and may not reflect the performance of all camera operators, the subjective preference for automated camera movements suggests a positive user experience with our system. Overall, the presented results are promising for the future acceptance and use of our system.

The user preferences between the toggle and smooth zoom is slightly ambiguous. From the user study, it is clear that the smooth zoom is preferred, but toggle zoom provides the advantage to switching to an overview immediately. This when combined with smooth zoom for smaller ball changes can provide a nice aesthetic, yet functional camera motion that can keep the ball in field of view. Moreover, the zoom model currently is based only on the

position of the ball on the panorama. This can be significantly improved by incorporating game context into the model. Some of the things can be velocity of the ball, player arrangement and special events (penalty, corner or throw-in).

Furthermore, it must be noted that this study focuses on one of the several points from where the action is captured on the soccer field. When it comes to capturing from one point in live, the camera man has little freedom in the grammar [111] of the video. In an actual broadcast, the producer mixes several streams together and this is where the grammar come into place.

Moreover current day's visual tracking algorithms' recall is not practically applicable to real-life scenarios. Owing to this, we still have a large manual component when it comes to estimating the ball position. We are currently exploring algorithms based on multi-sensor data to track the ball with a high recall rate. When we track the ball successfully, we will be able to provide a complete system functional in real-time. However, we do have an accurate tracking of the player positions, meaning that the system can easily follow a single player or a group of players.

5.6 Summary

In this chapter, we presented approaches to steer the virtual camera and performed objective analysis on them. We studied the preferences of users among automatic approaches and also the human operated camera. We concluded that the simple automatic approaches have the potential to provide movement that is on par with a human operator - i.e., a user can specify something to track (and, if tracking data for this object is available), the system is able to generate a pleasant experience for the users within the system's real-time requirements.

To be able to get such an experience in the described version of the system, we need to transfer the entire panorama video and some tracking meta data to the client. The network does not pose any serious problems on the metadata, but the panorama video needs high throughput. The next chapter therefore addresses this problem using video tiling and DASH-like

techniques to reduce the bandwidth requirement and provide a good user experience.

Chapter 6

Tiling

The previous chapters explore the details of providing a live interactive camera services to clients. However, one of the major challenges in providing such services is to provide it to several concurrent users. In this chapter, we explore the costs of scaling such services to a large number of users. Initially, we present the costs of using the client as a thin client and as a device capable of doing more than just playing the video. We then solve the problem of transferring panoramic video to clients with a bandwidth constraint at the client side. This chapter is an extension and reorganization of content from papers [46] (Appendix H) and [47] (Appendix E).

6.1 Scaling Costs

There are two ways of providing virtual camera services over the network to users. The virtual view extraction can happen in two different places, using the principles presented in the previous chapters, either on the server or the client side. In this chapter, we explore the costs of both approaches.

6.1.1 Server Side

One approach is to perform the perspective projection on the server, then encode the output video, and transfer the encoded stream to the client. The job of the client in this case is simply to decode a video. It must be noted

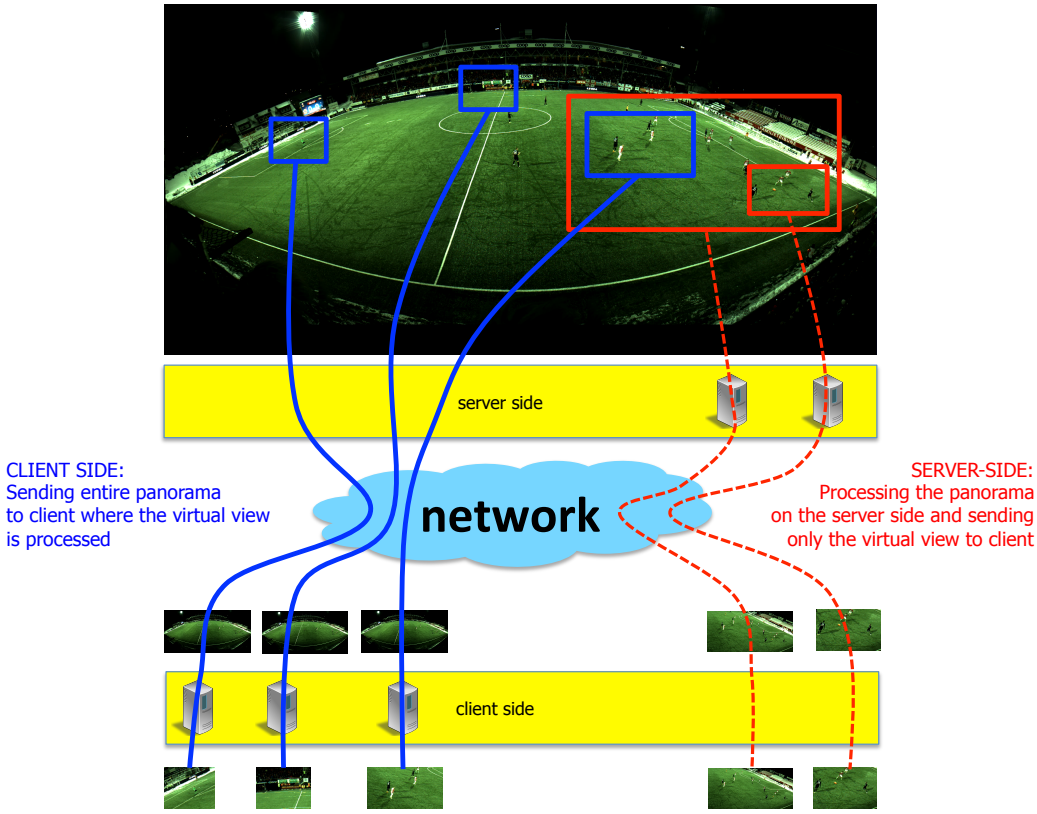


Figure 6.1: Multiple virtual views may be generated from the same panorama video. The virtual views can either be processed at the client- or the server side. The client side (blue) shows that the entire panorama video must be transmitted over the network, whereas the server-side approach (red) process first and then transmits only the finished virtual view video. (Note that the zoomed views are not rectangular crops as in this figure.)

that the tasks mentioned before could be different for each instance of the client. This way of realizing the system is shown by the red lines in figure 6.1.

In this scenario, the server-side processing becomes a large bottleneck as it must not only generate the virtual view, but it must also encode the video for compression for every individual client. This implies that every client connected to the server has its instance of virtual view generation and encoding running on the server. To be able to serve more streams from a single server, we experimented with a hardware-based video encoder implementation. Several modern multi-core architectures include a programmable hardware video

encoder (i.e., Intel QuickSync, AMD VCE and Nvidia NVENC). In our experiments, we used the second generation of hardware NVENC hardware encoder[1] from Nvidia [108] found in the Maxwell GPU architecture.

The GPU we use in our experiments is an Nvidia GeForce GTX 750 Ti. This GPU is based on the first generation Maxwell GM107 architecture. The GPU can encode 16 full HD (1920×1080) video streams at 30 frames per seconds [108]. Experiments showed that this was the limiting factor in how many unique views we could create in real-time due to the optimizations on virtual view generation from chapter 4. This implies that if we want to provide a service to 100,000 concurrent users, we would require clusters totaling about 6,250 GPUs in 2015. Such an initial installation costs about 937,500 USD merely for the GPUs. Running such a system continuously accounts, according to a simple lab measurement measuring the power consumption of a single GPU, for 0.227 MW of power consumption for GPUs alone. In table 6.1, a few estimates of number of GPUs cost and power consumed by GPUs are presented for different resolutions when requested by all 100,000 clients. Note that this number may be reduced further with lower resolutions, but our currently used version of NVENC is limited to 32 sessions per process. Managing multiple processes creates a significant overhead, but this will likely be solved in a future driver update, which is why we do not include lower numbers.

Resolution	# GPUs	Cost	Power
1080p	6250	937,500 USD	227 KW
720p	3125	468,750 USD	102 KW

Table 6.1: Estimated GPU resource requirements for 100,000 concurrent users.

[1]NVENC is a fully dedicated hardware video encoder which does not use the 3D engine on the GPU, leaving it free to perform other compute tasks. The encoder supports resolutions up to 4096x4096 (4K resolution) at 30 frames per second, and can encode multiple streams in parallel. Support for advanced parts of the H.264 standard such as MVC, B-frames and CABAC entropy coding are also present. The encoder API has several presets defined for different encoding scenarios. We use the high performance (HP) preset for our experiments.

6.1.2 Client Side

In Chapter 4, we introduced the virtual camera idea, where the extraction of the virtual view was performed on the client side as shown by the blue lines in figure 6.1. This approach requires the entire panorama video to be transferred to the client. The panorama texture is then used to create the virtual views on the client device itself. Since the process is client side, the user interaction is instant, i.e., the view is changed already in the next frame. However, even though this approach reduces the computational requirement on the server side, it demands for large bandwidths per user.

In our current setup, the average size of each *3-second* segments of the panorama video is approximately 2.1 *MB*. The bandwidth requirement for the client becomes about 5.7 Mbps merely for the transfer of the panorama video (the segment size depends on multiple factors, e.g., weather conditions, GOP structure). Even though this number is feasible on broadband networks, we still have some time to go to achieve these rates on shared mobile networks [110]. In addition processing the panorama video on small mobile devices can also be a challenging task. This is, however, changing as modern GPUs are included in modern mobile devices, too. Moreover, these numbers are estimated for a panoramic texture that is of the size 4096×1680. In future systems, it will be desirable and feasible to get a much higher resolution panorama to provide a higher quality zoom. Even though cache servers at different locations reduce high bandwidth requirement out of the panorama servers, for example using CDN-like infrastructures, the last-mile to the client still poses a huge problem in delivering the entire panorama at high quality.

Even after the panorama is successfully transferred, the client needs to process it so that a virtual view can be extracted. Earlier, we demonstrated that this can be accomplished in real-time on commodity graphics hardware in Chapter 4. It must be noted that when the virtual view is extracted on the client-side, it need not be further encoded. It can be displayed directly on the screen from the GPU. However, it must be noted that our experiments are performend on desktop hardware and the implications of doing the same

on mobile hardware might be different. Power consumption will become a priority alongside the real-time performance.

6.1.3 Summary

The server-side approach, where we process all the information on the server and transfer only the virtual view encoded into H.264 to the client, runs quickly into problems when scaling to a large number of users. However, the server-side approach implies that the client device just needs to be a thin client device that can simply decode an H.264 stream and display it on a screen. The client-side approach demands more processing from the client device and has an increased bandwidth requirement for smooth virtual view experience.

Recently we have seen that the client devices' capabilities have extended beyond just acting as a thin client. Clients are equipped with the processing power to perform more than just decoding. Hence, we chose the client side approach. However, an important cost that we have seen previously is the bandwidth. On the other hand, the entire panorama is not required to extract the virtual view. So, we used *tiling* to reduce the bandwidth costs.

6.2 Introduction to Tiling

When splitting te panorama into multiple smaller pieces using tiling, we divide the panorama into a collection of tiles that are of equal spatial size as we can see in figure 6.2. We then store these tiles in different qualities like segments in a HAS-type of solution. When the user is viewing a specific region on the panorama, she will only request the tiles corresponding to those regions in high quality and the rest in low quality. Our hypothesis is that we can design a scheme where we potentially save bandwidth and not interfere with the viewing experience of the user.

Figure 6.2 shows the basic idea of tiling. One can see that the virtual view is showing the area around the left goal post. The panorama is divided into 8×8 tiles. The tiles required for extracting virtual view are colored and the

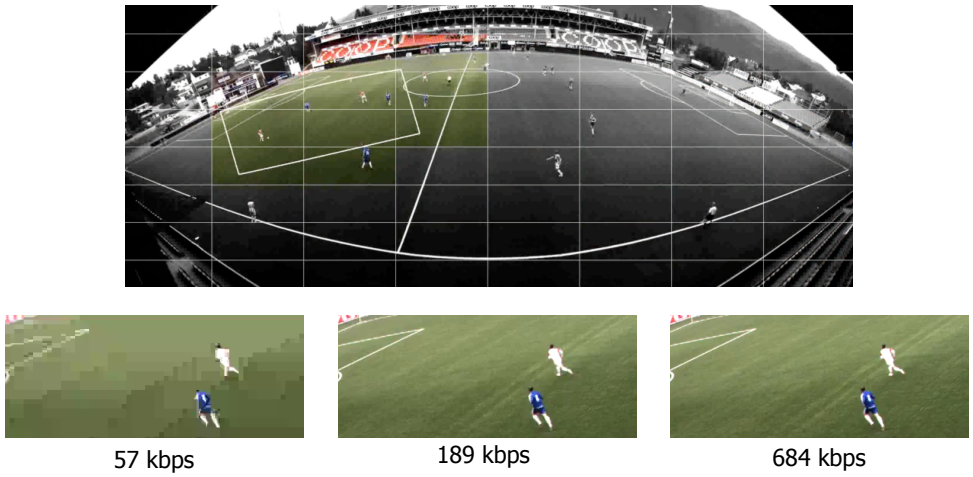


Figure 6.2: Using tiling on the panorama with tiles in different quality.

rest are not. The figure also presents a tile in multiple qualities. For efficient encoding, the tiles must be encoded in segments similar to DASH. Usually, the advantage of having longer segments is that the number of intra encoded frames can be limited to one per segment and thus high encoding efficiency. However, the disadvantage is that one cannot update the quality of a tile during the segment and has to wait until the end of the segment. There, it becomes extremely important to decide which tiles to fetch depending on the user operation of virtual camera. In the following sections, we discuss related works, our implementation and *tile selection* approaches. Later on, we discuss evaluation metrics and evaluate various tile selection strategies.

6.3 Related Work

A considerable amount of research has been aimed at the delivery of personalized views to users, addressing both the problems of regions-of-interest and the scalability of distribution. In this section, we give a brief introduction.

Only a few works [77, 106] discuss the distribution of live panorama video and even those lack a complete evaluation. Most industry projects transfer the entire panorama before starting the interaction, leading to a not *true-live*

component. However, YouTube 360 has just released the first 360^o videos that deliver 4-sided cube-panorama videos stitched into a single video stream and allow pan and tilt operations (no zoom) in the Chrome browser.

Streaming Options. Tiled video can be processed into an individual stream for each viewer on the server side [90], but this approach does not scale to a large number of concurrent viewers who can chose individual views.

All distributed tiling systems face the challenge of user interaction that changes the user's view rapidly, requiring new tiles between two consecutive frames. Users can notice a delayed reaction to their interaction within a few milliseconds [118]. To avoid this latency, tiling systems that extract views on the receiver side choose to retrieve all tiles (within interaction range) at all times, but at a less than perfect quality to save bandwidth.

HTTP Adaptive Streaming (HAS) is well-suited for this multi-quality delivery because it can deliver multiple quality levels to large audiences with the help of standard Web caches to increase scalability. However, retrieval decisions can only been made on segment boundaries, which means that visual quality can be reduced for several frames after user interaction affects the required tiles.

Faster quality improvement could be achieved by downloading a higher quality version of a segment that comes into visual range, decode it, skip frames that have already been played out at low quality, and continue with high-quality frames. The technique puts sudden high demands on download bandwidth and decoding. Alternatively, Scalable Video Coding (SVC) Mid Grain Scalability (MGS) could be combined with HAS [145]. Quality could be increased by retrieving an enhancement layer, which puts less load on bandwidth, and allows the receiver to improve frame quality immediately after skipping to the correct frame in the enhancement layer. However, an H.264 SVC-encoded video has 10% bandwidth overhead per enhancement layer compared to a non-scale video of the same quality [81].

Push-based streaming systems are an alternative because they can encode each tile as a continuous stream. Solutions that require multicast [61, 99] cannot be used on a large scale due to the lack of IP multicast. But also in a unicast solution, a push server can respond to a receiver's request for

higher quality within one Round-Trip Time (RTT) of a user request. One method works by updating Session Description Protocol (SDP) [127], which can switch the unicast delivery of layers on and off, but of course, the SVC overhead mentioned above applies here as well. An even faster method is based on Real-time Transport Protocol (RTP) [150], which can send a bit-rate request and instruct the server to send new Intra frame as soon as possible. This option is interesting, as it works either with SVC (suffering the mentioned overhead), with non-layered codecs but live encoding (or transcoding) at the sender, or a set of parallel streams where switching is supported through SI/SP frames [75]. The overhead of the SI/SP method lies between the other two approaches. All of these RTP-based methods have in common that packet loss can occur, and it is therefore today usual to use MPEG 2-TS packaging [57], but this in itself incurs a 20% bandwidth overhead [125]. However, DASH-like Apple HLS uses MPEG2-TS as well.

Considering that all of the approaches demand that the base-layer quality of all reachable tiles is streamed at all times, the bandwidth overhead of the various alternatives to HAS seemed too large for our scenario. We have therefore chosen a HAS with 1-second segments and discuss the quality implications of the qualities switching delay below.

Tiling Approaches Using HAS. Even though not directly related to the cylindrical/spherical panorama systems that provide free PTZ camera movement, there are some works [26,51,78,90,98,119] that provide an approximate interaction. [98] discuss tiling in interactive panorama video. However, their panorama is a perspective one and the virtual camera performs merely cropping, which is identical to cropping from a high-resolution video. Similarly, [90] provide zoomable playout on mobile devices for bigger resolution videos. [51] present an approach for the zoomable video where the tiles are optimally selected and sent from the server side. [146] performed a user study to determine the effect of tiling on the zoomable video presentation. Except for [98], these works do not support a completely random PTZ camera. [78] present a similar system where the tiles are encoded at multiple qualities and retrieved depending on the current view, however they do not discuss smooth random movement. Their interface is similar to that of a zoomable video,

where you can pick a portion of the entire video presented in a thumbnail and that part is cropped from the full resolution and presented. [26] present a tiled-streaming system where PTZ operations are performed simply as cropping of a high resolution video. Hence, to our knowledge, our work is the first to handle the problem of tiling and discuss its trade-offs in the context of a random PTZ camera on a cylindrical panorama texture.

6.4 Implementation

The tiling generation and retrieval operations are highlighted in figure 6.3. All components run in real-time, and the user can thus control the virtual camera during a live stream.

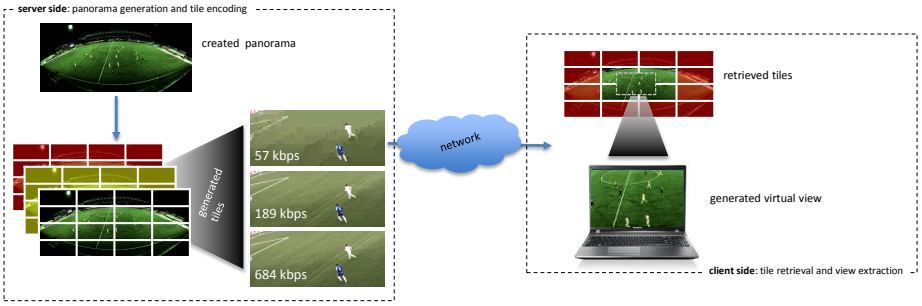


Figure 6.3: At the server side, we divide the generated panorama video into 8x8 tiles, and then encode each tile in different qualities. The client retrieves tiles in qualities based on the current position of the virtual camera (high quality tiles for the virtual view and low quality (red) tiles outside the field of view).

6.4.1 Server Side

The idea for the server is a simple HAS like video server. As described in section 3.5.1, the cylindrical panorama images are generated from five 2K cameras whose the shutters and exposures are perfectly synchronized and the seams are calculated dynamically for every frame. Then, for the tiling, the panorama frames are divided into 64 tiles (8x8), and one video stream is generated for each tile. Each video tile is encoded into 1-second segments

at multiple qualities (and bit rates) using *libav* and *x264*. Each tile can then be requested individually by client using HAS.

6.4.2 Client Side

Once the tiles with multiple qualities are available on the server, the client fetches tiles and generates the virtual view from the retrieved panorama. The task of the client is to retrieve high-quality tiles for the virtual view and lower quality tiles for the surrounding tiles. It can be seen in figure 6.2 that the high quality tiles are required only in the colored area. Thus the client is able to supply the user with a high quality virtual view, while at the same time trying to save bandwidth compared to the full quality panorama retrieval approaches discussed in the previous section.

However, the system must fetch, spatially, every tile in the panorama video, whatever might be the quality. In this way, the system can still provide data if the user interactively moves the virtual camera in contrast to presenting black areas or a static image if none of the surrounding tiles are retrieved at all. To accomplish this, the client is designed as shown in figure 6.4. There are four major components in the client system, (i) a File Manager, (ii) a Decoder, (iii) a Renderer and (iv) a Tile Selector.

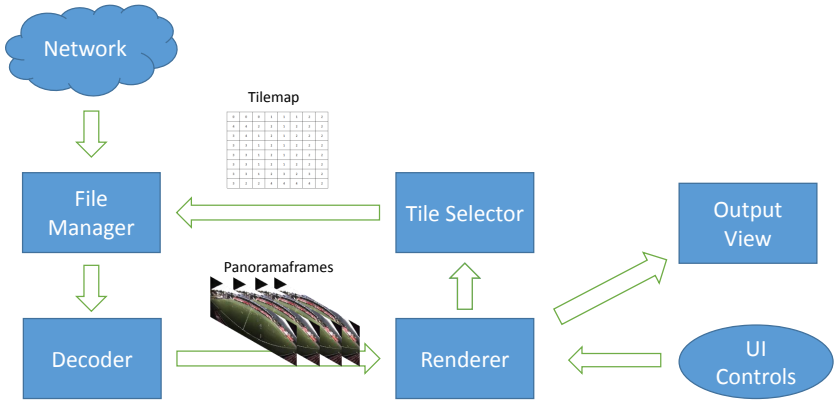


Figure 6.4: The architecture of a client supporting tiling.

File Manager. The File Manager component is responsible for request-

ing appropriate tiles in a given quality from the server (determined by the Tile Selector described below). The H.264 byte stream corresponding to the tiles is transferred as fetched and forwarded straight to the Decoder module instead of saving it as a local file, thus bypassing the disk.

Figure 6.5: Sample output frame from the decoder module.

Figure 6.6: Extract from the frame in figure 6.5 to show the difference in tile qualities.

Decoder. Once the tiles are available from the File Manager, the Decoder module starts decoding frames and pushing them to a common panorama texture. Since the tiles are spatially independent of each other, this process is heavily parallelizable. The operations are frame-synchronized to avoid placing a frame from two different tiles at different time instants into the same panorama frame. Figure 6.5 shows an example frame[2] where one can observe that the panorama frame is reconstructed from different quality tiles.

[2]Due to possibly limited resolution of printers, it is recommended to analyze the images on screen.

Renderer. As soon as a panorama frame is decoded, it is pushed to the rendering module. This module is responsible for creating the virtual views using the PTZ parameters from the user's requirement of the virtual camera direction. In addition, it provides user interaction or virtual view controls. In most interactive systems, the functionality of the Renderer is limited to this. However, in order to support tiling, we need to save the information of the panorama parts that are currently being viewed. This information is transferred to the Tile Selector module which again uses the information to select the tile qualities for the next iteration. Figure 6.6 shows an example virtual view.

Tile Selector. Once a frame is displayed, the panorama location from where the current view is extracted is transferred to the Tile Selector from the Renderer. This information plays a crucial role in selecting the next tile set. The most complex component is the Tile Selector, because tiles must be retrieved in order to have a high quality view, but still at the lowest possible bandwidth, i.e., high quality tiles in the colored area of figure 6.2, low in the grey area, where the complicating factor is the unknown movement of the interactive view during the segment.

Finally, it important to point out that all these modules need to perform in real-time to provide a smooth interactive experience to the user while keeping the bandwidth consumption at a minimum required level. One can observe that this can be challenging task at the Decoder module, where several videos are expected to be decoded concurrently in real-time and also frame-synchronized.

6.5 Tile Selector Approaches

There are several possibilities as to how retrieval of tiles can be achieved. Here, we present a few schemes considered in this thesis.

As described before, the Tile Selector is responsible for determining appropriate qualities (and bitrates) for the different tiles and adapt according to the viewer interaction. It must be mentioned here that the tile selection approaches only aim at reducing bandwidth and maximizing the quality of

the view at the same time, however, extensive work has been done before to adapt the quality to the network and CPU conditions. This kind of optimizations is out of the scope for this thesis. Instead, we aim at optimizing the quality for user interaction.

Let $Q = \{q_0, q_1, \ldots, q_{n-1}\}$ be the set of n available quality levels and T_i be the tile quality at tile i, then the problem can be written as a simple labeling problem in equation 6.1. The quality levels are in a decreasing order where q_0 is the highest quality tile.

$$T_i = q \quad \text{where} \quad q \in Q \tag{6.1}$$

There are several ways to perform this labeling, which will ultimately influence the bandwidth consumed and the user experience of the system. A *binary tile occupancy map*, containing information on which tiles are currently used to generate the virtual view, is used in the labeling process. The binary occupancy map has $B_i = 1$ at tile i when the view needs pixels from the tile i on the panorama, for example the color and grey scale tiles correspond to the 1 and 0 in the map, respectively, in figure 6.2. Even using the same binary occupancy map, there are several ways to select a tile quality, and below we briefly outline some of the algorithms evaluated in this study. The three first algorithms make a binary decision between a predefined, yet configurable, high or low quality. The last approach allows for a gradual (multi-level) decrease of quality depending on the importance of a tile.

6.5.1 Binary

The binary approach is a simple approach, where high quality is assigned to the required tiles and low quality to the ones that are not required (figure 6.7). Using the binary occupancy map described above, this becomes rather trivial. Hence, the binary approach can be formulated as following:

$$T_i = \begin{cases} q_h & \text{if } B_i = 1 \\ q_l & \text{else} \end{cases} \tag{6.2}$$

where $l > h$. The inequality is the only requirement, however, the choice of exact quality levels can be considered tuning.

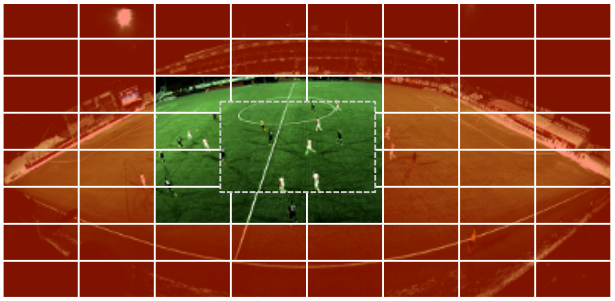


Figure 6.7: Tiled binary.

6.5.2 Rescaled

As seen in section 6.3, a commonly used approach for tiling is to send a low quality base thumbnail video and provide only the required high quality tiles [51, 98] (figure 6.8). To create the thumbnail video, the source video is down-scaled and stored. During the process of virtual view generation, the pixels from the available high quality tiles are used. For the pixels where the high quality data is missing, the thumbnail video is up-scaled and used, which can be considered as low quality tiles.

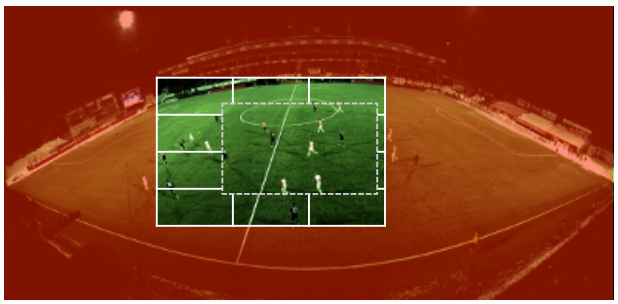


Figure 6.8: Thumbnail.

6.5.3 Prediction

When a user moves the virtual camera, there is a chance that the view will be generated by some low quality tiles since the tile quality is only changed at the segment boundary. In order to lower the probability that this occurs, it is beneficial to try to predict future movements and retrieve a higher quality tile if there is a high probability that the user moves the view into a tile (figure 6.9, i.e., similar to the tiled binary, but where the high quality area is enlarged according to the prediction). In this respect, it is beneficial to predict the path across several frames in future. There are several models available for prediction. However, to keep the comparison to the state-of-the-art consistent, we used the Auto Regressive Moving Average (ARMA) prediction [98]. Here, let θ_t be the position and $\delta\theta_t$ be the velocity of the view at time instant t. The velocity at the current instant is estimated as,

$$\delta\theta_t = \alpha\delta\theta_{t-1} + (1-\alpha)(\theta - \theta_{t-1}) \tag{6.3}$$

Then, the future position at $t+f$ is estimated as

$$\hat{\theta}_{t+f} = \theta_t + f\delta\theta_t \tag{6.4}$$

where f is the number of frames predicted in future. This can be used straight away to figure out a future binary occupancy map. This map can be used in any of the approaches mentioned here. But for the sake of comparison, we use the Predictive approach only with the Rescaled approach.

Figure 6.9: Predicted.

6.5.4 Pyramid

The pyramid is a scheme, where we chose qualities with a gradually decreasing quality according to the distance from the virtual camera (figure 6.10). Here, we introduce the term *priority* (p_i) that varies in $[0, 1]$, where 0 means highly important to 1 least important. Depending on the importance, we fetch the corresponding quality. But there is another catch. If we just decide on the importance, we might end up fetching high quality for a lot of tiles for a zoomed-out virtual view. Here, the maximum quality level (q_{max}) comes into picture. This quantity depends on the number of high priority tiles. We select q_H as the quality level to be used when all the tiles are used for the virtual view.

$$q_{max} = (\frac{\sum_{i \in T} b_i}{N}) q_H \tag{6.5}$$

$$T_i = \begin{cases} q_{max} & \text{if } b_i = 1 \\ q_{max} + p_i(n - q_{max} - 1) & \text{else} \end{cases} \tag{6.6}$$

After q_{max} is calculated, we count the occupancy of the neighbourhood and then assign that as its p_i as shown in equation 6.7. As one can observe, there are several tuning parameters. One is the q_H, which determines the quality at a certain zoom level. The second is the selection of the neighbourhood itself, which can be determined by the weights α_j. We can either make the weights isotropic or anisotropic. Given the fact that one is more prone to pan than to tilt, anisotropic weights can lead to similar performance as the isotropic one while consuming less bandwidth.

$$p_i = 1 - \frac{\sum_{j \in \mathcal{N}} \alpha_j b_{ij}}{\sum_{j \in \mathcal{N}} \alpha_j} \tag{6.7}$$

6.6 Experimental Setup

The problem of bandwidth reduction is a strict trade-off of two conflicting constraints. One constraint is the bandwidth itself, which can be measured

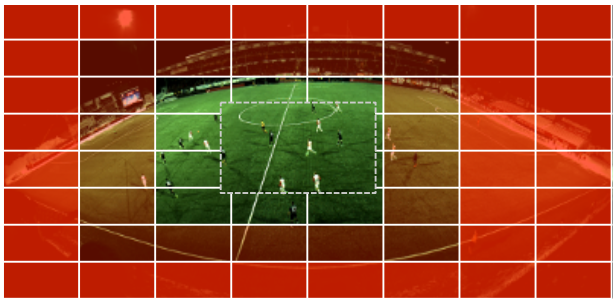


Figure 6.10: Pyramid.

straight away as the rate of data transferred. The second constraint is the quality of experience, which is not trivial to measure. When developing approaches, we need to consider how well the approaches are performing with respect to these constraints and which approaches provide the best trade-off between bandwidth consumption and quality. We compare the two different pipelines in figure 6.11, and we use the final output (the rendered virtual view) for comparison. In the tiling pipeline, we first create the tiles in different qualities, then fetch tiles in different qualities and then the view is rendered. Whereas, the original pipeline is directly sent to the virtual viewer and the view is rendered. We do not use high-quality tiles in the original pipeline, because high-quality tiles are transcoded from the original panorama.

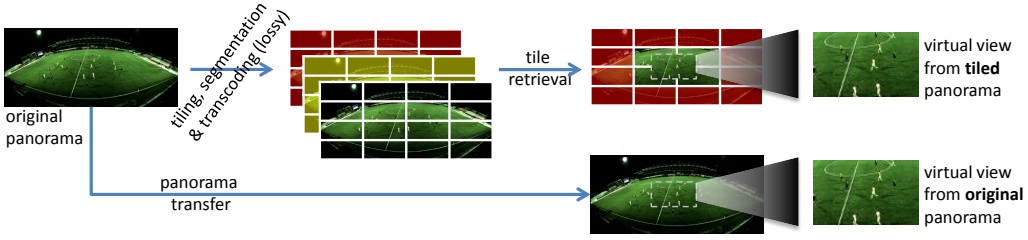


Figure 6.11: Pipeline differences: original vs. tiled panorama.

We assessed the quality selection schemes described in section 6.5 by comparing their performance on the entire first half of a soccer game (approx-

	q_0	q_1	q_2	q_3	q_4
Mean (all tiles)	21.0	14.8	7.4	4.7	3.0
Std dev. (all tiles)	9.8	6.8	4.1	2.4	0.9
Mean (corner tile)	9.5	8.0	4.5	3.1	2.5
Std dev. (corner tile)	1.4	1.1	0.3	0.07	0.03
Mean (center tile)	33.2	22.4	10.4	6.0	3.6
Std dev. (center tile)	9.6	6.7	3.4	1.8	0.9
Mean (Panorama)	1344.4	944.2	476.3	301.6	191.8
Std dev. (Panorama)	110.1	62.0	21.1	11.4	6.1

Table 6.2: Statistics for the tiles. All values are in KB/s.

imately 47 minutes). We used five quality levels with increasing Constant Rate Factor (CRF) values, the levels along with the CRF values are $q_0(21)$, $q_1(24)$, $q_2(30)$, $q_3(36)$ and $q_5(48)$. Table 6.2 presents statistics regarding the tiles over the period of 47 minutes. The *corner tile* refers to the left hand upper most tile and the *center tile* refers to the tile approximately in the center of the field. We also present the bandwidth statistics for the entire panorama if it were to be fetched in a single quality. It is a rather challenging task to compress the data for 47 minutes into a few values. However, we can observe the general trend in the table. The corner tiles, where there is no activity and mostly black pixels, are best compressed. Whereas, the tiles in the center, where most of the action takes place, occupy the most bandwidth. We can also observe that even though at individual tile level, the standard deviation is about 50% of the mean, at the level of panorama, the standard deviation is significantly low (less than 10%).

For the investigation, we compared the quality of four pre-generated sequences of PTZ operations, called *paths* ($s1 - s4$). We created four paths whose pan/tilt operations follow the general soccer game flow, but the zoom varies as described and labeled in table 6.3. The quality selection methods were labeled as shown in table 6.4.

6.6.1 Paths

In order to study the effectiveness of the approaches, we tested all the approaches using the same paths and studied various costs. The path classes

Label	Path
s1	The virtual camera is severely zoomed-in
s2	The zoom is at a medium level
s3	An overview video where the view is zoomed-out
s4	A dynamic zoom factor depending on the situation

Table 6.3: Paths: sequences of PTZ operations.

Label	Approach
l1	Binary with q_0 and q_4
l2	Binary with q_1 and q_3
l3	Rescaled with no prediction
l4	Rescaled with 100 frames prediction
l5	Pyramidal with isotropic weights
l6	Pyramidal with anisotropic weights
l7	Pyramidal with isotropic weights (different parameters)
l8	Full Panorama input (no tiling)

Table 6.4: Labelling of approaches for analysis.

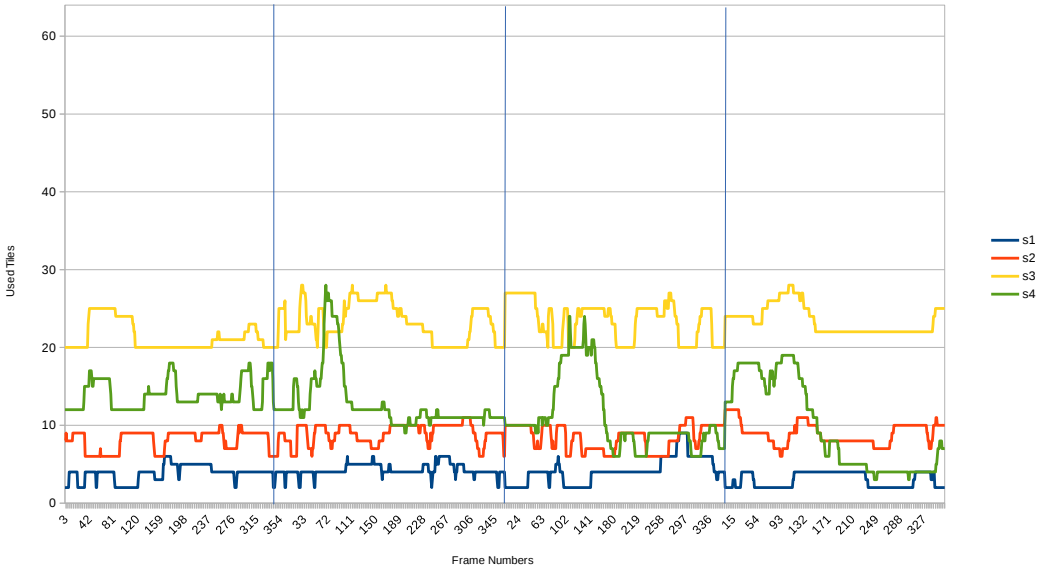


Figure 6.12: Number of tiles used along time, out of 64 possible, per frame in sequences of 12-second durations at 4 pre-determined time instants.

are as follows:

- *Zoomed-in Class (s1):* In this class, the viewer is assumed to be interested in zoomed-in sections of the panorama. For example, the scenario where the viewer is following a player closely.

- *Zoomed-out Class (s2):* In this class, the viewer is assumed to be interested in the watching the game at a zoomed-out level. For example, the viewer is interested in following the general game-play.

- *Medium-zoomed Class (s3):* In this class, the viewer is assumed to be watching the game at a medium zoom level. An example scenario would be to follow the defense of a specific team.

- *Random zoom Class (s4):* In this class, the viewer has full freedom to control the zoom to whatever level she wishes. This can be a scenario where the viewer picks the zoom depending on the game play.

Figure 6.13 presents examples of shots in each class. The same frame is viewed for different PTZ selections. The number of tiles being used is different for each view.

6.7 QoE Evaluation Metrics

To be able to evaluate different strategies, it is beneficial to understand the effect of the approaches on video quality. There are several metrics that are commonly used in estimating the quality of experience. However, most metrics are not suitable for truly understanding the quality.

The challenge for well studied objective metric scenarios has so far been to match subjective viewing experiences for videos of finite duration (8-12 seconds). Objective methods that try to solve this challenge and that have undergone rigorous independent testing [12] are meant for constant-quality videos (with uniform disturbances). They can estimate QoE if degradation in a video spans several frames and work well for individual HAS segments. However, they may not be suitable when the user is presented with a view that is stitched from several independently adapting HAS video tile (most recent ITU-T standard was published after this research). In this scenario, only parts of the video suffer from distortion and there are updates that can instantly change the degradation.

(a) Zoomed in.

(b) Medium zoomed.

(c) Zoomed out.

(d) Random zoomed.

Figure 6.13: Same frame at different zoom levels.

To compare the quality selection schemes in our paper, we have therefore conducted a user study to assess whether the image similarity metric SSIM [147] or OpenVQ, an independent implementation of a perceptual video quality metric described in ITU-T J.247 [71, Annex B], provide good estimates of subjective quality assessment. All user study experiments are performed using 12-second excerpts from the 4 sequences mentioned in table 6.3. We should explicitly mention that the subjective experiments are conducted only to study whether the objective metrics perform on par with what human subjects think of the videos in our exact usage scenario. Even though it can be tempting to analyse the input from the users to evaluate the tiling approaches, we do not perform such an analysis because correct practices in crowdsourcing for subjective studies are still under heavy debate in the research community.

6.7.1 PSNR

A commonly used measure in evaluating video qualities is PSNR. As [152] explains, it is solely a pixel difference metric, and quite unrelated to subjective experience. Already [95] explained its limits, while [65] have clarified that it can predict human preference in one particular case: when the same content has been encoded with different compression strengths.

6.7.2 SSIM

The Structured Simularity Index (SSIM) [147] is a metric for assessing differences between images. It is supposed to model human subjective experience quite well, but [34] have demonstrated that this fails for a variety of possible image degradations. In spite of this, SSIM is even used for estimation the quality of video. $x264$ makes encoding decisions based on it, and [148] construct a video quality assessment tool based on it.

6.7.3 OpenVQ

Perceptual Evaluation of Video Quality (PEVQ) is a full-reference algorithm that outputs mean opinion scores (MOS) as an objective video quality metric. After evaluation by the Video Quality Experts Group (VQEG), it has become part of the standard ITU-T J.247 [71]. Out of all candidates, PEVQ achieved the best error rate with respect to subjective studies conducted by two independent institutes.

Unfortunately, PEVQ is not freely available for researchers, so we used OpenVQ, which is based on J.247 Annex B, but not a one-to-one implementation of PEVQ. The *patented* temporal alignment has been dropped, because neither HAS nor RTP-based streaming suffer from temporal misalignment. Furthermore, flaws in the formulas in J.247 Annex B force a rather loose interpretation. The dataset used for evaluating J.247 candidates is not publicly available, but with a ground truth of ICCRyN datasets [6,112,115], OpenVQ achieves results close to those reported for PEVQ in J.247.

6.7.4 Missing Pixel Percentage

Mavlankar *et al.* [98] introduce the notion of missing-pixel percentage to evaluate the accuracy of their prediction and thus the quality of the virtual view. A *missing pixel* is a pixel in the virtual view where the corresponding high quality panorama data is not available for rendering. A percentage of missing pixels can be calculated against the total number of pixels in the virtual view. The average percentage of missing pixels across several seconds is used to evaluate various approaches in [97].

6.7.5 Pixel Histogram

In the previous tiling approaches [51, 97], where the selection is mostly a binary process using either a high quality tile or a low quality thumbnail, the missing pixel percentage can contain a lot of information about the quality. But, we also include three pyramidal approaches in the evaluation and they use multiple quality levels. Hence, we propose using a *Tile Histogram*. In a frame of the output virtual view, we count the percentage of pixels fetched from each tile.

6.8 Subjective Evaluation

As mentioned in section 6.5, we have tiled videos following a HAS model. An adaptation decision for each tile is made once a second. We do not aim at generating a single quality value for an entire 47-minute testcase, because we have not found any valid basis for doing so in the literature. Instead, we verify how well the above objective metrics describe user experience on a second-by-second basis.

6.8.1 Design of the User Study

We compared the results of the objective metrics with subjective evaluations across a range of tiling approaches. The user study was designed to investigate two aspects of the subjective perception of quality. We consider the

noticeability of quality distortions and the experienced *annoyance* related, but distinct. We ran two consecutive experiments, one to address the detection of tiling distortions, and one to address the annoyance resulting from the distortions. In addition, we included five-point absolute category ratings for subjective assessments of overall video quality, adhering to ITU-T P.911 [70].

In both experiments, participants watched sequences with durations of 12 seconds extracted from the sequences described at the beginning of section 6.7. These were originally chosen as representations of different football scenarios and hence provided variety to participants and served to increase generality. Since all sequences included pre-recorded camera panning and zooming movements, our final stimulus collection contained sequences with frequent tile shifts and varying changes in compression rate and video quality. In the detection experiment, we instructed participants to pay attention to the quality of the presented sequences and to push down the spacebar the moment they noticed a change for the worse, holding it down for the entire duration of the quality drop. The annoyance experiment followed the same procedure, only changing the instructions to ask participants to push down the spacebar while they experienced annoyance due to low video quality. At the end of each sequence, participants rated the overall video quality on a 5-point scale ranging from "bad" to "excellent".

In order to secure a sufficient number of participants, we recruited crowdsourcing workers. This approach requires some extra methodological considerations due to challenges that concern lack of task adherence and comprehension, and in turn, reduced data consistency [11, 58–60, 80, 120, 122]. Thus, we initially conducted 3 pilot studies to ensure that the experiments were presented in a succinct, but understandable, format. The first was completed by colleagues and students, the following two on crowdsourceing platforms Microworkers and Crowdflowers. Following each pilot, we adapted the experiments according to the received feedback. For the final study, we used Microworkers and collected data from different participants. Although we implemented quality measures such as gold samples and majority votes, the highly subjective nature of the task did not allow more than the most basic automatic filtering to exclude non-complying individuals. We excluded only

participants who failed to complete the experiment, and on manual inspection removed participants who had obviously attempted to circumvent the experimental tasks, altogether 15%. In total 246 participants completed the *annoyance* experiments and 242 participants completed the detection (*noticeability*) experiments. We employed a strategy where a key is generated at the end of the experiment individually for each participant. However, some participants managed to find flaws in our key-generation program and recieve a key without having to record response for all sequences. These are verified by running through all the input data and verifying individually whether a response has been recorded for all sequences after completion of the experiments. All other potential exclusion criteria were found to potentially exclude valid human perceptions as well. In this respect, Riegler *et al.* [123] provide an indepth discussion about various filtering approaches and their weaknesses on this specific dataset.

We then calculated the agreement between participants' quality ratings for each sequence using Fleiss Kappa [37]. Because this statistic depends on the number of raters and comparisons [133], we consider it in the context of the possible minimum and maximum values, which are established at -0.80 and 1. For the detection experiment, inter-rater agreements varied between 0.22 and 0.37 across the different sequences and quality conditions. The annoyance experiment yielded values between 0.24 and 0.39. With respect to the arguably subjective and variable measures of detection and annoyance, we judge these positive agreement scores as indications that participants adhered to the task at hand.

6.8.2 Performance of Evaluation Metrics

The analysis of user studies for perception is always challenging, especially when the users are expected to provide time-varying input. For example, our study aims at recording perception differences among users, but records also response time differences between them. Some techniques exist to over-come this like Dynamic Time Warping, which takes two signals that are *assumed* to be similar. However, due to the weakness of such assumptions, we ignored

response times and averaged user inputs across all users.

The results of our user study show a reasonably strong relation between the user input and OpenVQ, but also SSIM. We used Kullback-Leibler-divergence (KLD) [82] to estimate the information loss in approximating subjective results with the objective metrics, and KLD stays below 0.05 for path $s1$ and below 0.01 for the other paths. Figure 6.14 shows that both OpenVQ and SSIM can be closer to the average subjective ratings than PSNR.

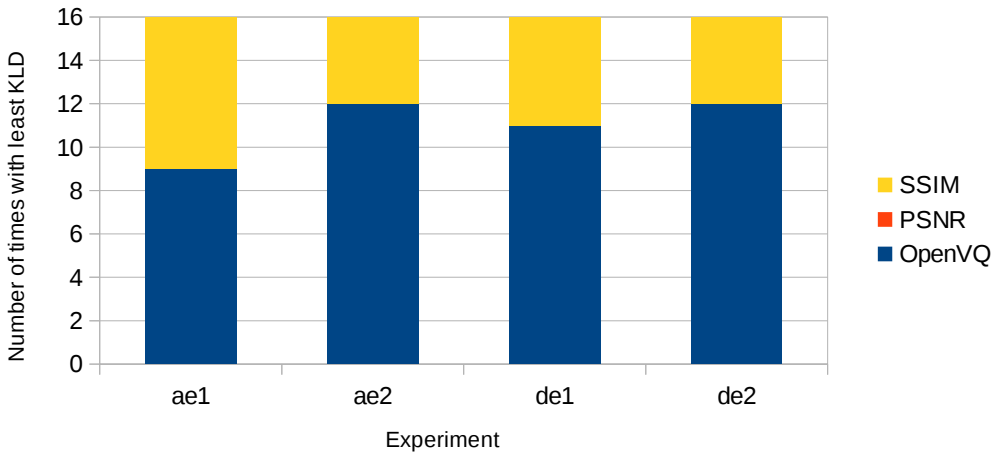


Figure 6.14: Number of times a metric had the least divergence from the user input in each task of the experiments among OpenVQ, PSNR and SSIM.

Although PSNR is unsuitable as a metric of visual quality (also quite easily shown to fail in the case where high- and low-quality tiles are merged into a single view like in figure 6.15), we present also PSNR results because they expose unexpected properties of the 1-second video segments. The PSNR results in figure 6.16a exposed regular severe degradation of the last frame in each 1-second segment. Although this is not noticable to a human observer even when single-stepping through the video, it is clear evidence of problems in *ffmpeg* or the way in which we use it. On visual inspection, stepping through each frame, we have not found any noticeable degradation. In conclusion, we use SSIM and OpenVQ metrics to further analyse the tiling approaches.

(a) l3s4 - PSNR: 30.39. (b) l7s4 - PSNR: 30.89.

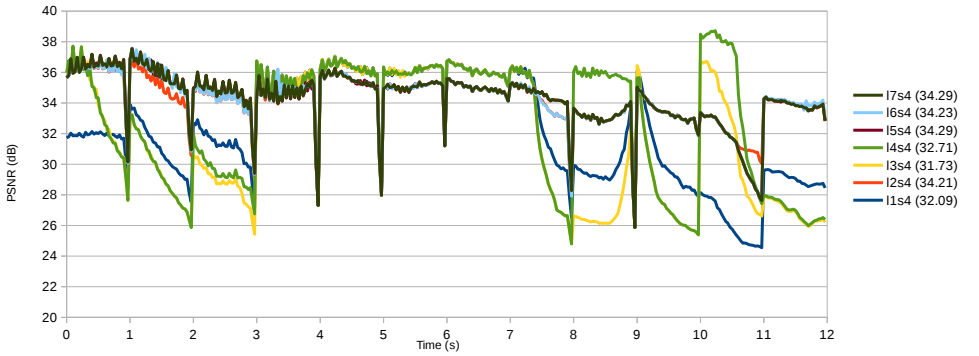
Figure 6.15: Example of severe differences within a frame (319), leading to similar PSNR values.

6.9 Results

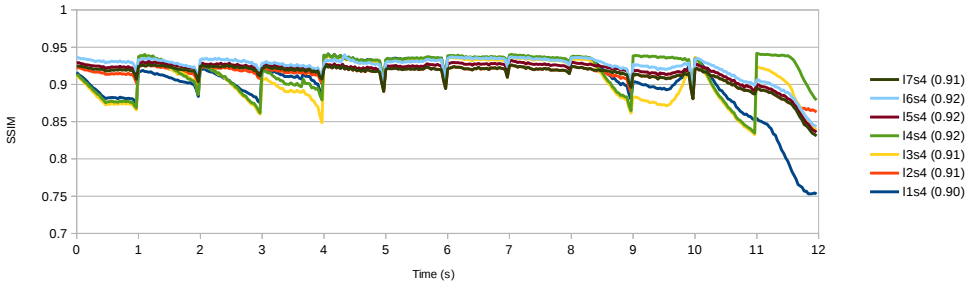
In this section, we mostly discuss the sequence $s4$ which is representative of the real-life virtual camera operation. However, using the other sequences we can observe zoom specific results. For example, $s1$ consumes the least amount of bandwidth due the required low number of tiles. We can also observe from $s3$ that when the user is interested in overview of the field, there is no need to fetch highest quality tiles.

6.9.1 Bandwidth

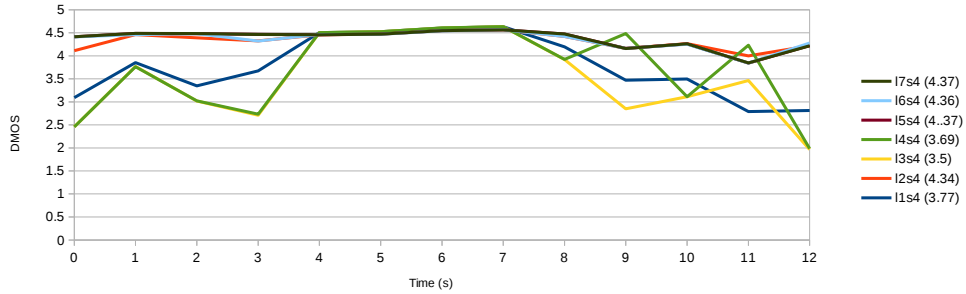
A simple way to determine the cost of network delivery is to measure the bandwidth. The most commonly used measurement is the average bandwidth along the entire run. Figure 6.17 presents a box-plot of each approach using different paths for the first half of the game. However, the interactive services can have different bandwidth requirements at different instances. Therefore, we use the running bandwidth profile to evaluate the performance of the approaches. Figure 6.18 shows the bandwidth profile for all approaches for a 90-second duration at 1000 seconds into the game. We can observe that there is some correlation with the number of tiles used at that time instant, which can be seen in figure 6.20.



(a) PSNR.



(b) SSIM.



(c) OpenVQ.

Figure 6.16: Different variation over the quality metrics across the 12 second clips. For reference, the average of each metric across the 12-second duration is also presented in parenthesis for each approach.

The methods are tuned to provide similar bandwidth with slight variations depending on the number of tiles used. However, it is quite evident that the approaches using highest quality tiles wherever required will have high bandwidth usage when a lot of tiles are used in the view. This can be

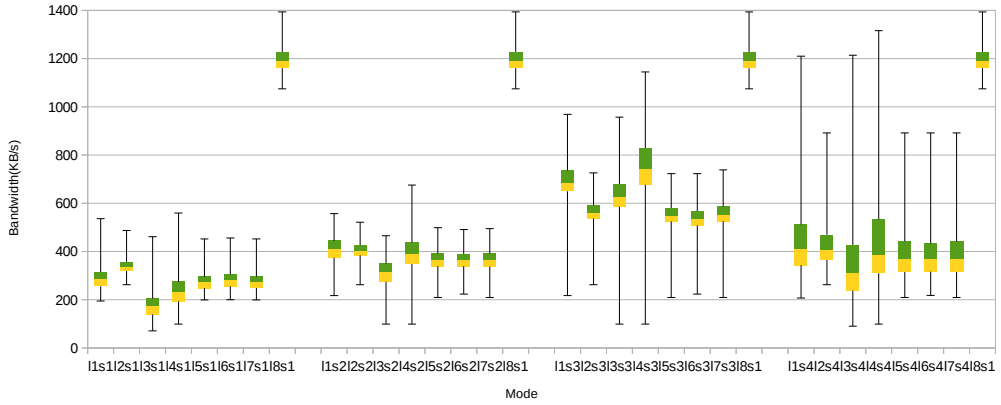


Figure 6.17: A boxplot of the bandwidth consumed in (KB/s) for different approaches over 2.834 seconds(47 min) representing the first half of a game.

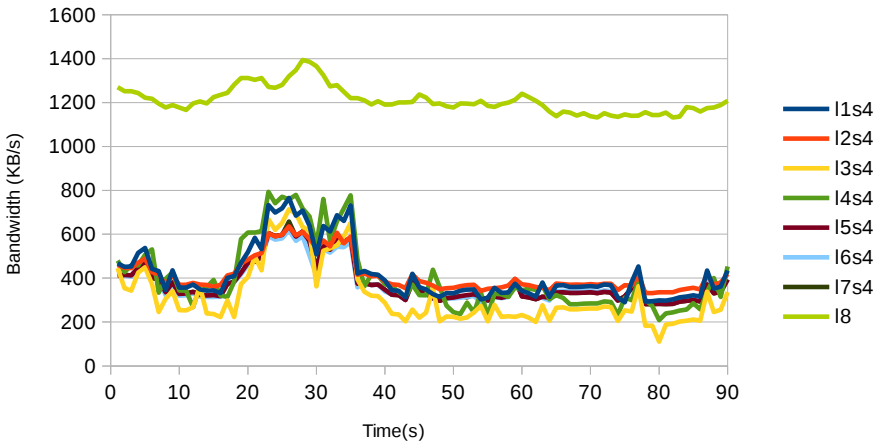


Figure 6.18: Bandwidth profile for 90 seconds duration in the middle of $s4$.

Tiles	Total	out21	out24	out30	out36	out48
2x2	17G	7.5G	5.0G	2.2G	1.2G	528M
4x4	18G	7.7G	5.3G	2.5G	1.5G	821M
8x8	23G	8.7G	6.4G	3.7G	2.4G	1.8G

Table 6.5: Size of the data for a soccer video of 6297 seconds using different tile granularity when compressing each tile with CRF values of 21, 24, 30, 36 and 48 on 1 second segments. In comparison, the size of the non-tiled panorama using the same segment length is 7.3 GB.

seen in the great bandwidth requirement for $l1$, $l3$ and $l4$. However, over the

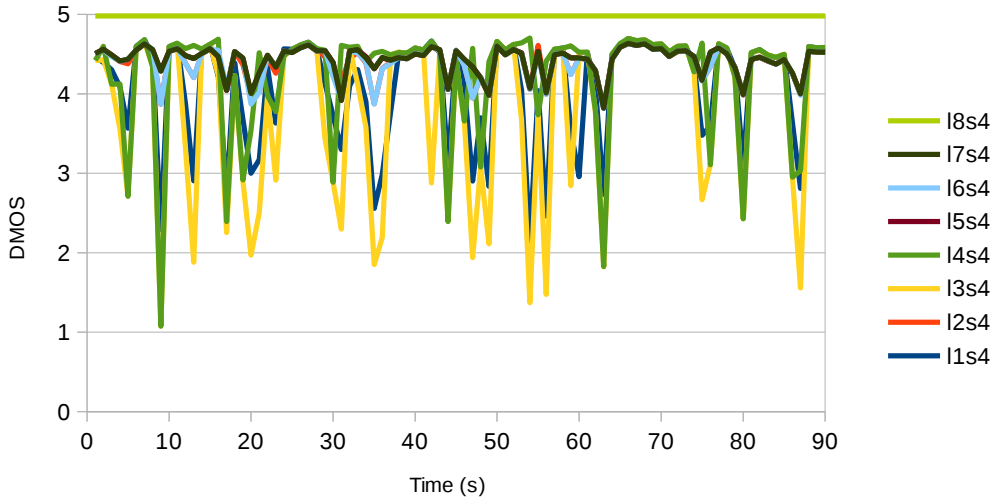
long run of the random zoom sequence ($s4$), which is probably most representative of a real scenario, the bandwidth consumption for all approaches is quite similar. For an estimate of the costs on the server side, we present the total disk space occupied by the tiled segments in table 6.5. Irrespective of the approach, it can be observed that the bandwidth savings are quite high, sometimes reducing the requirements to 25% of the full panorama. Hence, it becomes important to evaluate the approaches in terms of subjective quality.

6.9.2 Quality

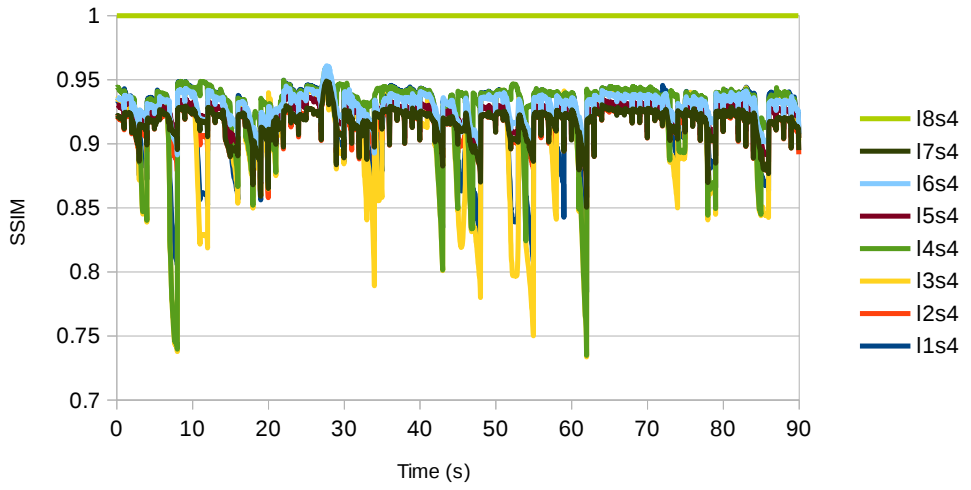
So far, no approach exists that can provide the best visual quality and low bandwidth usage at the same time during the entire virtual view operation. From figure 6.19, we can observe that all methods suffer from quality degradation at times. However, some approaches, especially the pyramidal ones, can provide decent bandwidth savings and also acceptable quality most of the time. The predictive approach is functional and provides improvement only when the actual positions match the predicted ones. However, with a completely random operation, this can be challenging even with sophisticated algorithms. Moreover, the prediction algorithms seem to be the most expensive on bandwidth. However, not all high quality tiles fetched are used for extraction of the virtual view.

In figure 6.19, we can see that a value of 0.93 for SSIM and 4.5 for DMOS runs along the time with drops depicting the quality changes during the virtual camera movement. These values imply that the visual quality of the tiled virtual view is on par with the original. Even in the drops, we observe that the pyramidal approaches perform better than the others. However, SSIM and OpenVQ are full-reference quality measures, which implies that the evaluation can only be carried out with the presence of the high-quality virtual view. However, there are ad-hoc measures that one can collect in the background without much resource consumption and that can provide some insight into the quality of a virtual view.

Mavlankar *et al.* [97] provide evaluation results also as average percentage of missing pixels for a 480×240 cropped view of 2560×704 pixels panorama.



(a) OpenVQ variation.



(b) SSIM variation.

Figure 6.19: Measured variation across 90 seconds at 1000 seconds into the soccer game for $s4$.

This 6.7% ratio is equivalent to using 4 tiles in a 64 tiled panorama ($s1$ from figure 6.12), in which case the average percentage of missing pixels of around 20%, from table 6.6, is coherent with their results. From table 6.6,

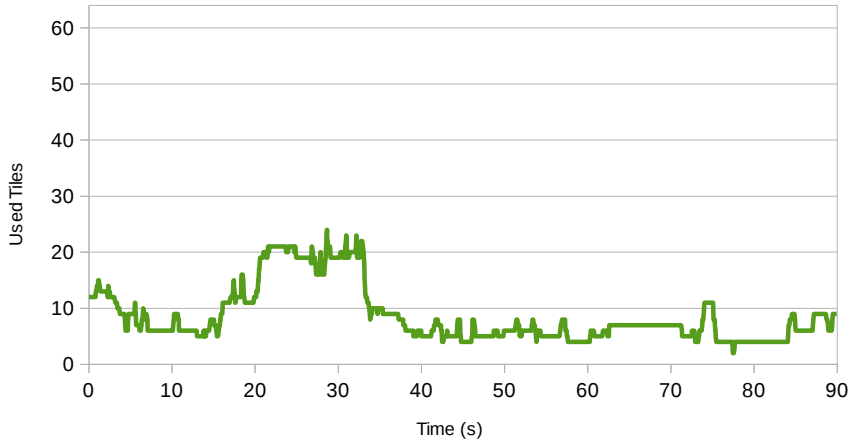


Figure 6.20: Number of used tiles across 90 seconds at 1000 seconds into the soccer game for $s4$.

Label/Sequence	s1	s2	s3	s4
l3	20.22	10.20	2.60	9.44
l4	18.20	8.56	1.94	6.53

Table 6.6: Average percentage of Missing pixels measurements over the entire first half of the game.

we can also observe that the missing pixel percentage varies depending on the zoom. An example profile of a pixel histogram is plotted in figure 6.21. One can observe some correlation between the pixel histogram profile and the variations in quality observed from OpenVQ or SSIM. Ad-hoc metrics like these can be used as reference to check the quality on the fly during the process. However, full-reference metrics provide the most accurate insights into the quality variations.

6.10 Discussion

The study presented uses HAS as the delivery method for tiled panorama video. From the analysis of quality metrics and bandwidth profiles for different movement path of a virtual camera, we make several observations for the various approaches. We find that the *pyramidal* approaches provide a stable quality across different zoom factors and random movements, i.e., it

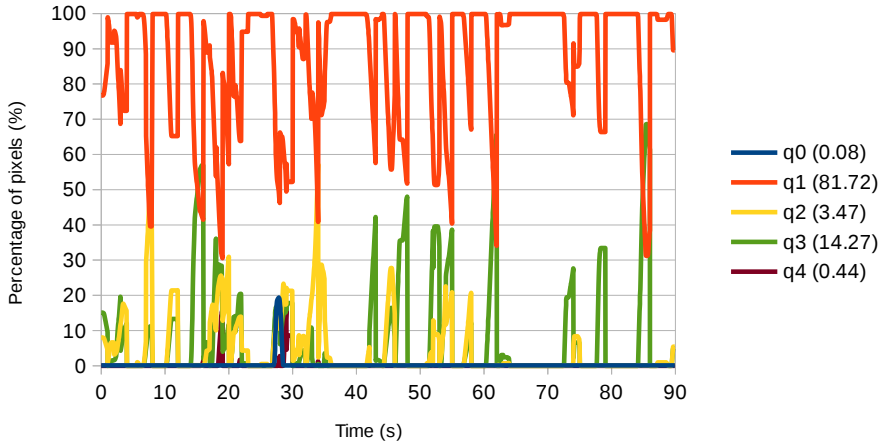


Figure 6.21: Pixel histogram across 90 seconds along with the average percentages for each level in the parenthesis.

is a good tradeoff between bandwidth savings and perceived video quality. When only a little portion of the panorama is used, we find that the *rescaling* approaches take up the least bandwidth, but the loss in quality is significant. The *prediction* results are not especially impressive when using a general prediction algorithm, and Mavlankar *et al.* [97] found that even context-based prediction does not lead in much improvement.

The adaptation strategies evaluated in this paper try to adapt the quality of a tile according to the movement of the view in order to have as good quality as possible in the area of the panorama used by the virtual camera to generate the view. There are, however, numerous works that similarly try to optimize the quality of traditional (non-panoramic) HAS streaming according to available resources. Clients of all major HAS variants, i.e., Apple HLS, Microsoft Smooth Streaming and DASH, have algorithms trying to have a high, stable quality. Additionally, researchers have presented approaches trying to optimize the segment retrieval, e.g., according to buffer occupancy [124] and consistent visual quality [87]. However, including this in the current study is out of the scope, but it is an interesting topic to pursue in the future combining optimal tile quality according to both the virtual view and the available resources.

We have also explored and analysed the effect of different segmented

streaming approaches on the quality for arena sport scenario, where the movement on the field is small compared to the entire field. It would be definitely interesting to explore them in different scenarios like the ones with *details but static* and *details with large movements*. However, these scenarios may require different treatment to achieve a good tradeoff between quality and bandwidth usage.

6.11 Summary

We have presented multiple approaches for tiling that can exploit the coding efficiency of H.264 to reduce bandwidth requirements for an interactive live PTZ system. We have evaluated the approaches using several methods and compared these methods for their closeness to subjective perception.

Based on our experimental results, we provide several conclusions. Overall, our results prove that pyramidal approaches reduce the bandwidth requirement and at the same time provides a similar QoE compared to a full-quality, non-tiled panorama system. Furthermore, utilizing the CRF parameter of H.264 provides better bandwidth savings and better visual quality compared to up-scaling a thumbnail video when the panoramic system is static and the movement in the scene is little compared to the scene itself. This is a rather common scenario for arena sports like Rugby, Soccer, Hockey, Cricket etc. Since a subjective study is a time-consuming and expensive way to evaluate the approaches, objective evaluation is a good alternative. We conclude that traditional evaluation methods will fail to correlate well with the subjective assessment of the experience, and a new metric, OpenVQ, closely capture subjective ratings.

Both the approaches and evaluation methods can be used with other interactive live PTZ camera systems as well. However, the tiling approaches, especially the quality levels, will require some parameter tuning specific to the application to gain optimal performance.

Chapter 7

Conclusion

The growing availability of high-speed Internet access has gone along with a growth in interactive and immersive multimedia applications, and panoramic video is a feature that is becoming more and more popular in various scenarios for its ability to increase immersion. We have designed and implemented a system called Bagadus in this context and used soccer as a case study. Sports like soccer have always provided opportunities for innovation in broadcasting. The challenges in providing interactive experiences in broadcasting to a large audience span several fields. One such challenge is to provide a real-time pannable and zoomable virtual camera to several thousands or even millions of users.

7.1 Summary & Contributions

Originally, Bagadus system was envisioned in the scope of soccer analytics. In this thesis, We mainly focussed on the video components of the system. We aimed at improving quality of the captured video, providing new ways of interacting with the broadcast and handling the challenges in real-time delivery of the interactive services. As a result, the single most important outcome of this dissertation is the working prototype of the live interactive video system running at Alfheim soccer stadium in Tromsø (used by Tromsø IL, a Norwegian elite club) and at Ullevaal stadium in Oslo (used by the

Norwegian National team).

From the technical point of view, there are several contributions from the dissertation. The contributions are covered in detail in the chapters, however, they are briefly listed as follows:

- The challenges involved in making a visually pleasing panorama automatically from a real-world outdoor stadium are several. We managed to generate a seamless panorama video of size 4096 pixels from multiple cameras in real-time at 50fps including features like HDR, synchronized capture and dynamic seam to solve challenging light conditions. In the process, we have made several contributions towards real-time panorama creation on a distributed panorama video pipeline.

- We present a virtual Pan-Tilt-Zoom (PTZ) camera in order to extract an interactive personalized virtual view from a panorama video in real-time. We present different approaches for virtual camera extraction using heterogeneous architecture and speeding it up using a parallel architecture. Even on a commodity laptop hardware, we are able to achieve speeds up to 300 fps for extraction of the virtual view.

- We developed methods for automatically steering the PTZ camera to user's requirements. In this mode of interaction, the user can request to follow the ball or a player or a group of players, and the system will automatically generate a virtual camera that smoothly pans/tilts/zooms according to the request using the data from the tracking subsystem.

- We also present valuable results from a subjective user study using the full pair-wise comparison tests to judge the performance of the automatic operation in comparison to an experienced human operator.

- A high resolution panorama video requires large bandwidth. We experimented with a DASH-based spatially segmented streaming approach to reduce the bandwidth required by splicing the panoramic video into tiles. We also demonstrate that, by fetching the appropriate quality tiles in relation to the user interaction, the quality of experience is not compromised, but the bandwidth consumption is greatly reduced.

In addition, the author has supervised several master students and published several papers (not all are included in the thesis). In particular, the thesis has contributed to six journal papers [40, 47, 84, 137–139], five conference papers [41, 42, 46, 143, 151], four technical demos [43–45, 103], two posters [76, 83], and one dataset paper [114].

Over the thesis, we touched upon several related works and the state-of-the-art. Due to the number of fields a real-time live interactive video delivery system touches upon, there is a large amount of work published. However, existing systems either (i) lack the real-time requirment, (ii) deal with only parts of such an interactive system, or (iii) do not elaborate on the system level details. Hence this thesis, detailing the challenges, research, development and evaluation of *Bagadus* system, extends the state-of-the-art by providing a comprehensive yet detailed journey into the world of real-time live interactive video delivery systems for sports. Finally, the entire code is made available as an open-source project [134]. This allows anyone to use the code and replicate our system for research purposes.

7.2 Future Scope

There are several places where further efforts can improve and extend the Bagadus system's capabilities. These include both extensions to functionalit and improvements to the technical solutions. Listing all is out of scope, but here we give some examples.

For example, the audio component has not been taken into consideration so far. However, audio plays an important role in the experience of a sports event. A similar approach, as in this thesis, could be followed for audio in terms of capturing, presenting and delivering it to the audience. In addition, multimodal data introduces further interesting challenges, like for example synchronization between different data streams.

We implemented the virtual viewer client for desktops and laptops. It would be interesting to take a step further and develop a similar service for mobile devices like cell-phones or tablets. This would introduce interesting research topics about the power consumption.

During the development of Bagadus system, a startup company (Forzasys AS) focussing on enabling users to create playlists from their favorite matches has been established. However, creation of highlights and such playlists can involve a lot manual work. Some researchers have focussed on automatically extracting highlights using several image processing techniques [4, 21, 27, 68, 103]. An interesting approach in settings like Bagadus, where the camera system is fixed and undergo minimal changes over time, would be to train a neural network that could automatically extract highlights. The accuracy and efficiency of such a network can benefit greatly from the fact that the videos are very similar in nature.

When we discuss the compression of panorama video, we use off-the-shelf H.264 encoding. However, H.264 is aimed at encoding general videos with great compression rate and low loss in quality. The H.264 performs a great job because the videos generated from Bagadus are mostly static with only a small percentage of pixels containing movement of the players. However, recent interest in VR applications definitely require further study and research into encoding of panoramic videos.

One field which we have not touched upon during the work in this thesis is *compressed domain processing*. The main data communication happens in compressed domain due to the size of raw data in videos. However, most processing happens after the video data is uncompressed. A major drawback of this is seen in the tiling chapter, where the amount of processing power required to decode several tiles in real-time is huge. In this regard, it can be beneficial to explore approaches to assemble a tiled video in compressed domain.

We discuss greatly about technologies enabling interaction with the videos. However, we only demonstrate simple use cases of such interaction. However, several interfaces can be designed using the frameworks developed in this thesis that suit the needs the users. There are several advancements in getting user input, like gesture recognition technology, haptic sensors, inertial motion sensors. Further work can explore using such sensor technologies to interact with the panorama video data in different ways.

An extension of the Bagadus-like systems in other domains, for example,

in medical field where a lot of data is generated during a diagnosis, would be very interesting. It can be beneficial to take the integration ideas from Bagadus and apply in such a system to save time for experts analysing the diagnosis data or to achieve *tele-diagnosis*.

7.3 Concluding Remarks

Finally, *virtual reality* is slowly emerging as a powerful field in several areas like medicine, sports, defense, survelliance and entertainment. In this thesis, we have researched enabling real-time interactivity in the broadcast of soccer videos. This led us on an interesting journey with a lot of challenges, approaches to solve them, and further new ways to go.

Bibliography

[1] A. Adams and R. Baker. *The print*. Little, Brown, 1983.

[2] J. Adams. Design of practical color filter array interpolation algorithms for digital cameras .2. In *Proceedings of the IEEE International Conference on Image Processing*, volume 1, pages 488–492 vol.1, Oct 1998.

[3] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. *ACM Transactions on Graphics*, 25(3):853–861, July 2006.

[4] Y. Ariki, S. Kubota, and M. Kumano. Automatic production system of soccer sports video by digital camera work based on situation recognition. In *Proceedings of the IEEE International Symposium on Multimedia*, pages 851–860, 2006.

[5] A. Au and J. Liang. Ztitch: A mobile phone application for immersive panorama creation, navigation, and social sharing. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, pages 13–18, Sept 2012.

[6] F. Boulos, W. Chen, B. Parrein, and P. Le Callet. Region-of-interest intra prediction for H.264/AVC error resilience. In *Proceedings of the IEEE International Conference on Image Processing*, pages 3109–3112, 2009.

[7] D. Bradley, A. Brunton, M. Fiala, and G. Roth. Image-based navigation in real environments using panoramas. In *Proceedings of the*

IEEE International Workshop on Haptic Audio Visual Environments and their Applications, pages 3 pp.–, Oct 2005.

[8] G. Bradski. Opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[9] W. Brendel, M. Amer, and S. Todorovic. Multiobject tracking as maximum weight independent set. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1273–1280, June 2011.

[10] J. Brosz and F. Samavati. Shape defined panoramas. In *Proceedings of the International Conference on Shape Modeling*, pages 57–67, June 2010.

[11] K. Brunnström, S. A. Beker, K. De Moor, A. Dooms, S. Egger, M.-N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, M.-C. Larabi, B. Lawlor, P. Le Callet, S. Möller, F. Pereira, M. Pereira, A. Perkis, J. Pibernik, A. Pinheiro, A. Raake, P. Reichl, U. Reiter, R. Schatz, P. Schelkens, L. Skorin-Kapov, D. Strohmeier, C. Timmerer, M. Varela, I. Wechsung, J. You, and A. Zgank. Qualinet White Paper on Definitions of Quality of Experience, Mar 2013. Qualinet White Paper on Definitions of Quality of Experience Output from the fifth Qualinet meeting, Novi Sad, March 12, 2013.

[12] K. Brunnstrom, D. Hands, F. Speranza, and A. Webster. VQEG validation and ITU standardization of objective perceptual video quality metrics. *IEEE Signal Processing Magazine*, 26(3):96–101, 2009.

[13] S. Brutzer, B. Hoferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1937–1944, 2011.

[14] Cairos technologies. VIS.TRACK, 2013. http://www.cairos.com/unternehmen/vistrack.php.

[15] Camargus. Premium Stadium Video Technology Inrastructure, 2013. http://www.camargus.com/.

[16] A. Carlier, V. Charvillat, W. T. Ooi, R. Grigoras, and G. Morin. Crowdsourced automatic zoom and scroll for video retargeting. In *Proceedings of the ACM International Conference on Multimedia*, pages 201–210, New York, NY, USA, 2010. ACM.

[17] P. Carr and R. Hartley. Portable multi-megapixel camera with real-time recording and playback. In *Proceedings of the Digital Image Computing Techniques and Applications*, pages 74–80, 2009.

[18] P. Carr, M. Mistry, and I. Matthews. Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording. In *Proceedings of the ACM International Conference on Multimedia*, pages 193–202, 2013.

[19] R. Carroll, M. Agrawal, and A. Agarwala. Optimizing content-preserving projections for wide-angle images. *ACM Transactions on Graphics*, 28(3):43:1–43:9, July 2009.

[20] E. Chang, S. Cheung, and D. Y. Pan. Color filter array recovery using a threshold-based variable number of gradients. In *Proceedings of SPIE, Sensors, Cameras, and Applications for Digital Photography*, volume 3650, pages 36–43, 1999.

[21] F. Chen and C. De Vleeschouwer. Personalized production of basketball videos from multi-sensored data under limited display resolution. *Computer Vision and Image Understanding*, 114(6):667–680, June 2010.

[22] K.-T. Chen, C.-C. Wu, Y.-C. Chang, and C.-L. Lei. A crowdsourceable QoE evaluation framework for multimedia content. *Proceedings of the ACM International Conference on Multimedia*, pages 491–500, 2009.

[23] D. Cok. Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal, Feb. 1987. US Patent 4,642,678.

[24] D. E. Comer, D. Gries, M. C. Mulder, A. Tucker, A. J. Turner, and P. R. Young. Computing as a discipline. *Communications of the ACM*, 32(1):9–23, Jan. 1989.

[25] D. Culibrk, O. Marques, D. Socek, H. Kalva, and B. Furht. A neural network approach to bayesian background modeling for video object segmentation. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, pages 474–479, 2006.

[26] L. D'Acunto, J. Redi, and O. Niamut. icacot-interactive camera-based coaching and training. In *Proceedings of the 3rd Workshop on Interactive Content Consumption*, 2015.

[27] S. Daigo and S. Ozawa. Automatic pan control system for broadcasting ball games based on audience's face direction. In *Proceedings of the ACM International Conference on Multimedia*, pages 444–447, 2004.

[28] T. De Pessemier, K. De Moor, I. Ketykó, W. Joseph, L. De Marez, and L. Martens. Investigating the influence of QoS on personal evaluation behaviour in a mobile context. *Multimedia Tools and Applications*, 57(2):335–358, Jan. 2012.

[29] A. Dearden, Y. Demiris, and O. Grau. Learning models of camera control for imitation in football matches. https://spiral.imperial.ac.uk:8443/handle/10044/1/12720.

[30] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the Annual Conference on Computer graphics and interactive techniques*, pages 369–378. ACM, 2008.

[31] P. Dizikes. Sports analytics: a real game-changer, 2013. http://web.mit.edu/newsoffice/2013/sloan-sports-analytics-conference-2013-0304.html.

[32] B. H. Do and S. C. Huang. Dynamic background modeling based on radial basis function neural networks for moving object detection. In

Proceedings of the IEEE International Conference on Multimedia and Expo, pages 1–4, July 2011.

[33] Dolphin Interconnect. Dolphin Interconnect PCI Express, 2013. http://www.dolphinics.no.

[34] R. Dosselmann and X. Yang. A comprehensive assessment of the structural similarity index. *Signal, Image and Video Processing*, 5(1):81–91, 2011.

[35] A. Eden, M. Uyttendaele, and R. Szeliski. Seamless image stitching of scenes with large motions and exposure differences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2498–2505, 2006.

[36] M. C. Q. Farias, J. M. Foley, and S. K. Mitra. Detectability and annoyance of synthetic blocky, blurry, noisy, and ringing artifacts. *IEEE Transactions on Signal Processing*, 55(6):2954–2964, June 2007.

[37] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[38] E. Foote, P. Carr, P. Lucey, Y. Sheikh, and I. Matthews. One-man-band: A touch screen interface for producing live multi-camera sports broadcasts. In *Proceedings of the ACM International Conference on Multimedia*, pages 163–172, 2013.

[39] J. Foote and D. Kimber. Flycam: practical panoramic video and automatic camera control. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume 3, pages 1419–1422 vol.3, 2000.

[40] V. R. Gaddam, R. Eg, R. Langseth, C. Griwodz, and P. Halvorsen. The cameraman operating my virtual camera is artificial: Can the machine be as good as a human? *ACM Transactions on Multimedia Computing, Communications, and Applications*, 11(4):56:1–56:20, June 2015.

[41] V. R. Gaddam, C. Griwodz, and P. Halvorsen. Automatic exposure for panoramic systems in uncontrolled lighting conditions: a football stadium case study. In *Proceedings of SPIE The Engineering Reality of Virtual Reality*, volume 9012, pages 90120C–90120C–9, 2014.

[42] V. R. Gaddam, R. Langseth, S. Ljødal, P. Gurdjos, V. Charvillat, C. Griwodz, and P. Halvorsen. Interactive zoom and panning from live panoramic video. In *Proceedings of the ACM International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 19:19–19:24, 2014.

[43] V. R. Gaddam, R. Langseth, H. K. Stensland, C. Griwodz, and P. Halvorsen. Automatic real-time zooming and panning on salient objects from a panoramic video. In *Proceedings of the ACM International Conference on Multimedia*, pages 725–726. ACM, November 2014.

[44] V. R. Gaddam, R. Langseth, H. K. Stensland, C. Griwodz, P. Halvorsen, and D. Johansen. Scaling virtual camera services to a large number of users. In *Proceedings of the ACM International Conference on Multimedia Systems*, pages 93–96, New York, NY, USA, 2015. ACM.

[45] V. R. Gaddam, R. Langseth, H. K. Stensland, P. Gurdjos, V. Charvillat, C. Griwodz, D. Johansen, and P. Halvorsen. Be your own cameraman: Real-time support for zooming and panning into stored and live panoramic video. In *Proceedings of the ACM Conference on Multimedia Systems*, pages 168–171. ACM, March 2014.

[46] V. R. Gaddam, H. B. Ngo, R. Langseth, C. Griwodz, D. Johansen, and P. Halvorsen. Tiling of panorama video for interactive virtual cameras: Overheads and potential bandwidth requirement reduction. In *Proceedings of the IEEE Picture Coding Symposium*, pages 204–209, May 2015.

[47] V. R. Gaddam, M. Riegler, R. Eg, C. Griwodz, and P. Halvorsen. [in review]tiling in interactive panoramic video: Approaches and evaluation. *IEEE Transactions on Multimedia*.

[48] J. Gao, S. J. Kim, and M. Brown. Constructing image panoramas using dual-homography warping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–56, June 2011.

[49] L. Goldmann, F. D. Simone, F. Dufaux, T. Ebrahimi, R. Tanner, and M. Lattuada. Impact of video transcoding artifacts on the subjective quality. In *Proceedings of the International Conference on Quality of Multimedia Experience*, pages 52–57, Trondheim, 2010.

[50] C. Grunheit, A. Smolic, and T. Wiegand. Efficient representation and interactive streaming of high-resolution panoramic views. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages III–209–III–212 vol.3, 2002.

[51] R. Guntur and W. T. Ooi. On tile assignment for region-of-interest video streaming in a wireless lan. In *Proceedings of the ACM International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 59–64, 2012.

[52] P. Halvorsen, S. Sægrov, A. Mortensen, D. K. Kristensen, A. Eichhorn, M. Stenhaug, S. Dahl, H. K. Stensland, V. R. Gaddam, C. Griwodz, and D. Johansen. Bagadus: An integrated system for arena sports analytics a soccer case study. In *Proceedings of the ACM International Conference on Multimedia Systems*, pages 48–59, Mar. 2013.

[53] M. Harris. Optimizing parallel reduction in cuda. *NVIDIA Developer Technology*, 2:45, 2007.

[54] C. Hermans, C. Vanaken, T. Mertens, F. Van Reeth, and P. Bekaert. Augmented panoramic video. *Computer Graphics Forum*, 27(2):281–290, 2008.

[55] S. Heymann, A. Smolic, K. Mueller, Y. Guo, J. Rurainsky, P. Eisert, and T. Wiegand. Representation, coding and interactive rendering of high-resolution panoramic images and video using mpeg-4. In *Proceedings of Panoramic Photogrammetry Workshop*, 2005.

[56] K. Hirakawa, S. Member, and T. W. Parks. Adaptive homogeneity-directed demosaicing algorithm. *IEEE Transactions on Image Processing*, 14:360–369, 2005.

[57] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar. RTP Payload Format for MPEG1/MPEG2 Video. RFC 2250 (Proposed Standard), Jan. 1998.

[58] T. Hoßfeld, M. Hirth, J. Redi, F. Mazza, P. Korshunov, B. Naderi, M. Seufert, B. Gardlo, S. Egger, and C. Keimel. Best practices and recommendations for crowdsourced qoe - lessons learned from the qualinet task force crowdsourcing, Oct 2014.

[59] T. Hoßfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia. Best practices for qoe crowdtesting: Qoe assessment with crowdsourcing. *IEEE Transactions on Multimedia*, 16(2):541–558, Feb 2014.

[60] T. Hoßfeld, C. Keimel, and C. Timmerer. Crowdsourcing quality-of-experience assessments. *Computer*, 47(9):98–102, Sept 2014.

[61] K. A. Hua, Y. Cai, and S. Sheu. Patching: a multicast technique for true video-on-demand services. In *Proceedings of the ACM International Conference on Multimedia*, pages 191–200, 1998.

[62] S. C. Huang. An advanced motion detection algorithm with video quality analysis for video surveillance systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(1):1–14, Jan 2011.

[63] S. C. Huang and B. H. Chen. Highly accurate moving object detection in variable bit rate video-based traffic monitoring systems. *IEEE*

Transactions on Neural Networks and Learning Systems, 24(12):1920–1931, Dec 2013.

[64] C. Hughes, M. Glavin, E. Jones, and P. Denny. Review of geometric distortion compensation in fish-eye cameras. In *Proceedings of the Irish Signals and Systems Conference*, pages 162–167, June 2008.

[65] Q. Huynh-Thu and M. Ghanbari. Scope of validity of PSNR in image/video quality assessment. *Electronics Letters*, 44(13):800–801, 2008.

[66] M. Ibrahim, R. Hafiz, M. Khan, Y. Cho, and J. Cha. Automatic reference selection for parametric color correction schemes for panoramic video stitching. In *Advances in Visual Computing*, volume 7431 of *Lecture Notes in Computer Science*, pages 492–501. 2012.

[67] Indumos. Indumos, 2013. http://www.indumos.su/images/upload/en/71/ptz140-camera-ptz.jpg.

[68] Interplay sports. The ultimate video analysis and scouting software, 2013. http://www.interplay-sports.com/.

[69] ITU-R. BT.500-11. Methodology for the subjective assessment of the quality of television pictures, 2002.

[70] ITU-T. P.911. Subjective audiovisual quality assessment methods for multimedia applications, 1998.

[71] ITU-T. J.247: Objective perceptual multimedia video quality measurement in the presence of a full reference, 2008.

[72] M. Jogan and A. Leonardis. Robust localization using panoramic view-based recognition. In *Proceedings of the IEEE International Conference on Pattern Recognition*, volume 4, pages 136–139 vol.4, 2000.

[73] D. Johansen, M. Stenhaug, R. B. A. Hansen, A. Christensen, and P.-M. Høgmo. Muithu: Smaller footprint, potentially larger imprint. In *Pro-*

ceedings of the IEEE International Conference on Digital Information Management, pages 205–214, Aug. 2012.

[74] P. Kaewtrakulpong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection. In *Proceedings of Workshop on Advanced Video Based Surveillance Systems*, pages 135–144, 2001.

[75] M. Karczewicz and R. Kurceren. The SP- and SI-frames design for H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):637–644, 2003.

[76] L. Kellerer, V. R. Gaddam, R. Langseth, H. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen. Real-time hdr panorama video. In *Proceedings of the ACM International Conference on Multimedia*, pages 1205–1208, New York, NY, USA, 2014. ACM.

[77] H. Kimata, M. Isogai, H. Noto, M. Inoue, K. Fukazawa, and N. Matsuura. Interactive panorama video distribution system. In *Proceedings of the IEEE Technical Symposium at ITU Telecom World*, pages 45–50, Oct 2011.

[78] H. Kimata, D. Ochi, A. Kameda, H. Noto, K. Fukazawa, and A. Kojima. Mobile and multi-device interactive panorama video distribution system. In *Proceedings of the Global Conference on Consumer Electronics*, pages 574–578, Oct 2012.

[79] R. Kimmel. Demosaicing: image reconstruction from color ccd samples. *IEEE Transactions on Image Processing*, 8(9):1221–1228, Sep 1999.

[80] A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 453–456, 2008.

[81] C. Kreuzberger, D. Posch, and H. Hellwagner. A scalable video coding dataset and toolchain for dynamic adaptive streaming over HTTP.

In *Proceedings of the ACM International Conference on Multimedia Systems*, pages 213–218, 2015.

[82] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, pages 79–86, 1951.

[83] R. Langseth, V. Gaddam, H. Stensland, C. Griwodz, and P. Halvorsen. An evaluation of debayering algorithms on gpu for real-time panoramic video recording. In *Proceedings of the IEEE International Symposium on Multimedia*, pages 110–115, Dec 2014.

[84] R. Langseth, V. R. Gaddam, H. K. Stensland, C. Griwodz, P. Halvorsen, and D. Johansen. An experimental evaluation of debayering algorithms on gpus for recording panoramic video in real-time. *International Journal of Multimedia Data Engineering and Management*, 6:1–16, 07/2015 2015.

[85] G. W. Larson, H. Rushmeier, and C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):291–306, 1997.

[86] J.-S. Lee, L. Goldmann, and T. Ebrahimi. Paired comparison-based subjective quality assessment of stereoscopic images. *Multimedia Tools and Applications*, 67(1):31–48, 2012.

[87] Z. Li, A. C. Begen, J. Gahm, Y. Shan, B. Osler, and D. Oran. Streaming video over HTTP with consistent quality. In *Proceedings of the ACM International Conference on Multimedia Systems*, pages 248–258, 2014.

[88] W.-Y. Lin, S. Liu, Y. Matsushita, T.-T. Ng, and L.-F. Cheong. Smoothly varying affine stitching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 345–352, June 2011.

[89] A. Lipton, H. Fujiyoshi, and R. Patil. Moving target classification and tracking from real-time video. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pages 8–14, Oct 1998.

[90] F. Liu and W. T. Ooi. Zoomable video playback on mobile devices by selective decoding. In *Proceedings of the Pacific-Rim Conference on Multimedia*, 2012.

[91] W. Lu and Y.-P. Tan. Color filter array demosaicking: new method and performance measures. *IEEE Transactions on Image Processing*, 12(10):1194–1210, Oct 2003.

[92] L. Maddalena and A. Petrosino. A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Transactions on Image Processing*, 17(7):1168–1177, July 2008.

[93] E. Malis, F. Chaumette, and S. Boudet. 2d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, Apr 1999.

[94] H. S. Malvar, L. wei He, and R. Cutler. High-quality linear interpolation for demosaicing of bayer-patterned color images. In *Proceedings of the IEEE International Conference on Speech, Acoustics, and Signal Processing*, 2004.

[95] J. L. Mannos and D. J. Sakrison. The effects of a visual fidelity criterion of the encoding of images. *IEEE Transactions on Information Theory*, 20(4):525 – 536, 1974.

[96] A. Manzanera and J. C. Richefeu. A new motion detection algorithm based on $\Sigma - \Delta$ background estimation. *Pattern Recognition Letters*, 28(3):320–328, Feb. 2007.

[97] A. Mavlankar and B. Girod. Pre-fetching based on video analysis for interactive region-of-interest streaming of soccer sequences. In *Proceedings of the IEEE International Conference on Image Processing*, pages 3061–3064, Nov 2009.

[98] A. Mavlankar and B. Girod. Video streaming with interactive pan/tilt/zoom. In M. Mrak, M. Grgic, and M. Kunt, editors, *High-Quality Visual Experience*, Signals and Communication Technology, pages 431–455. 2010.

[99] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. *ACM Computer Communication Review*, 26:117–130, 1996.

[100] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, pages 39–46, New York, NY, USA, 1995. ACM.

[101] D. Menon, S. Andriani, and G. Calvagno. Demosaicing with directional filtering and a posteriori decision. *IEEE Transactions on Image Processing*, 16(1):132–141, 2007.

[102] P. Migliorati, F. Pedersini, L. Sorcinelli, and S. Tubaro. Semantic segmentation applied to image interpolation in the case of camera panning and zooming. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 25–28 vol.5, April 1993.

[103] A. Mortensen, V. R. Gaddam, H. K. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen. Automatic event extraction and video summaries from soccer games. In *Proceedings of the ACM International Conference on Multimedia Systems*, pages 176–179. ACM, March 2014.

[104] K. Q. M. Ngo, G. Ravindra, A. Carlier, and W. T. Ooi. Supporting zoomable video streams with dynamic region-of-interest cropping. In *Proceedings of the ACM International Conference on Multimedia Systems*, page 259, New York, New York, USA, 2010. ACM Press.

[105] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. l. Halvorsen. Flicker effects in adaptive video streaming to handheld devices. In *Proceedings of the ACM International Conference on Multimedia*, pages 463–472, Scottsdale, 2011.

[106] O. Niamut, J. Macq, M. Prins, R. Van Brandenburg, N. Verzijp, and P. Alface. Towards scalable and interactive delivery of immersive media. In *Proceedings of the New European Media Summit*, pages 69–74, 2012.

[107] NRK. Hogmo mapping players with video surveillance, 2015. http://www.nrk.no/sport/fotball/hogmo-kartlegger-spillerne-med-videoovervaking-1.12018111.

[108] NVIDIA. NVIDIA - NVIDIA hardware video encoder. `http://developer.download.nvidia.com/compute/nvenc/v4.0/NVENC_AppNote.pdf`, 2014.

[109] NVIDIA. NVIDIA - cuFFT. `https://developer.nvidia.com/cufft`, 2015.

[110] Ookla. Household download index. http://www.netindex.com/download/, 2014.

[111] J. Owens. In J. Owens, editor, *Television Sports Production (Fourth Edition)*. Focal Press, Boston, fourth edition edition, 2006.

[112] S. Pechard, M. Carnec, P. Le Callet, and D. Barba. From SD to HD television: Effects of H.264 distortions versus display size on quality of experience. In *Proceedings of the IEEE International Conference on Image Processing*, pages 409–412, 2006.

[113] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 338–343, Jun 1997.

[114] S. A. Pettersen, D. Johansen, H. Johansen, V. Berg-Johansen, V. R. Gaddam, A. Mortensen, R. Langseth, C. Griwodz, H. K. Stensland, and P. Halvorsen. Soccer video and player position dataset. In *Proceedings of the ACM International Conference on Multimedia Systems*, pages 18–23, New York, NY, USA, 2014. ACM.

[115] Y. Pitrey, U. Engelke, M. Barkowsky, R. Pépion, and P. L. Callet. Aligning subjective tests using a low cost common set. In *Proceedings of Conference on Quality of Experience for Multimedia Content Sharing*, 2011.

[116] Prozone. Prozone Sports – Introducing Prozone Performance Analysis Products, 2013. http://www.prozonesports.com/products.html.

[117] Z. Qi and J. R. Cooperstock. Overcoming parallax and sampling density issues in image mosaicing of non-planar scenes. In *Proceedings of the British Machine Vision Conference*, September 2007.

[118] K. Raaen, R. Eg, and C. Griwodz. Can gamers detect cloud delay? In *Proceedings of the International Workshop on Network and Systems Support for Games*, pages 1–3, 2014.

[119] J. Redi, L. D'Acunto, and O. Niamut. Interactive uhdtv at the commonwealth games: An explorative evaluation. In *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video*, pages 43–52, 2015.

[120] J. A. Redi, Y. Zhu, H. de Ridder, and I. Heynderickx. *Visual Signal Quality Assessment: Quality of Experience (QoE)*, chapter How Passive Image Viewers Became Active Multimedia Users, pages 31–72. 2015.

[121] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. *ACM Transactions on Graphics*, 21(3):267–276, 2002.

[122] F. Ribeiro, D. Florencio, and V. Nascimento. Crowdsourcing subjective image quality evaluation. In *Proceedings of the IEEE International Conference on Image Processing*, pages 3097–3100, 2011.

[123] M. Riegler, V. R. Gaddam, M. Larson, P. Halvorsen, and C. Griwodz. Crowdsourcing as self fulfilling prophecy: Influence of discarding work-

ers in subjective assessment tasks. In *Proceedings of 14th International Workshop on Content-based Multimedia Indexing*, 2016.

[124] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen. Video streaming using a location-based bandwidth-lookup service for bitrate planning. *ACM Transactions on Multimedia Computing, Communications and Applications*, 8(3), 2011.

[125] H. Riiser, P. Halvorsen, C. Griwodz, and D. Johansen. Low overhead container format for adaptive streaming. In *Proceedings of the ACM International Conference on Multimedia Systems*, pages 193–198, 2010.

[126] M. A. Robertson, S. Borman, and R. L. Stevenson. Estimation-theoretic approach to dynamic range enhancement using multiple exposures. *Journal of Electronic Imaging*, 12(2):219–228, 2003.

[127] J. Rosenberg and H. Schulzrinne. An offer/answer model with session description protocol (SDP). RFC 3264 (Proposed Standard), June 2002.

[128] S. Sægrov, A. Eichhorn, J. Emerslund, H. K. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen. Bagadus: An integrated system for soccer analysis (demo). In *Proceedings of the the International Conference on Distributed Smart Cameras*, Oct. 2012.

[129] V. D. Salvo, A. Collins, B. McNeill, and M. Cardinale. Validation of Prozone: A new video-based performance analysis system. *International Journal of Performance Analysis in Sport*, 6(1):108–119, June 2006.

[130] O. H. Schmitt. A thermionic trigger. *Journal of Scientific Instruments*, 15(1):24, 1938.

[131] I. Sevcenco, P. Hampton, and P. Agathoklis. Seamless stitching of images based on a haar wavelet 2d integration method. In *Proceedings of the IEEE International Conference on Digital Signal Processing*, pages 1–6, July 2011.

[132] W. R. Shadish, T. D. Cook, and D. T. Campbell. Statistical conclusion validity and internal validity. In *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*, chapter 2, pages 33–63. Cengage Learning, 2 edition, 2002.

[133] J. Sim and C. C. Wright. The kappa statistic in reliability studies: use, interpretation, and sample size requirements. *Physical therapy*, 85(3):257–268, 2005.

[134] Simula. Bagadus repository. `https://bitbucket.org/mpgs/bagadus`, 2014.

[135] Stats Technology. STATS — SportVU — Football/Soccer, 2013. http://www.sportvu.com/football.asp.

[136] H. K. Stensland, H. Espeland, C. Griwodz, and P. Halvorsen. Tips, tricks and troubles: Optimizing for cell and gpu. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 75–80, New York, NY, USA, 2010. ACM.

[137] H. K. Stensland, V. R. Gaddam, M. Tennøe, E. Helgedagsrud, M. Næss, H. K. Alstad, C. Griwodz, and P. Halvorsen. Processing panorama video in real-time. *International Journal of Semantic Computing*, 08:209–227, 2014.

[138] H. K. Stensland, V. R. Gaddam, M. Tennøe, E. Helgedagsrud, M. Næss, H. K. Alstad, A. Mortensen, R. Langseth, S. Ljødal, Ø. Landsverk, C. Griwodz, P. Halvorsen, M. Stenhaug, and D. Johansen. Bagadus: An integrated real-time system for soccer analytics. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2014.

[139] H. K. Stensland, M. A. Wilhelmsen, V. R. Gaddam, A. Mortensen, R. Langseth, C. Griwodz, and P. Halvorsen. Using a commodity hardware video encoder for interactive applications. *International Journal of Multimedia Data Engineering and Management (IJMDEM)*, 6:17–31, 07/2015 2015.

[140] X. Sun, J. Foote, D. Kimber, and B. Manjunath. Region of interest extraction and virtual camera control based on panoramic video capturing. *IEEE Transactions on Multimedia*, 7(5):981–990, 2005.

[141] R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, pages 251–258. ACM, 1997.

[142] W.-K. Tang, T.-T. Wong, and P.-A. Heng. A system for real-time panorama generation and display in tele-immersive applications. *IEEE Transactions on Multimedia*, 7(2):280–292, April 2005.

[143] M. Tennøe, E. O. Helgedagsrud, M. Næss, H. K. Alstad, H. K. Stensland, V. R. Gaddam, D. Johansen, C. Griwodz, and P. Halvorsen. Efficient implementation and processing of a real-time panorama video pipeline. In *Proceedings of the IEEE International Symposium on Multimedia*, pages 76–83, 2013.

[144] M. D. Tocci, C. Kiser, N. Tocci, and P. Sen. A versatile HDR video production system. *ACM Transactions on Graphics*, 30(4):41, 2011.

[145] I. Unanue, I. Urteaga, R. Husemann, J. D. Ser, V. Roesler, A. Rodriguez, and P. Sanchez. A tutorial on H.264/SVC scalable video coding and its tradeoff between quality, coding efficiency and performance. In *Recent Advances on Video Coding*, pages 3–26. Intech, 2011.

[146] H. Wang, V.-T. Nguyen, W. T. Ooi, and M. C. Chan. Mixing tile resolutions in tiled video: A perceptual quality assessment. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 25:25–25:30, 2014.

[147] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[148] Z. Wang, L. Lu, and A. C. Bovik. Video quality assessment based on structural distortion measurement. *Signal Processing: Image Communication*, 19(2):121–132, Feb. 2004.

[149] G. Ward. A contrast-based scalefactor for luminance display. *Graphics gems IV*, pages 415–421, 1994.

[150] S. Wenger, U. Chandra, M. Westerlund, and B. Burman. Codec control messages in the RTP audio-visual profile with feedback (AVPF). RFC 5104 (Proposed Standard), Feb. 2008.

[151] M. A. Wilhelmsen, H. K. Stensland, V. R. Gaddam, A. Mortensen, R. Langseth, C. Griwodz, and P. Halvorsen. Using a commodity hardware video encoder for interactive video streaming. In *Proceedings of the IEEE International Symposium on Multimedia*, 2014.

[152] S. Winkler. *Digital Video Quality: Vision Models and Metrics*. Wiley, 2005.

[153] L. Wolf, M. Guttmann, and D. Cohen-Or. Non-homogeneous content-driven video-retargeting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–6, Oct 2007.

[154] W. Wu, A. Arefin, R. Rivas, K. Nahrstedt, R. M. Sheppard, and Z. Yang. Quality of experience in distributed interactive multimedia environments: Toward a theoretical framework. In *Proceedings of the ACM International Conference on Multimedia*, pages 481–490, Beijing, 2009.

[155] Y. Xiong and K. Pulli. Color correction for mobile panorama imaging. In *Proceedings of the ACM International Conference on Internet Multimedia Computing and Service*, pages 219–226, 2009.

[156] W. Xu and J. Mulligan. Performance evaluation of color correction approaches for automatic multi-view image and video stitching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 263–270, 2010.

[157] W. Xu and J. Mulligan. Panoramic video stitching from commodity hdtv cameras. *Multimedia Systems*, 19(5):407–426, 2013.

[158] T. Yokoi and H. Fujiyoshi. Virtual camerawork for generating lecture video from high resolution images. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, July 2005.

[159] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the IEEE International Conference on Pattern Recognition*, pages 28 – 31 Vol.2, aug. 2004.

[160] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773 – 780, 2006.

[161] D. Zorin and A. H. Barr. Correction of geometric perceptual distortions in pictures. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, pages 257–264, 1995.

[162] ZXY. ZXY Sport Tracking, 2013. http://www.zxy.no/.

Appendix A

Publications

Four journal articles, three peer-reviewed conference publications and three technical demos are included in the thesis. This appendix contains abstracts of each publication, as well as the contribution of the author of this thesis.

A.1 Journal Articles

A.1.1

Title: Bagadus: an Integrated Real-Time System for Soccer Analytics

Authors: H. K. Stensland, **V. R. Gaddam**, M. Tennøe, E. O. Helgedagsrud, M. Næss, H. K. Alstad, A. Mortensen, R. Langseth, S. Ljødal, Ø. Landsverk, C. Griwodz, P. Halvorsen, M. Stenhaug and D. Johansen

Published: ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), 2014

Abstract: The importance of winning has increased the role of performance analysis in the sports industry, and this underscores how statistics and technology keep changing the way sports are played. Thus, this is a growing area of interest, both from a computer system view in managing the technical challenges and from a sport performance view in aiding the development of athletes. In this respect, Bagadus is a real-time prototype of a sports analytics application using soccer as a case study. Bagadus integrates a sensor system, a soccer analytics annota-

tions system, and a video processing system using a video camera array. A prototype is currently installed at Alfheim Stadium in Norway, and in this article, we describe how the system can be used in real-time to playback events. The system supports both stitched panorama video and camera switching modes and creates video summaries based on queries to the sensor system. Moreover, we evaluate the system from a systems point of view, benchmarking different approaches, algorithms, and trade-offs, and show how the system runs in real time.

Contributions: The author has contributed in developing the image processing algorithms in the panorama pipeline. He also actively participated in the design of various components of the pipeline. The author also contibuted to the paper writing process.

A.1.2

Title: Processing Panorama Video in Real-Time

Authors: H. K. Stensland, **V. R. Gaddam**, M. Tennøe, E. O. Helgedagsrud, M. Næss, H. K. Alstad, C. Griwodz, P. Halvorsen, and D. Johansen

Published: International Journal of Semantic Computing (IJSC), 2014

Abstract: There are many scenarios where high resolution, wide field-of-view video is useful. Such panorama video may be generated using camera arrays where the feeds from multiple cameras pointing at different parts of the captured area are stitched together. However, processing the dierent steps of a panorama video pipeline in real-time is challenging due to the high data rates and the stringent timeliness requirements. In our research, we use panorama video in a sport analysis system called Bagadus. This system is deployed at Alfheim stadium in Tromsø, and due to live usage, the video events must be generated in real-time. In this paper, we describe our realtime panorama system built using a low-cost CCD HD video camera array. We describe how we have implemented different components and evaluated alternatives. The performance results from experiments ran on commodity hardware with and without co-processors like graphics processing units (GPUs) show that the entire pipeline is able to run in real-time.

Contributions: The author has contributed in developing the image processing algorithms in the panorama pipeline. He also actively participated in the design of various components of the pipeline. The author also contibuted to the paper writing process.

A.1.3

Title: The Cameraman Operating My Virtual Camera Is Artificial: Can The Machine Be As Good As A Human?

Authors: **V. R. Gaddam**, R. Eg, C. Griwodz, and P. Halvorsen

Published: ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP), 2015

Abstract: In this article, we argue that the energy spent in designing autonomous camera control systems is not spent in vain. We present a real-time virtual camera system that can create smooth camera motion. Similar systems are frequently benchmarked with the human operator as the best possible reference; however, we avoid a priori assumptions in our evaluations. Our main question is simply whether we can design algorithms to steer a virtual camera that can compete with the user experience for recordings from an expert operator with several years of experience? In this respect, we present two low-complexity servoing methods that are explored in two user studies. The results from the user studies give a promising answer to the question pursued. Furthermore, all components of the system meet the real-time requirements on commodity hardware. The growing capabilities of both hardware and network in mobile devices give us hope that this system can be deployed to mobile users in the near future. Moreover, the design of the presented system takes into account that services to concurrent users must be supported.

Contributions: The author has designed and implemented the algorithms. He also implemented the user-studies. He contributed to the writing process.

A.1.4

Title: [In Review]Tiling in Interactive Panoramic Video: Approaches and Evaluation

Authors: **V. R. Gaddam**, M. Riegler, R. Eg, C. Griwodz, and P. Halvorsen

In Review: IEEE Transactions on Multimedia (T-MM), 2016

Abstract: Interactive panoramic systems are currently on the rise. However, one of the major challenges involved in such a system is the overhead to transfer a full quality panorama to the client where only a part of the panorama is used to extract a virtual view. Thus, a system should maximize the user experience and at the same time minimize the bandwidth required. In this paper, we apply tiling to deliver different qualities of different parts of the panorama. Tiling has traditionally been applied to delivery of very high-resolution content to clients, and here, we apply similar ideas in a real-time interactive panoramic video system. A major challenge is movement of such a virtual view, where clients regions of interest change dynamically and independently from each other. We show that our algorithms, which progressively increases quality towards the point of the view, manages to (i) reduce the bandwidth requirement and (ii) provide a similar QoE compared to a full panorama system.

Contributions: The author has designed and implemented the algorithms. He also implemented the user-studies. He contributed to the writing process.

A.2 Conference Publications

A.2.1

Title: Automatic Exposure for Panoramic Systems in Uncontrolled Lighting Conditions: a Football Stadium Case Study

Authors: **V. R. Gaddam**, C. Griwodz, and P. Halvorsen

Published: SPIE Electronic Imaging (EI), 2014.

Abstract: One of the most common ways of capturing wide field-of-view scenes is by recording panoramic videos. Using an array of cameras with limited overlapping in the corresponding images, one can generate good panorama images. Using the panorama, several immersive display options can be explored. There is a two fold synchronization problem associated to such a system. One is the temporal synchronization, but this challenge can easily be handled by using a common triggering solution to control the shutters of the cameras. The other synchronization challenge is the automatic exposure synchronization which does not have a straight forward solution, especially in a wide area scenario where the light conditions are uncontrolled like in the case of an open, outdoor football stadium.

In this paper, we present the challenges and approaches for creating a completely automatic real-time panoramic capture system with a particular focus on the camera settings. One of the main challenges in building such a system is that there is not one common area of the pitch that is visible to all the cameras that can be used for metering the light in order to find appropriate camera parameters. One approach we tested is to use the green color of the field grass. Such an approach provided us with acceptable results only in limited light conditions.A second approach was devised where the overlapping areas between adjacent cameras are exploited, thus creating pairs of perfectly matched video streams. However, there still existed some disparity between different pairs. We finally developed an approach where the time between two temporal frames is exploited to communicate the exposures among the cameras where we achieve a perfectly synchronized array. An analysis of the system and some experimental results are presented in this paper. In summary, a pilot-camera approach running in auto-exposure mode and then distributing the used exposure values to the other cameras seems to give best visual results.

Contributions: The author has designed and implemented the algorithms into the system. He contributed to the writing process.

A.2.2

Title: Interactive Zoom and Panning From Live Panoramic Video

Authors: **V. R. Gaddam**, R. Langseth, S. Ljødal, P. Gurdjos, V. Charvillat, C. Griwodz, and P. Halvorsen.

Published: ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 2014.

Abstract: Panorama video is becoming increasingly popular, and we present an end-to-end real-time system to interactively zoom and pan into high-resolution panoramic videos. Compared to existing systems using perspective panoramas with cropping, our approach creates a cylindrical panorama. Here, the perspective is corrected in real-time, and the result is a better and more natural zoom. Our experimental results also indicate that such zoomed virtual views can be generated far below the frame-rate threshold. Taking into account recent trends in device development, our approach should be able to scale to a large number of concurrent users in the near future.

Contributions: The author has desgined and implemented the virtual viewer. He has also contributed to the writing process.

A.2.3

Title: Tiling of Panorama Video for Interactive Virtual Cameras: Overheads and Potential Bandwidth Requirement Reduction

Authors: **V. R. Gaddam**, H. B. Ngo, R. Langseth, C. Griwodz, D. Johansen, and P.Halvorsen

Published: IEEE Picture Coding Symposium (PCS), 2015.

Abstract: Delivering high resolution, high bitrate panorama video to a large number of users introduces huge scaling challenges. To reduce the resource requirement, researchers have earlier proposed tiling in order to deliver different qualities in different spatial parts of the video. In our work, providing an interactive moving virtual camera to each user, tiling may be used to reduce the quality depending on the position of the virtual view. This raises new challenges compared to existing tiling

approaches as the need for high quality tiles dynamically change. In this paper, we describe a tiling approach of panorama video for interactive virtual cameras where we provide initial results showing the introduced overheads and the potential reduction in bandwidth requirement.

Contributions: The author has designed and implemented the tiling in the virtual viewer. He also contributed to the writing process.

A.3 Technical Demos

A.3.1

Title: Demo: Be Your Own Cameraman: Real-Time Support for Zooming and Panning Into Stored and Live Panoramic Video

Authors: **V. R. Gaddam**, R. Langseth, H. K. Stensland, P. Gurdjos, V. Charvillat, C. Griwodz, D. Johansen, and P. Halvorsen

Published: ACM International Conference on Multimedia Systems (MMSys), 2014.

Abstract: High-resolution panoramic video with a wide field-of-view is popular in many contexts. However, in many examples, like surveillance and sports, it is often desirable to zoom and pan into the generated video. A challenge in this respect is real-time support, but in this demo, we present an end-to-end real-time panorama system with interactive zoom and panning. Our system installed at Alfheim stadium, a Norwegian premier league soccer team, generates a cylindrical panorama from five 2K cameras live where the perspective is corrected in real-time when presented to the client. This gives a better and more natural zoom compared to existing systems using perspective panoramas and zoom operations using plain crop. Our experimental results indicate that virtual views can be generated far below the frame-rate threshold, i.e., on a GPU, the processing requirement per frame is about 10 milliseconds.

The proposed demo lets participants interactively zoom and pan into stored panorama videos generated at Alfheim stadium and from a live 2-camera array on-site.

Contributions: The author has implemented the virtual viewer. He also contributed to the writing process.

A.3.2

Title: Demo: Automatic Real-Time Zooming and Panning on Salient Objects From a Panoramic Video

Authors: **V. R. Gaddam**, R. Langseth, H. K. Stensland, C. Griwodz, and P. Halvorsen

Published: ACM International Conference on Multimedia (MM), 2014.

Abstract: The proposed demo shows how our system automatically zooms and pans into tracked objects in panorama videos. At the conference site, we will set up a two-camera version of the system, generating live panorama videos, where the system zooms and pans tracking people using colored hats. Additionally, using a stored soccer game video from a five 2K camera setup at Alfheim stadium in Tromsø from the European league game between Tromsø IL and Tottenham Hotspurs, the system automatically follows the ball.

Contributions: The author has implemented the algorithms for controlling the virtual camera. He also contributed to the writing process.

A.3.3

Title: Demo: Scaling Virtual Camera Services to a Large Number of Users

Authors: **V. R. Gaddam**, R. Langseth, H. K. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen

Published: ACM International Conference on Multimedia Systems (MMSys), 2015.

Abstract: By processing video footage from a camera array, one can easily make wide-field-of-view panorama videos. From the single panorama video, one can further generate multiple virtual cameras supporting personalized views to a large number of users based on only the few physical cameras in the array. However, giving personalized services to large numbers of users potentially introduces both bandwidth and

processing bottlenecks, depending on where the virtual camera is processed.

In this demonstration, we present a system that address the large cost of transmitting entire panorama video to the end-user where the user creates the virtual views on the client device. Our approach is to divide the panorama into tiles, each encoded in multiple qualities. Then, the panorama video tiles are retrieved by the client in a quality (and thus bit rate) depending on where the virtual camera is pointing, i.e., the video quality of the tile changes dynamically according to the user interaction. Our initial experiments indicate that there is a large potential of saving bandwidth on the cost of trading quality of in areas of the panorama frame not used for the extraction of the virtual view.

Contributions: The author has implemented tiling into the virtual viewer. He also contributed to the writing process.

Appendix B

[Journal] Bagadus: An Integrated Real-Time System for Soccer Analytics

[Authors:] H. K. Stensland, **V. R. Gaddam**, M. Tennøe, E. O. Helgedagsrud, M. Næss, H. K. Alstad, A. Mortensen, R. Langseth, S. Ljødal, Ø. Landsverk, C. Griwodz, P. Halvorsen, M. Stenhaug and D. Johansen
[Published:] ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), 2014

Bagadus: An Integrated Real-Time System for Soccer Analytics

HÅKON KVALE STENSLAND, VAMSIDHAR REDDY GADDAM, MARIUS TENNØE, ESPEN HELGEDAGSRUD, MIKKEL NÆSS, HENRIK KJUS ALSTAD, ASGEIR MORTENSEN, RAGNAR LANGSETH, SIGURD LJØDAL, ØYSTEIN LANDSVERK, CARSTEN GRIWODZ, and PÅL HALVORSEN, University of Oslo and Simula Research Laboratory
MAGNUS STENHAUG and DAG JOHANSEN, University of Tromsø

The importance of winning has increased the role of performance analysis in the sports industry, and this underscores how statistics and technology keep changing the way sports are played. Thus, this is a growing area of interest, both from a computer system view in managing the technical challenges and from a sport performance view in aiding the development of athletes. In this respect, Bagadus is a real-time prototype of a sports analytics application using soccer as a case study. Bagadus integrates a sensor system, a soccer analytics annotations system, and a video processing system using a video camera array. A prototype is currently installed at Alfheim Stadium in Norway, and in this article, we describe how the system can be used in real-time to playback events. The system supports both stitched panorama video and camera switching modes and creates video summaries based on queries to the sensor system. Moreover, we evaluate the system from a systems point of view, benchmarking different approaches, algorithms, and trade-offs, and show how the system runs in real time.

Categories and Subject Descriptors: H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Video*

General Terms: Experimentation, Measurement, Performance

Additional Key Words and Phrases: Real-time panorama video, system integration, camera array, sensor tracking, video annotation, sport analytics, soccer system

ACM Reference Format:

Håkon Kvale Stensland, Vamsidhar Reddy Gaddam, Marius Tennøe, Espen Helgedagsrud, Mikkel Næss, Henrik Kjus Alstad, Asgeir Mortensen, Ragnar Langseth, Sigurd Ljødal, Øystein Landsverk, Carsten Griwodz, and Pål Halvorsen. 2014. Bagadus: An integrated real-time system for soccer analytics. ACM Trans. Multimedia Comput. Commun. Appl. 10, 1s, Article 14 (January 2014), 21 pages.
DOI: http://dx.doi.org/10.1145/2541011

1. INTRODUCTION

Sport analysis has become a large industry, and a large number of (elite) sports clubs study their game performance, spending a large amount of resources. This analysis is performed either manually or using one of the many existing analytics tools. In the area of soccer, several systems enable trainers

This work has been performed in the context of the *iAD* Centre for Research-Based Innovation (project number 174867) funded by the Norwegian Research Council.
H. K. Stensland's (corresponding author) email: haakonks@ifi.uio.no.

DOI: http://dx.doi.org/10.1145/2541011

and coaches to analyze the game play in order to improve the performance. For instance, in Interplay Sports [2013], video streams are manually analyzed and annotated using a soccer ontology classification scheme. ProZone [2013] automates some of the manual annotation process by video-analysis software. In particular, it quantifies player movement patterns and characteristics like speed, velocity, and position of the athletes, and it has been successfully used at, for example, Old Trafford in Manchester and Reebook Stadium in Bolton [Salvo et al. 2006]. Similarly, STATS SportVU Tracking Technology [Stats 2013] uses video cameras to collect the positioning data of the players within the playing field in real time. This is further compiled into player statistics and performance. Camargus [2013] provides a very nice video technology infrastructure but lacks other analytics tools. As an alternative to video analysis, which often is inaccurate and resource hungry, both the Cairo's VIS.TRACK [Cairos Technologies 2013b] and ZXY Sport Tracking [ZXY 2013] systems use global positioning and radio-based systems for capturing performance measurements of athletes. Thus, these systems can present player statistics, including speed profiles, accumulated distances, fatigue, fitness graphs and coverage maps, in many different ways, such as charts, 3D graphics, and animations.

To improve game analytics, video that replays real game events becomes increasingly important. However, the integration of the player statistics systems and video systems still requires a large amount of manual labor. For example, events tagged by coaches or other human expert annotators must be manually extracted from the videos, often requiring hours of work in front of the computer. Furthermore, connecting the player statistics to the video also requires manual work. One recent example is the Muihtu system [Johansen et al. 2012], which integrates coach annotations with related video sequences, but the video must be manually transferred and mapped to the game timeline.

As these examples show, there exist several tools for soccer analysis. However, to the best of our knowledge, there does not exist a system that fully integrates all these features. In this respect, we have presented earlier [Halvorsen et al. 2013] and demonstrated [Sægrov et al. 2012] a system called Bagadus. This system integrates a camera array video capture system with the ZXY Sport Tracking system for player statistics and a system for human expert annotations. Bagadus allows the game analytics to automatically play back a tagged game event or extract a video of events extracted from the statistical player data, for example, all sprints at a given speed. Using the exact player position provided by sensors, a trainer can also follow individuals or groups of players, where the videos are presented either using a stitched panorama view or by switching cameras. Our earlier work [Halvorsen et al. 2013; Sægrov et al. 2012] demonstrated the integrated concept but did not have all operations, like generation of the panorama video, in real time. In this article, we present enhancements providing live, real-time analysis and video playback by using algorithms to enhance the image quality, parallel processing, and offloading to co-processing units like GPUs. Our prototype is deployed at Alfheim Stadium (Tromsø IL, Norway), and we use a dataset captured at a Norwegian premier league game to demonstrate our system.

The remainder of the article is structured as follows. Next, in Section 2, we give a brief overview of the basic idea of Bagadus and introduce the main subsystems. Then, we look at the video-, tracking-, and analysis-subsystems in more detail in Sections 3, 4, and 5, respectively. Then, we briefly explain the case study at Alfheim Stadium in Section 6. Section 7 provides a brief discussion of various aspect of the system before we conclude in Section 8.

2. BAGADUS – THE BASIC IDEA

Interest in sports analysis systems has recently increased a lot, and it is predicted that sports analytics will be a real game-changer, that is, "statistics keep changing the way sports are played—and changing minds in the industry" [Dizikes 2013]. As already described, several systems exist, some for a long time, already providing game statistics, player movements, video highlights, etc. However, to a

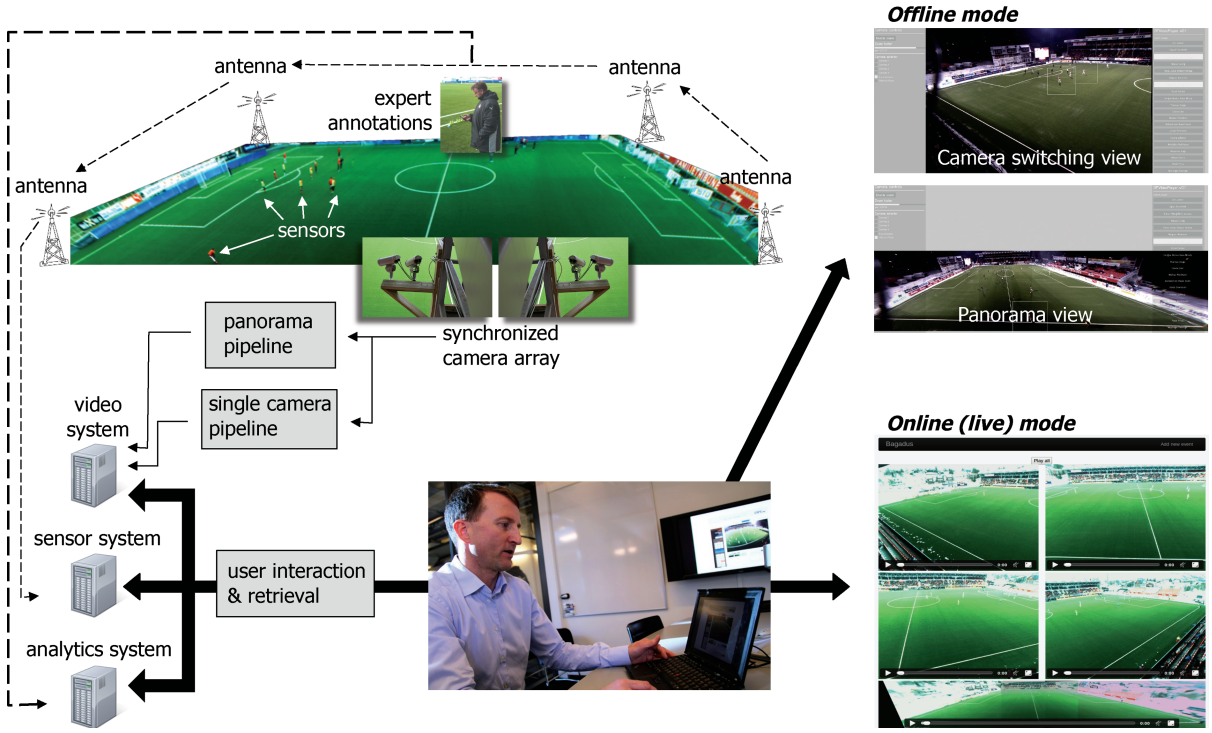


Fig. 1. Overall Bagadus architecture.

large degree, the existing systems are offline systems, and they require a large portion of manual work to integrate information from various computer systems and expert sport analytics. In this respect, *Bagadus* is a prototype that aims to fully integrate existing systems and enable real-time presentation of sport events. Our system is built in cooperation with the Tromsø IL soccer club and the ZXY Sport Tracking company for soccer analysis. A brief overview of the architecture and interaction of the different components is given in Figure 1. The Bagadus system is divided into three different subsystems which are integrated in our soccer analysis application.

The *video* subsystem consists of multiple, small, shutter-synchronized cameras that record a high resolution video of the soccer field. They cover the full field with sufficient overlap to identify common features necessary for camera calibration and image stitching. Furthermore, the video subsystem supports two different playback options. The first allows playback of video that switches between streams delivered from the different cameras, either manually selecting a camera or automatically following players based on sensor information. The second option plays back a panorama video stitched from the different camera feeds. The cameras are calibrated in their fixed position, and the captured videos are each processed and stored using a capture–debarrel–rotate–stitch–encode–store pipeline. In an offline mode, Bagadus allows a user to zoom in on and mark player(s) in the retrieved video on the fly (see Figure 1), but this is not yet supported in the live mode used during the game.

To identify and follow players on the field, we use a *tracking* (sensor) subsystem. In this respect, tracking people through camera arrays has been an active research topic for several years. The accuracy of such systems has improved greatly, but there are still errors. Therefore, for stadium sports, an interesting approach is to use sensors on players to capture the exact position. In this area, ZXY Sport

Tracking [ZXY 2013] provides such a sensor-based solution that provides player position information. Bagadus uses this position information to track players, or groups of players, in single camera views, stitched views, or zoomed-in modes.

The third component of Bagadus is an *analytics* subsystem. Coaches have for a long time analyzed games in order to improve their own team's game play and to understand their opponents. Traditionally, this has been done by making notes using pen and paper, either during the game or by watching hours of video. Some clubs even hire one person per player to describe the player's performance. To reduce the manual labor, we have implemented a subsystem that equips members of the trainer team with a tablet (or even a mobile phone), where they can register predefined events quickly with the press of a button or provide textual annotations. In Bagadus, the registered events are stored in an analytics database and can later be extracted automatically and shown along with a video of the event.

Bagadus implements and integrates many well-known components to support our arena sports analytics application scenario. The main novelty of our approach is then the combination and integration of components enabling automatic presentation of video events based on the sensor and analytics data that are synchronized with the video system. This gives a threefold contribution: (1) a method for spatially mapping the different coordinate systems of location (sensor) data and video images to allow for seamless integration; (2) a method for recording and synchronizing the signals temporally to enable semantic extraction capabilities; and (3) the integration of the entire system into an interactive application that can be used online and offline.

Thus, in the offline mode, Bagadus will, for example, be able to automatically present a video clip of all the situations where a given player runs faster than 10 meters per second or when all the defenders were located in the opponent's 18-yard box (penalty box). Furthermore, we can follow single players and groups of players in the video and retrieve and play back the events annotated by expert users. Thus, where people earlier used a huge amount of time analyzing the game manually, Bagadus is an integrated system where the required operations and the synchronization with video is automatically managed. In the online mode, Bagadus receives expert annotated events by the team analytics team and enables immediate playback during a game or a practice session.

3. VIDEO SUBSYSTEM

To be able to record high-resolution video of the entire soccer field, we have installed a camera array using small industry cameras which, together, cover the entire field. The video subsystem then extracts, process, and delivers video events based on given time intervals, player positions, etc. There are two versions of the video subsystem. One non-real-time system and one live real-time system. Both the video subsystems support two different playback modes. The first mode allows the user to play video from the individual cameras by manually selecting a camera or by automatically following players. The second mode plays back a panorama video stitched from the four camera feeds. The non-real-time system plays back recorded video stored on disks, and because of the processing times, it will not be available before the match is finished. The live system, on the other hand, supports playing back video directly from the cameras, and events will be available in real time.

3.1 Camera Setup

To record high-resolution video of the entire soccer field, we have installed a camera array consisting of four Basler industry cameras with a 1/3-inch image sensor supporting 30fps and a resolution of 1280×960. The cameras are synchronized by an external trigger signal in order to enable a video-stitching process that produces a panorama video picture. For a minimal installation, the cameras are mounted close to the middle line under the roof covering the spectator area, that is, approximately 10 meters from the side line and 10 meters above the ground. With a 3.5mm wide-angle lens, each

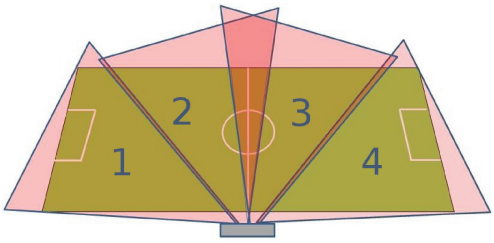


Fig. 2. Camera setup at Alfheim Stadium.

camera covers a field-of-view of about 68 degrees, that is, all four cover the full field with sufficient overlap to identify common features necessary for camera calibration and stitching (see Figure 2).

The cameras are managed using our own library, called Northlight, to manage frame synchronization, storage, encoding, etc. The system is currently running on a single computer with an Intel Core i7-3930K @ 3.2GHz and 16GB memory. Northlight integrates the SDK provided by Basler for the cameras, video encoding using x264, and color-space conversion using FFmpeg.

3.2 Digital Zoom

Bagadus supports digital zooming on tracked players, where the tracked player is kept in the center of the image while zooming in. An important operation here is interpolation, where we use known data to estimate values at unknown points when we resize or remap (i.e., distort) the image. In this respect, we have compared four different interpolation algorithms, that is, nearest neighbor, bilinear, bicubic, and Lanczos interpolation. In image processing, bicubic interpolation is often chosen over bilinear interpolation or nearest neighbor in image resampling when speed is not an issue. Lanczos interpolation has the advantages of bicubic interpolation and is known to produce sharper results than bicubic interpolation. In Bagadus, our initial tests show that the average interpolation times per frame are 4.2ms, 7.4ms, 48.3ms, and 240ms for nearest-neighbor, bilinear, bicubic, and Lanczos interpolation, respectively [Halvorsen et al. 2013]. Due to our time constraints, we use nearest-neighbor interpolation.

3.3 Stitching

Tracking game events over multiple cameras is a nice feature, but in many situations, it may be desirable to have a complete view of the field. In addition to the camera selection functionality, we therefore generate a panorama picture by combining images from multiple trigger-synchronized cameras. The cameras are calibrated in their fixed position using a classical chessboard pattern [Zhang 1999], and the stitching operation requires a more complex processing pipeline. We have alternative implementations with respect to what is stored and processed offline, but in general, we must (1) correct the images for lens distortion in the outer parts of the frame due to a fish-eye lens; (2) rotate and morph the images into the panorama perspective due to different positions covering different areas of the field; (3) correct the image brightness due to light differences; and (4) stitch the video images into a panorama image. Figure 3 shows the process of using four warped camera images into a single large panorama image. The highlighted areas in the figure are the regions where the cameras overlap.

After the initial steps, the overlapping areas between the frames are used to stitch the four videos into a panorama picture before storing it to disk. We first tried the open-source solutions given by computer vision library OpenCV, which are based on the automatic panoramic image stitcher by Brown and Lowe [2007], that is, we used the auto-stitcher functions using planar, cylindrical, and spherical projections. Our analysis shows that neither of the OpenCV implementations are perfect, having large execution times and varying image quality and resolutions [Halvorsen et al. 2013]. The fastest

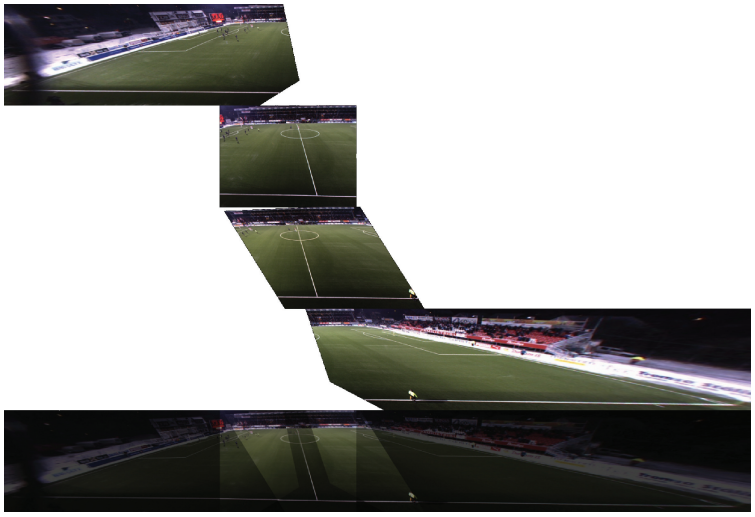
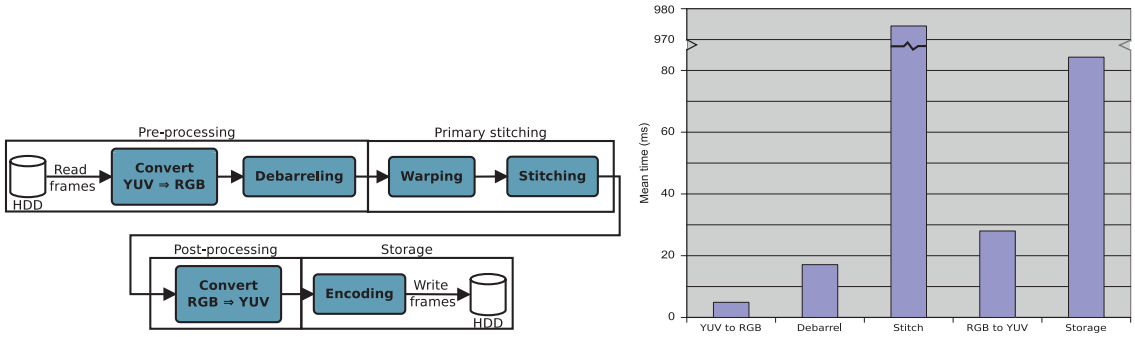


Fig. 3. The stitching process. Each image from the four different frames are warped and combined into a panorama.



(a) Stitching components in a single-threaded loop.

(b) Per-frame processing time (in milliseconds).

Fig. 4. The Bagadus single-threaded processing loop stitching implementation.

algorithm is the spherical projection, but it has severe barreling effects, and the execution time is 1746ms per frame—far above our real-time goal. Therefore, a different approach called homography stitching [Hartley and Zisserman 2004] has been selected, where we use a homography given by the projective geometry translating ZXY's coordinate system to pixel coordinates.

3.4 Non-Real-Time Processing Loop Implementation

As a first proof-of-concept prototype [Halvorsen et al. 2013], we implemented the stitching operation as a single-threaded sequential processing loop, as shown in Figure 4(a), that is, processing one frame per loop iteration. As seen in the figure, it consists of four main parts. One preprocessing part that reads video frames from either disk or cameras converts the video from YUV to RGB, which is used by the rest of the pipeline and debarreling to remove any barrel distortion from the cameras. For this version of the system, the debarreling functions in OpenCV is used. The next part is the primary stitching part using the homography-based stitching algorithm to stitch the four individual camera frames into a 7000×960 panorama frame. As we can observe from Figure 4(b), this is the most resource-demanding

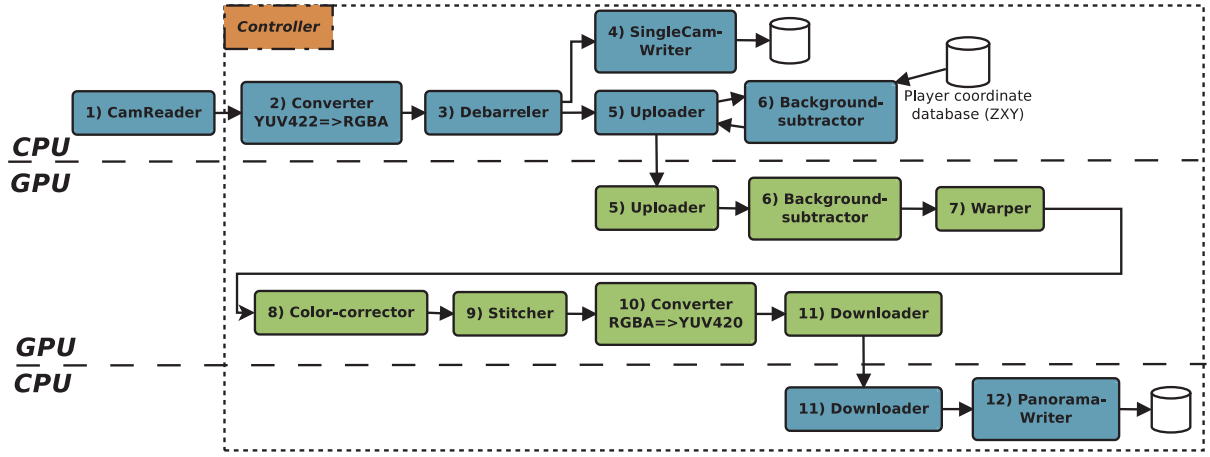


Fig. 5. The parallel and distributed processing implementation of the stitching pipeline.

part of the system. After the stitching, the postprocessing is responsible for converting the video back from RGB to YUV due to lacking support for RGB in the x264 video encoder. The single-threaded loop means that all the steps are performed sequentially for one set of frames before the next set of frames is processed. The performance is presented in Figure 4(b), and the total execution time per panorama frame exceeds 1100ms on average. In order to meet our 30fps requirement, our next approach improves the performance by parallelizing and distributing the operations in a processing pipeline and offloading several steps onto a GPU.

3.5 Real-Time Parallel and Distributed Processing Implementation

The previous sections displayed some severe processing overheads with respect to generating a 30fps panorama video in real time. In this section, we address this by implementing the modules in a parallel pipeline in contrast to the loop previously described, and we offload compute-intensive parts of the pipeline to a modern GPU, as seen in Figure 5.

3.5.1 *Implementation.* Figure 5 shows that the parallel pipeline is separated into two main parts: one part running on the CPU, and the other part running on a GPU. Several of the CPU modules in the pipeline are the same as in the non-real-time loop. The *CamReader*, *Converter*, *Debarreler*, *SingleCamWriter*, and *PenoramaWriters* are based on the same design, but are now running in their own threads and with an updated version of the x264 encoder. The *controller module* is new and is responsible for initializing the pipeline, synchronizing the different modules, handling global errors and frame drops, and transferring data or data pointers between the different modules. The controller also checks the execution speed. If an earlier step in the pipeline runs too slow, and one or more frames have been lost from the cameras, the controller tells the modules in the pipeline to skip the delayed or dropped frame and reuse the previous frame.

A *background subtractor* module is running both on the CPU and on the GPU. This module is new in the pipeline and is responsible for determining which pixels of a video belong to the foreground and which pixel belong to the background. The background subtractor can also get input from the ZXY sensor system to improve the performance and precision. Even though we have enhanced the background subtraction with sensor data input, there are several implementation alternatives. When determining which algorithm to implement, we evaluated two different alternatives, that is,

those of Zivkovic [2004] and Zivkovic and van der Heijden [2006] and those of KaewTraKulPong and Bowden [2001]. Both algorithms uses a Gaussian mixture model (GMM), are implemented in OpenCV, and have shown promising results in other surveys [Brutzer et al. 2011]. In the end, Zivkovic provided the best accuracy, which is important for our scenario, and it was therefore selected.

There are also several modules that are running primarily on the GPU. The *Uploader* and *Downloader* are managing the dataflow to and from the GPU. The Uploader transfers RGB frames and the background subtraction player pixel maps from the CPU to the GPU for further processing. The Downloader transfers back the stitched video in YUV 4:2:0 format for encoding. Both modules use double-buffering and asynchronous transfers.

The main parts of the panorama creation is performed by the *warper*, *color-corrector*, and *stitcher* modules running on the GPU. The warper module warps (as previously described) the camera frames and the foreground masks from the background subtractor module to fit the common panorama plane. Here, we used the Nvidia Performance Primitives library (NPP) for an optimized implementation. The Color-corrector in this implementation is added to the pipeline because it is nearly impossible to calibrate the cameras to output the exact same colors because of the uncontrolled lighting conditions. This means that, to generate a best-possible panorama video, we correct the colors of all the frames to remove eventual color disparities. This operation is performed after the images are warped. The reason for this is that locating the overlapping regions is easier with aligned images, and the overlap is also needed when stitching the images together. The implementation is based on the algorithm presented in Xiong and Pulli [2009], which has been optimized to run in real-time with CUDA.

The stitcher module is similar to the homography stitcher in the loop implementation, where a seam is created between the overlapping camera frames. Our previous approach uses static cuts for seams, which means that a fixed rectangular area from each frame is copied directly to the output frame. Static cut panoramas are very fast but can introduce graphical errors in the seam area, especially when there is movement in the scene, as illustrated in Figure 6(a). Thus, to make a better visual result, a dynamic cut stitcher is introduced. This module now creates seams by first creating a rectangle of adjustable width over the static seam area. Then, it treats all pixels within the seam area as graph nodes. Each of these edges' weights are calculated using a custom function that compares the absolute color difference between the corresponding pixel in each of the two frames we are trying to stitch. The weight function also checks the foreground masks from the background subtractor to see if any player is in the pixel, and if so, it adds a large weight to the node. We then run a simplified version of the Dijkstra graph algorithm (only going up in the image) on the graph to create a minimal cost route from the bottom of the image to the end at the top. An illustration of how the final seam looks can be seen in Figure 6(b), while the seams without and with color correction are shown in Figures 6(c) and 6(d).

3.5.2 *Execution Time Evaluation.* To evaluate the processing performance of the parallel and distributed processing pipeline implementation, we used a single computer with an Intel Server Adapter i350-T4 for connecting the four cameras with gigabit ethernet, an Intel Core i7-3930K six-core processor with 32GB RAM, and a single Nvidia GeForce GTX Titan graphics processor.

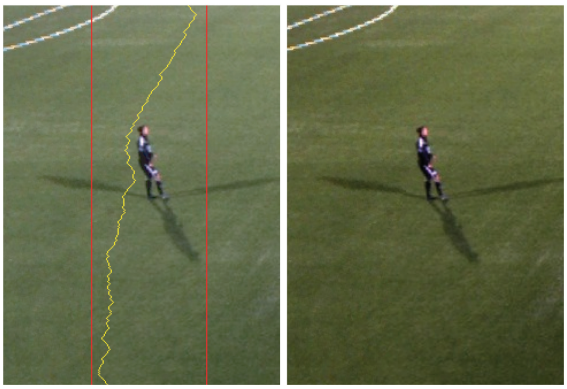
The overall performance of the parallel pipeline is shown in Figure 7(a). The CPU modules are marked in blue, and the GPU modules are marked in green. The uploader and downloader module run both on the CPU and the GPU, but we have chosen to mark them as CPU modules, since they both are controlled by the CPU.

Images from all four cameras are asynchronously transfered to the GPU as soon as they are available. The number of threads and blocks on the GPU is automatically adjusted by how many cores are

(a) The original fixed-cut stitch with a straight vertical seam.

(b) The new dynamic stitch with color correction.

(c) Dynamic stitch with no color correction. In the left image, one can see the seam search area between the red lines, and the seam in yellow. In the right image, one clearly sees the seam going outside the player, but there are still color differences.

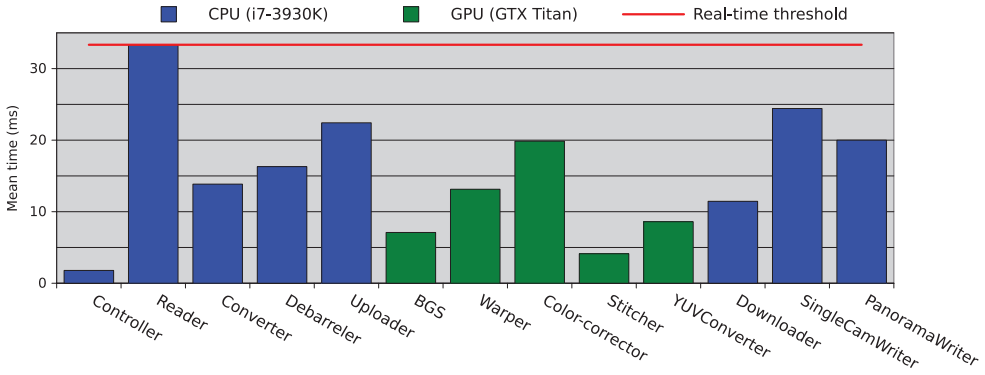


(d) Dynamic stitch with color correction. In the left image, one can see the seam search area between the red lines and the seam in yellow. In the right image, one cannot see the seam, and there are no color differences.

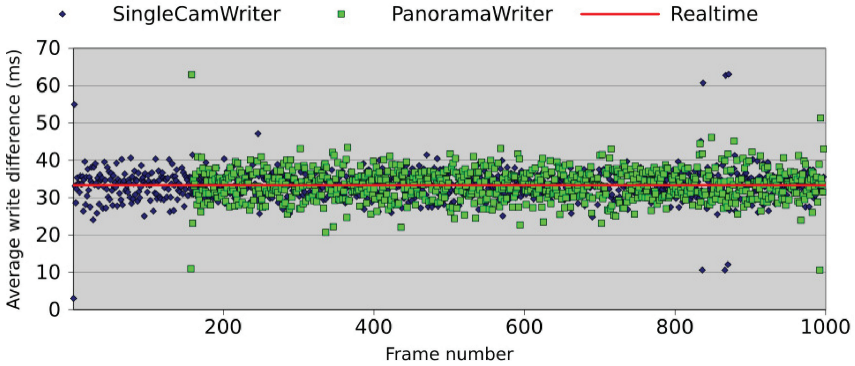
Fig. 6. Stitcher comparison: improving the visual quality with dynamic seams and color correction.

available on the GPU. The modules executing on the GPU synchronize with barriers: when one module finishes, the next will be stared. Data is stored in global memory, and pointers to the data are transfered between the different modules. When processing is finished on the GPU, data is asynchronously transfered back to the CPU for encoding and writing to disk.

We can see that when executing the whole pipeline, all modules perform well below the real-time threshold. Note that the reader module is limited by the cameras which produce a new frame every 33ms. Remember that all these modules run in parallel, sharing the processing elements. Thus, since all modules perform better than the 33ms threshold, we are able to deliver panorama frames in real time. This is further demonstrated by measuring the differences between the single camera writes and the differences between the panorama writes. In Figure 7(b), we present the write differences between the frames, and we observe that a new frame is output every 33ms, that is, equal to the input rate of



(a) The processing performance of the different pipeline modules.



(b) Pipeline write differences (showing times for 1,000 frames). Note that the delayed start of panorama writes is caused by the frame delay buffer implemented in the uploader module.

Fig. 7. The processing performance of the parallel and distributed processing pipeline.

the cameras. These results show that our parallel and distributed processing implementation executes in real time on a single off-the-shelf computer.

4. TRACKING SUBSYSTEM

Tracking people through camera arrays has been an active research topic for several years, and many approaches have been suggested (e.g., [Ben Shitrit et al. 2011; Berclaz et al. 2011; Jiang et al. 2007; Xu et al. 2004]). The accuracy of such tracking solutions vary according to scenarios and is continuously improving, but they are still giving errors, that is, both missed detections and false positives [Ben Shitrit et al. 2011]. Often these approaches perform well in controlled lighting conditions, like indoor sport arenas, but the widely varying light conditions in an outdoor stadium provide bigger challenges.

For stadium sports, an interesting approach is to use sensors on players to capture the exact position. ZXY Sport Tracking [ZXY 2013] provides such a solution, where a sensor system submits position and orientation information at a maximum accuracy error of about one meter at a frequency of 20Hz. As indicated in Figure 1, the players wear a data chip with sensors that sends signals to antennas located around the perimeter of the pitch. The sensor data is then stored in a relational database system. Based

on these sensor data, statistics like total length run, number of sprints of a given speed, foot frequency, heart rate, etc., can be queried in addition to the exact position of all players at all times. Due to the availability of the ZXY system at our case study stadium, Bagadus uses the sensor system position information to extract videos of, for example, particular players, and the rest of the system can be used to extract time intervals of the video (e.g., all time intervals where player X sprints towards his own goal).

The ZXY sensor belt is worn by all the players on TIL (the home team); it is voluntary for the visiting team to use the sensor belts. If they choose to use the belts, they will have access to the data recorded during the match. The belts are small and compact and do not disturb the players during the match; they are also approved by FIFA for use during international matches.

Although the amount of data generated by the position sensors is small compared to video, a game of 90 minutes still produces approximately 2.4 million records. Nevertheless, as we show later in Section 6, we still have reasonable response times from when we send a complex database query until the video starts to play the corresponding query result events.

4.1 Mapping Sensor Positions to Image Pixels

The ZXY system reports the players' positions on the field using the Cartesian coordinate system. In order to locate a player in the video, we need a transformation from the sensor coordinates to the image pixels for all valid pixel coordinates in a video frame. In this respect, we calculate a 3×3 transformation matrix using fixed known points on the field, as shown in Figure 8(a). Then, using the homography between two planes, each plane can be warped to fit the other, as shown in Figures 8(c) and 8(d), using camera 2 as an example. The accuracy of the mapping is fairly good, that is, only in the outer areas of the image where debarreling have changed some pixels can we see a very small deviation between the planes. However, if we look at the mapping to the stitched image in Figure 8(b), the accuracy is reduced due to imperfections in the image processing when debarreling and, in particular, when warping and rotating. Nevertheless, at the distance between the cameras and the players, the accuracy seems to be good enough for our purposes (though inaccuracies in the mapping might also contribute to inaccurate tracking, as shown later).

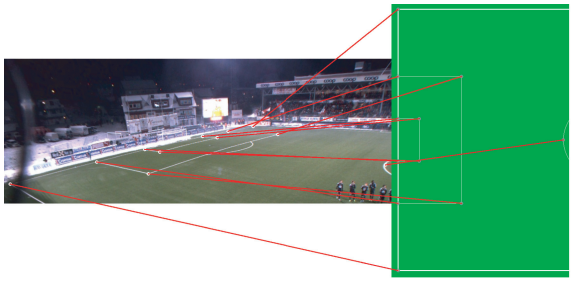
In order to have a system where the players are tracked in real time, the $ZXY(x, y) \rightarrow pixel(u, v)$ mapping using the 3×3 matrix must be fast. A profile of the system when tracking all 22 soccer players indicates that about 7.2–7.7 microseconds are consumed for this operation, that is, coordinate translation is hardly noticeable compared to the other components in the system.

4.2 Automatic Camera Selection

As shown in Figure 2, the four cameras cover different parts of the field. To follow a player (or group of players) and be able to automatically generate a video selecting images across multiple cameras, we also need to map player positions to the view of the cameras. In this respect, we use the same mapping as described in Section 4.1, using our own transformation matrix for each camera. Selecting a camera is then only a matter of checking if the position of the player is within the boundaries of the image pixels. When tracking multiple players, we use the same routine and count the number of tracked players present in each camera and select the camera with the most tracked players.

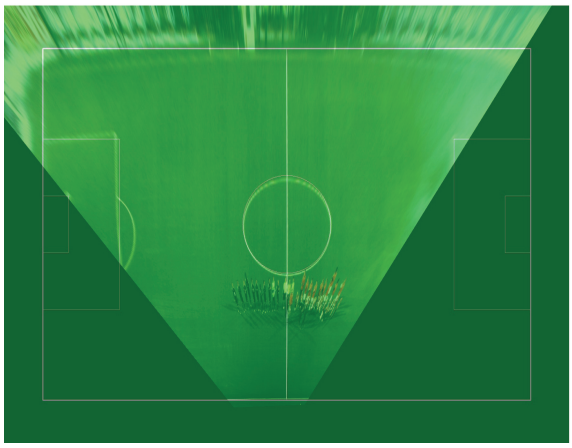
5. ANALYTICS SUBSYSTEM

To improve a team's performance and understand their opponents, coaches analyze the game play in various ways. Traditionally, this has been done by making notes using pen and paper, either during the game or by watching hours of video. To reduce the manual labor, we have, in close colaboration with the coach-team, developed Muithu, a novel notational analysis system [Johansen et al. 2012] that



(a) Mapping between coordinates in the ZXY plane and the image plane.

(b) Warping and superimposing the ZXY plane onto the stitched image (cropped out only parts of the field for readability).



(c) Warping and superimposing the image from camera 2 to the ZXY plane.

(d) Warping and superimposing the ZXY plane onto the image from camera 2.

Fig. 8. Pixel mapping between the video images and the ZXY tracking system.

is non-invasive for the users, mobile, and lightweight. A cellular phone is used by head coaches during practice or games for annotating important performance events. A coach usually carries a cellular, even during practice. Thus, to avoid any extra coach devices, the cellular is used in the notational process as a notational device. Input is given using the tile-based interface shown in Figures 9(b) and 9(c), and Figure 9(a) illustrates use of the system by a coach during a recent game in the Norwegian elite division. Our experience indicates that this simple drag-and-drop user interaction requires in the order of 3 seconds per notational input. All the events in the app can be custmized by the coaches, and the number of input notations for a regular 90-minute elite soccer game varies slightly over different games, but for the 2012 season, the average is in the order of 16 events per game [Johansen et al. 2012].

In order to be usable during a game, the user interface of Muithu has to be easy to use and fast. It is therefore based on managing tiles in a drag-and-drop fashion, and it can be easily configured with input tiles and hierarchies of tiles. In the case study described in Section 6, one preferred configuration pattern for general practice is to have a two-layer hierarchy, where the root node is a number or all of the players involved. The next layer is a set of 3–4 training goals associated with each individual player. By simply touching the picture of a player on a tile, his specific training goals appear on adjacent

(a) Event tagging. (b) Select player. (c) Drag to event.

Fig. 9. Operation of the mobile device during a game (a). Select a player (b) and drag the image tile to the appropriate event type (c) to register an event.

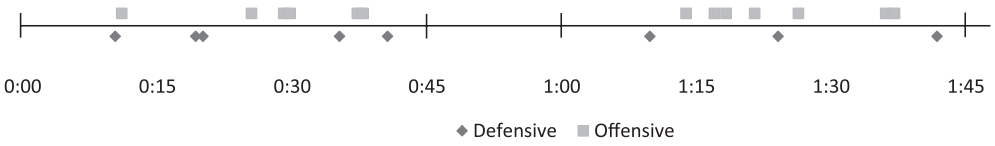


Fig. 10. An example of notations captured during a game (time axis in HH:MM after game start). Observe that offensive notations are displayed above the timeline, defensive notations below.

tiles. Dragging the face tile over one of these goal tiles is then sufficient for capturing the intended notation.

For heated game purposes, a simpler configuration is preferred: typically one tile for offensive and one for defensive notations (see Figure 9(c)). Using this interface as an example, Figure 10 depicts the distribution of such notations during a home game in September 2012.

Recall of performance-related events without any observation aids is traditionally problematic in soccer, but the recall abilities of the head coaches using Muithu have improved rapidly approaching almost 1 (100%). A small but fundamental detail is the use of *hindsight* recording, which implies that the coach observes an entire situation and determines afterwards whether it was a notable event worth capturing. By tagging in retrospect, the coach essentially marks the end of a notable event, and the system finds the start of the sequence by a preconfigured interval length. This simple yet not so intuitive approach has reduced the number of false positives, that is, increased precision dramatically.

Only those events tagged by the head coaches are retrieved for movement patterns, strategy, and tactics evaluation. The key to this process is that the video footage is automatically retrieved from the video system when the event is selected in the video playout interface. This scales both technically and operationally, which enables expedited retrieval. The video sequence interval according to the recorded event time-stamp is a configuration option easy to change, but operational practice has shown that an interval around 15 seconds is appropriate for capturing the event on video. It is also possible to adjust this interval, both when the event is created and during playback.

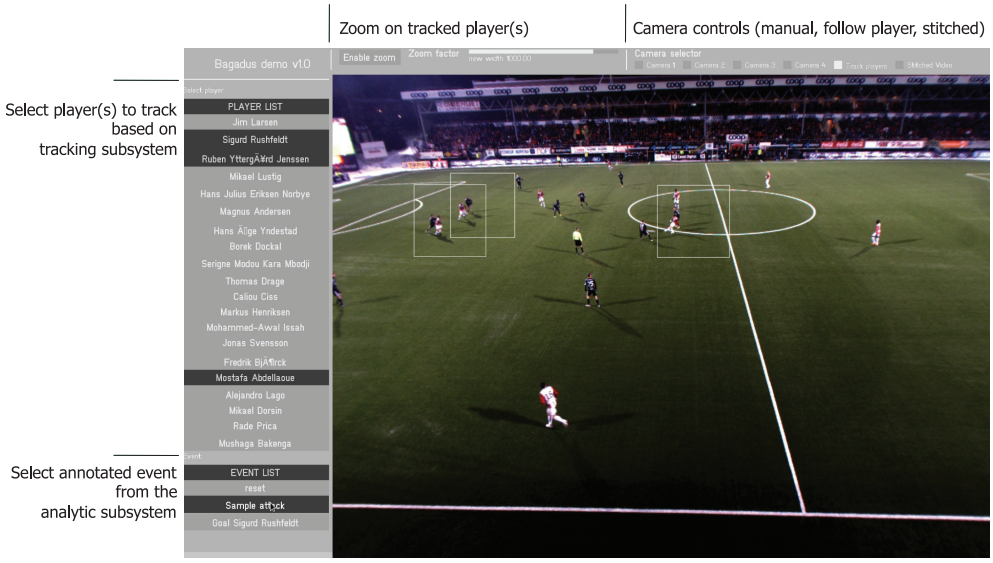


Fig. 11. The offline Linux interface (tracking three players in camera-switching mode).

```
SELECT timestamp, x_pos, y_pos
FROM zxy_oversample
WHERE (y_pos > 17.5 AND y_pos < 50.5)
  AND (x_pos >  0.0 AND x_pos < 16.5)
  AND timestamp > 45
  AND tag_id = ("the tag_id of player X")
```

Fig. 12. Example query.

6. ALFHEIM STADIUM CASE STUDY

We have a prototype installation at Alfheim Stadium in Tromsø (Norway). The interface of the offline prototype [Halvorsen et al. 2013][1] is shown in Figure 11, where we can follow and zoom in on particular player(s) and play back expert-annotated events from the game in panorama video- and camera-switching mode.

In the offline mode, the system has support for generating automatic summaries, that is, selecting multiple time intervals and playing it out as one video (not yet integrated into the user interface). This means that the game analytics, for example, may perform queries against the ZXY database and get the corresponding video events. An example could be to see "all the events where defender X is in the other team's 18-yard box in the second half". In this example, the position and corresponding time of player X in the former example is returned by the pseudo-query shown in Figure 12. Here, the player is located within the [0.0, 16.5] in the x-coordinate and [17.5, 50.5] on the y-axis (using the metric system) defining the 18-yard box. The returned time stamps and positions are then used to select video frames (selecting the correct camera or the panorama picture) which are automatically presented to the user. Extracting summaries like the preceding example used to be a time-consuming

[1]A video of the (offline) Linux-based system is available at `http://www.youtube.com/watch?v=1zsgvjQkL1E`. At the time of the submission, we have not been able to make a video of the online system.

Table I. Latency Profiling (in ms) of the Event Extraction Operation Using ZXY and the Video System

Operation	Mean	Minimum	Maximum	Standard deviation
Query received	2.7	1.5	5.3	0.38
Query compiled	4.9	2.9	7.8	0.61
First DB row returned	500.4	482.4	532.1	5.91
First video frame displayed	671.2	648.0	794.6	8.82

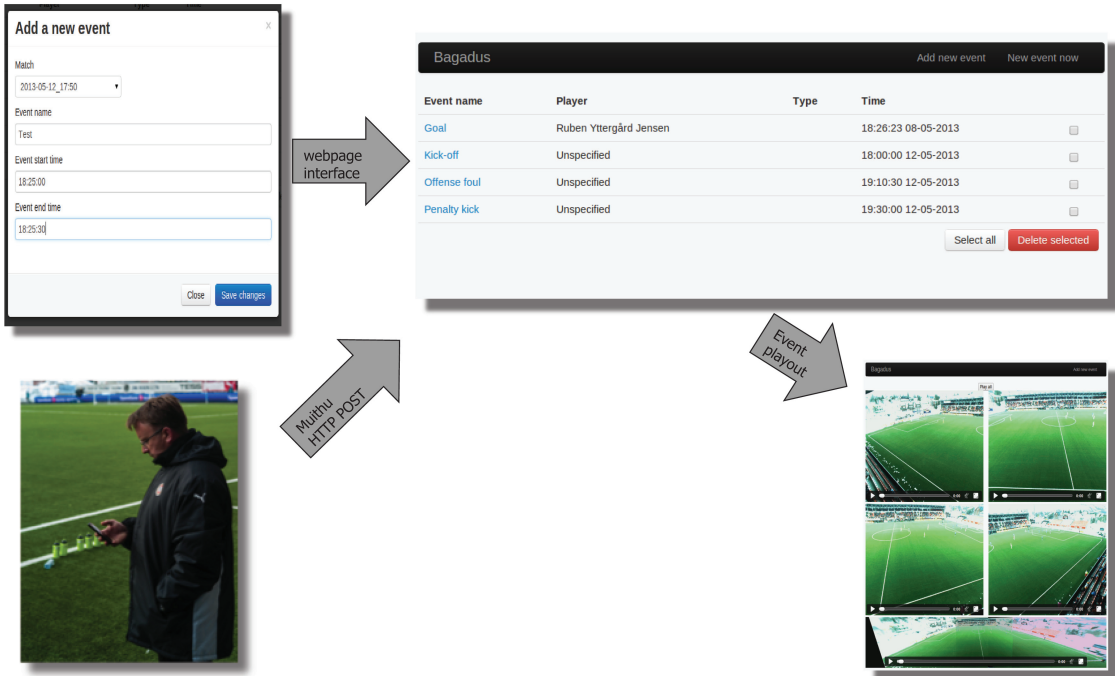


Fig. 13. The online HTML5 interface used for expert annotated events. Here the events are sorted by player and then time.

and cumbersome (manual) process. Bagadus, on the other hand, automates the video generation. For instance, the response time of returning the resulting video summary from the preceding query was measured to be around 671ms (see Table I for more detailed statistics). Note that this was measured on a local machine, that is, if the display device is remote, network latency must be added. The SQL queries are made for expert users. We have also implemented a number of predefined queries that are availible in the user interface.

As shown in the online mode HTML5 interface in Figure 13, we can in a similar way extract video events based on expert annotations. Events may be tagged through a Web interface or using the mobile phone sending an HTTP POST command, and all annotated events from the analytics subsystem then appear in the list of events. Using a standard Web browser, the corresponding videos start by clicking on the event title. Thus, the integration of subsystems enable event playout during a game or a practice session.

(a) Bagadus. (b) Broadcaster 1. (c) Broadcaster 2.

Fig. 14. Lighting challenges at Alfheim Stadium. Comparison between Bagadus and two professional Norwegian broadcasters. (The images are from the same game but different situations during the game.)

7. DISCUSSION

Performance analysis of athletes in the sports industry is a growing field of interest. In the context of computer systems managing the technical challenges, there are numerous issues that must be addressed to provide real-time operations. In this respect, our Bagadus soccer analysis application integrates a sensor system, soccer analytics annotations, and video processing of a video camera array. There exist several components that can be used, and we have investigated several alternatives in our research. Furthermore, by providing a parallel video-processing pipeline distributing load on multiple CPUs and GPUs, Bagadus supports analysis operations at 30fps. Note, however, that our prototype aims to prove the possible integration at the system level with real-time performance, rather than being optimized for optimal resource utilization, that is, there are several areas with potential for further optimizations.

For example, most stitching processes assume the pinhole camera model where there is no image distortion because of lenses. In our work, we have observed that a camera can be calibrated to minimize lens distortion caused by imperfections in a lens but making a perfect calibration is hard. This makes finding a homography between planes difficult and error-prone, which affects the stitched result.

Another problem we have identified is parallax errors. In this respect, OpenCV's auto-stitcher has functionality for selecting seams at places where parallax errors are less obvious. However, when stitching video recorded from cameras capturing the field from the same position but with different angles (requiring rotation and warping), parallax errors will become prominent. Such problems arise because the centers of the projection of different cameras are not aligned well enough. We are looking at solutions to eliminate this problem: one of the most interesting solutions is the arrangement of cameras over cross, such as each camera capturing one side of the field, similar to Fehn et al. [2006].

Furthermore, the stitching itself can be moved from a homography-based stitching with dynamic seams to avoid moving objects to more advanced warping techniques, like the one mentioned in Lin et al. [2011]. A rather intriguing challenge would be to incorporate such a process into Bagadus and perform this approach in real time, too. Moreover, we have later found several promising alternative algorithms in the area of video processing (vision) (e.g., [Lin et al. 2011; Jin 2008; Li and Du 2010; Ozawa et al. 2012]), and there is also scope for further improvement in color correction [Xiong and Pulli 2010], since the exposure times and other parameters across the cameras may vary.

A major challenge is managing variations in lighting conditions. In most weather conditions, our current setup works fine, but our main challenge here is a low and bright sun. The visual quality is exceptional when it is partly or completely cloudy, but the striking difference between the amount of light available from highlights and shadows during a clear day leaves us with a choice of having a good dynamic range in only one region. An example from Alfheim Stadium is shown in Figure 14. When

there are intensely bright and dark areas in the image (Figure 14(a)), most cameras have problems creating a representative image. Particularly, in our Alfheim case study, the location of the stadium is only 2,271km from the North Pole (69.6489986°N). The sun is significant lower on the sky than most of the habitable world, resulting in challenges, as shown in the figure. In such a case, aiming for good quality in highlights leads to loss of details in shadows. Our system currently lacks the ability to make an appropriate decision which often depends on the events on the field. Professional broadcasters also experience these problems, but they have people manning cameras (and thus also the exposure settings) as well as a someone controlling the live broadcast who also can perform manual adjustments (Figures 14(c) and 14(b)).

Our system needs to handle this without human interaction and in real time. The problem is related to suboptimal auto-exposure and insufficient dynamic range on the camera sensors. Improvements can be achieved several ways. In this respect, one could solve common auto-exposure problems as proposed in Kao et al. [2011] and use real-time assembling of high-dynamic-range (HDR) video by using low-dynamic-range images [Ali and Mann 2012; Guthier et al. 2012]. Investigations of such approaches are currently ongoing.

The GPU implementation has been tested on an Nvidia GeForce Titan (GK110) GPU with compute 3.5 capabilities and has been profiled with Nvidia's Visual Profiler to investigate the possibilities of scaling the pipeline to more cameras with higher resolution. Currently, we are only using a small portion of the available PCI Express bandwidth between the CPU and the GPU. Our uploader uses 737MB/sec, and our downloader uses 291MB/sec. The theoretical bidirectional bandwidth of a 16-lane PCI Express 3.0 link is 16GB/sec. The real-time pipeline uses seven kernels running concurrently on the GPU. These seven kernels have an average compute utilization of 14.8% on this GPU. The individual CUDA kernels are also not optimized for the architecture used in our benchmarks, since the priority was to get the entire pipeline in real time. There is therefore a lot of potential on the GPU for scaling the pipeline to a larger number of cameras with higher resolution.

In our case study, we have analyzed data and retrieving video from only one game. However, we have shown earlier how one could search for events and generate video summaries on-the-fly in terms of a video playlist [Johansen et al. 2009] over large libraries of video content. In the used test scenario, there are events identified from multiple subcomponents, for example, the sensor system and the annotation system. In many cases, it would be valuable to be able to search across all the metadata and also across games. This is a feature we are currently adding, that is, the underlying video system fully supporting the video extraction, but the interface has not yet been implemented.

The design of Bagadus having three tightly integrated, but still separate subsystems, enables easy subsystem replacement. For example, we have used ZXY to track players, providing some extra nice features (heart rate, impact, etc.). However, tracking players (or, generally, objects) through video analysis is a popular research area (e.g., both in sports [Fehn et al. 2006; Yongduek et al. 1997; Iwase and Saito 2004; Kang et al. 2003] and surveillance [Fuentes and Velastin 2006; Chen et al. 2011; Siebel and Maybank 2002]). Thus, the Bagadus idea should easily be transferable to arenas where the sensor system is unavailable or to other arena sports, like ice hockey, handball, baseball, tennis, American football, rugby, etc. Similarly, video-processing components can easily be replaced to match other codec's and other filters or to suit other end devices and platforms. Equally, the annotation system can be replaced (or expanded) to retrieve metadata of events from other sources, like on-the-fly live text commentaries found in newspapers and online TV stations, like we did in our DAVVI system [Johansen et al. 2009].

One engineering challenge in systems like Bagadus is time synchronization at several levels. First, to be able to stitch several images to a panorama image, the shutters must be synchronized at the sub-millisecond level, that is, as the players are moving fast across cameras, imperfect synchronization

Fig. 15. An example of when the tracking box fails to capture the tracked player. Even though our analysis of the system indicates very infrequent errors, it may be various reasons for failed tracking, for example, both clock skew, sensor system accuracy, and coordinate mapping.

would lead to massive pixel offsets across camera perspectives resulting in severely blurred composite images of players. This is currently solved using an external trigger box (i.e., embedded trigger controller based on an ATMega16 microcontroller) which sends an input signal to the camera's electronic shutter. Another observed challenge in this respect is that the clock in the trigger box drifts slightly compared to our computer clocks depending on temperature (which changes a lot under the harsh outdoor conditions in northern Norway). While the shutters across cameras remains in sync, a drifting clock leads to slight variations in frame rate of the captured video. Similarly, Bagadus integrates several subsystems running on different systems. In this respect, the clock in the ZXY system also slightly drifts compared to the clock in our video capture machines (which will be potentially solved when we switch ZXY to the same NTP server). So far, these small errors have been identified, but since we alleviate the problem in our video player by fetching a couple of seconds more video data around a requested event time stamp, the effects have been small. Another more visible (still very infrequent) effect of time skew is that the box-marker marking the players in the video gives small misplacement errors, as shown in Figure 15. However, the bounding box is slightly larger compared to the personobject itself. This means that the player is usually contained in the box, even though not exactly in the middle. At the current stage of our prototype, we have not solved all the synchronization aspects, but it is subject to ongoing work.

The ZXY's tracking system installed at Alfheim Stadium has a maximum accuracy error of one meter (their new system reduces this error down to a maximum of 10 centimeters). This means that if a player is at a given position, the measured coordinate on the field could be $\pm$ one meter. This could give effects like those shown in Figure 15, but for the practical purposes of our case study, it has no influence on the results.

The players are tracked as described using the ZXY Sport Tracking system. Another issue which is not yet included in Bagadus is ball tracking, that is, a feature that could potentially improve the analysis further. Even though ball tracking is not officially approved by the international soccer associations due to the limited reliability and failure to provide 100% accuracy, there exist several approaches. For example, Adidas and Cairos Technologies have tried to put sensors inside the ball, that is, using a magnetic field to provide pinpoint accuracy of the ball's location inside the field [McKeegan 2007; Cairos Technologies 2013a]. Other approaches include using multiple cameras to track the ball.

Hawk-Eye [2013] is one example which tries to visually track the trajectory of the ball and display a record of its most statistically likely path as a moving image. Nevertheless, ball tracking in Bagadus is a future feature.

This article presents Bagadus in the context of sports analysis for a limited user group within a team. However, the applicability we conjecture is outside the trainer and athlete sphere, since we have a potential platform for next-generation personalized edutainment. We consider use case scenarios where users can subscribe to specific players, events, and physical proximities in real time. For instance, when the main activity is around the opponent goal, a specific target player can be zoomed into. Combine this with commonplace social networking services, and we might have a compelling next-generation social networking experience in real time.

8. CONCLUSIONS

We have presented a real-time prototype of a sports analysis system called Bagadus targeting automatic processing and retrieval of events in a sports arena. Using soccer as a case study, we described how Bagadus integrates a sensor system, a soccer analytics annotations system, and a camera array video processing system. Then, we showed how the system removes the large amount of manual labor traditionally required by such systems. We have described the different subsystems and the possible trade-offs in order to run the system in real-time mode. Compared to our initial demonstrator [Halvorsen et al. 2013], the improved processing pipeline parallelizing the operational steps and distributing workload to both CPUs and GPUs enables real-time operations, and the picture quality has been improved using dynamic seams and color correction. Furthermore, we have presented functional results using a prototype installation at Alfheim Stadium in Norway. Bagadus enable a user to follow and zoom in on particular player(s), playback events from the games using the stitched panorama video and/or the camera switching mode, and create video summaries based on queries to the sensor system.

Finally, there are still several areas for future improvements, for example, in the areas of image quality improvements handling a wide range of lighting conditions, performance enhancements as our profiling results show that we can optimize the resource utilization further and subjective user evaluations. All these areas are subjects for ongoing work, for example, we are testing algorithms discussed in Section 7 for improving the image quality, we are evaluating higher-resolution cameras like the 2K Basler aca2000-50gc, and we are further optimizing and distributing algorithms onto multiple cores and offloading calculations to GPUs for speed improvements and better utilization of both cores and buses.

ACKNOWLEDGMENTS

The authors also acknowledge support given by Kai-Even Nilssen and Håvard Johansen who have been helpful with the practical installation at Alfheim, the coaches in TIL (Per-Mathias Høgmo and Agnar Christensen) who have given feedback on the functionality of the system, and Rune Stoltz Bertinussen for taking player photos.

REFERENCES

Mir Adnan Ali and Steve Mann. 2012. Comparametric image compositing: Computationally efficient high dynamic range imaging. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 913–916.

Horesh Ben Shitrit, Jerome Berclaz, Francois Fleuret, and Pascal Fua. 2011. Tracking multiple people under global appearance constraints. In *Proceedings of the IEEE International Conference on Computer Vision (CCV)*. 137–144.

Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. 2011. Multiple object tracking using k-shortest paths optimization. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 9, 1806–1819.

Matthew Brown and David G. Lowe. 2007. Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vision* 74, 1, 59–73.

S. Brutzer, B. Hoferlin, and G. Heidemann. 2011. Evaluation of background subtraction techniques for video surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1937–1944.

Cairos Technologies. 2013a. Goal Line Technology (GLT) system. http://www.cairos.com/unternehmen/gltsystem.php.

Cairos Technologies. 2013b. VIS.TRACK. http://www.cairos.com/unternehmen/vistrack.php.

Camargus. 2013. Premium Stadium Video Technology Inrastructure. http://www.camargus.com/.

Chao-Ho Chen, Tsong-Yi Chen, Je-Ching Lin, and Da-Jinn Wang. 2011. People tracking in the multi-camera surveillance system. In *Proceedings of the 2nd International Conference on Innovations in Bio-inspired Computing and Applications (IBICA)*. 1–4.

Peter Dizikes. 2013. Sports analytics: A real game-changer. http://web.mit.edu/newsoffice/2013/sloan-sports-analytics-conference-2013-0304.html.

Christoph Fehn, Christian Weissig, Ingo Feldmann, Markus Muller, Peter Eisert, Peter Kauff, and Hans Bloss. 2006. Creation of high-resolution video panoramas of sport events. In *Proceedings of the 8th IEEE International Symposium on Multimedia (ISM)*. 291–298.

Luis M. Fuentes and Sergio A. Velastin. 2006. People tracking in surveillance applications. *Image Vision Comput.* 24, 11, 1165–1171.

Benjamin Guthier, Stephan Kopf, and Wolfgang Effelsberg. 2012. Optimal shutter speed sequences for real-time HDR video. In *Proceedings of the IEEE International Conference on Image Systems and Techniques (IST)*. 303–308.

Pål Halvorsen, Simen Sægrov, Asgeir Mortensen, David K. C. Kristensen, Alexander Eichhorn, Magnus Stenhaug, Stian Dahl, Håkon Kvale Stensland, Vamsidhar Reddy Gaddam, Carsten Griwodz, and Dag Johansen. 2013. Bagadus: An integrated system for arena sports analytics a soccer case study. In *Proceedings of the 4th ACM Multimedia Systems Conference (MMSys)*. 48–59.

R. I. Hartley and A. Zisserman. 2004. *Multiple View Geometry in Computer Vision* 2nd Ed. Cambridge University Press.

Hawk-Eye. 2013. Football::Hawk-Eye. http://www.hawkeyeinnovations.co.uk/page/sports-officiating/football.

Interplay Sports. 2013. The ultimate video analysis and scouting software. http://www.interplay-sports.com/.

Sachiko Iwase and Hideo Saito. 2004. Parallel tracking of all soccer players by integrating detected positions in multiple view images. In *Proceedings of the 7th International Conference on Pattern Recognition (ICPR)*. 751–754.

Hao Jiang, Sidney Fels, and James J. Little. 2007. A linear programming approach for multiple object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hailin Jin. 2008. A three-point minimal solution for panoramic stitching with lens distortion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1–8.

Dag Johansen, Håvard Johansen, Tjalve Aarflot, Joseph Hurley, Åge Kvalnes, Cathal Gurrin, Sorin Sav, Bjørn Olstad, Erik Aaberg, Tore Endestad, Haakon Riiser, Carsten Griwodz, and Pål Halvorsen. 2009. DAVVI: A prototype for the next generation multimedia entertainment platform. In *Proceedings of the 17th ACM International Conference on Multimedia (MM)*. 989–990.

Dag Johansen, Magnus Stenhaug, Roger Bruun Asp Hansen, Agnar Christensen, and Per-Mathias Høgmo. 2012. Muithu: Smaller footprint, potentially larger imprint. In *Proceedings of the 7th International Conference on Digital Information Management (ICDIM)*. 205–214.

P. Kaewtrakulpong and R. Bowden. 2001. An improved adaptive background mixture model for realtime tracking with shadow detection. In *Proceedings of the Video-Based Surveillance Systems*. 135–144.

Jinman Kang, Isaac Cohen, and Gerard Medioni. 2003. Soccer player tracking across uncalibrated camera streams. In *Proceedings of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*. 172–179.

Wen-Chung Kao, Li-Wei Cheng, Chen-Yu Chien, and Wen-Kuo Lin. 2011. Robust brightness measurement and exposure control in real-time video recording. *IEEE Trans. Instrument. Measur.* 60, 4, 1206–1216.

Jubiao Li and Junping Du. 2010. Study on panoramic image stitching algorithm. In *Proceedings of the 2nd Pacific-Asia Conference on Circuits, Communications and Systems (PACCS)*. 417–420.

Wen-Yan Lin, Siying Liu, Y. Matsushita, Tian-Tsong Ng, and Loong-Fah Cheong. 2011. Smoothly varying affine stitching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 345–352.

Noel McKeegan. 2007. The Adidas intelligent football. http://www.gizmag.com/adidas-intelligent-football/8512/.

Tomohiro Ozawa, Kris M. Kitani, and Hideki Koike. 2012. Human-centric panoramic imaging stitching. In *Proceedings of the Augmented Human International Conferences Series (AH)*. 20:1–20:6.

Prozone. 2013. Prozone Sports – Introducing Prozone performance analysis products. http://www.prozonesports.com/products.html.

Simen Sægrov, Alexander Eichhorn, Jørgen Emerslund, Håkon Kvale Stensland, Carsten Griwodz, Dag Johansen, and Pål Halvorsen. 2012. Bagadus: An integrated system for soccer analysis (demo). In *Proceedings of the 6th International Conference on Distributed Smart Cameras (ICDSC)*.

Valter Di Salvo, Adam Collins, Barry McNeill, and Marco Cardinale. 2006. Validation of Prozone: A new video-based performance analysis system. *Int. J. Perform. Anal. Sport* 6, 1, 108–119.

Nils T. Siebel and Stephen J. Maybank. 2002. Fusion of multiple tracking algorithms for robust people tracking. In *Proceedings of the 7th European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science, vol. 2353, Springer-Verlag, Berlin Heidelberg, 373–387.

Stats. 2013. STATS—SportVU—Football/Soccer. http://www.sportvu.com/football.asp.

Yingen Xiong and Kari Pulli. 2009. Color correction for mobile panorama imaging. In *Proceedings of the 1st International Conference on Internet Multimedia Computing and Service (ICIMCS)*. 219–226.

Yingen Xiong and Kari Pulli. 2010. Fast panorama stitching for high-quality panoramic images on mobile phones. *IEEE Trans. Consumer Electron.* 56, 2.

Ming Xu, James Orwell, and Graetne Jones. 2004. Tracking football players with multiple cameras. In *Proceedings of the International Conference on Image Processing (ICIP)*. 2909–2912.

Sunghoon Choi Yongduek, Sunghoon Choi, Yongduek Seo, Hyunwoo Kim, and Ki sang Hong. 1997. Where are the ball and players? Soccer game analysis with color-based tracking and image mosaick. In *Proceedings of the 9th International Conference on Image Analysis and Processing (ICIAP)*. Lecture Notes in Computer Science, vol. 1311, Springer-Varlag, Berlin Heidelberg, 196–203.

Zhengyou Zhang. 1999. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV)*. 666–673.

Z. Zivkovic. 2004. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*. 28–31. Vol. 2.

Zoran Zivkovic and Ferdinand van der Heijden. 2006. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recog. Lett.* 27, 7, 773–780.

ZXY. 2013. ZXY Sport Tracking. http://www.zxy.no/.

Received May 2013; revised August 2013; accepted October 2013

Appendix C

[Journal] Processing Panorama Video in Real-Time

[Authors:] H. K. Stensland, **V. R. Gaddam**, M. Tennøe, E. O. Helgedagsrud, M. Næss, H. K. Alstad, C. Griwodz, P. Halvorsen, and D. Johansen
[Published:] International Journal of Semantic Computing (IJSC), 2014

International Journal of Semantic Computing
Vol. 8, No. 2 (2014) 209–227

DOI: 10.1142/S1793351X14400054

World Scientific
www.worldscientific.com

Processing Panorama Video in Real-time

Håkon Kvale Stensland*, Vamsidhar Reddy Gaddam,
Marius Tennøe, Espen Helgedagsrud, Mikkel Næss,
Henrik Kjus Alstad, Carsten Griwodz and Pål Halvorsen
University of Oslo/Simula Research Laboratory
Oslo, Norway
**haakonks@ifi.uio.no*

Dag Johansen
University of Tromsø
Tromsø, Norway

There are many scenarios where high resolution, wide field of view video is useful. Such panorama video may be generated using camera arrays where the feeds from multiple cameras pointing at different parts of the captured area are stitched together. However, processing the different steps of a panorama video pipeline in real-time is challenging due to the high data rates and the stringent timeliness requirements. In our research, we use panorama video in a sport analysis system called Bagadus. This system is deployed at Alfheim stadium in Tromsø, and due to live usage, the video events must be generated in real-time. In this paper, we describe our real-time panorama system built using a low-cost CCD HD video camera array. We describe how we have implemented different components and evaluated alternatives. The performance results from experiments ran on commodity hardware with and without co-processors like graphics processing units (GPUs) show that the entire pipeline is able to run in real-time.

Keywords: Real-time panorama video; system integration; camera array.

1. Introduction

A wide field of view (panoramic) image or video is often used in applications like surveillance, navigation, scenic views, educational exhibits and sports analysis. Here, video feeds are often captured using multiple cameras capturing slightly overlapping areas, and the frames are processed and stitched into a single unbroken frame of the whole surrounding region. To prepare the individual frames for stitching and finally generating the panorama frame, each individual frame must be processed for barrel distortion, rotated to have the same angle, warped to the same plane and corrected for color differences. Then, the frames are stitched to one large panorama image, where the stitch operation also includes searching for the best possible seam in the overlapping areas to avoid seams through objects of interest in the video. Finally, the panorama frame is encoded to save storage space and transfer bandwidth and,

written to disk. As several of these steps include direct pixel manipulation and movement of large amounts of data, the described process is very resource hungry.

In [30], we described our implementation of a real-time panorama video pipeline for an arena sports application called Bagadus [11, 28], and this is an extended version providing more details. In our panorama setup, we use a static array of low-cost CCD HD video cameras, each pointing at a different direction, to capture the wide field of view of the arena. These different views are slightly overlapped in order to facilitate the stitching of these videos to form the panoramic video. Several similar non-real-time stitching systems exist (e.g. [23]), and a simple non-real-time version of this system has earlier been described and demonstrated at the functional level [11, 26]. Our initial prototype is the first sports application to successfully integrate per-athlete sensors [17], an expert annotation system [16] and a video system, but due to the non-real-time stitching, the panorama video was only available to the coaches some time after a game. The first prototype also did not use any form of color correction or dynamic seam detection. Hence, the static seam did not take into account moving objects (such as players), and the seam was therefore very visible.

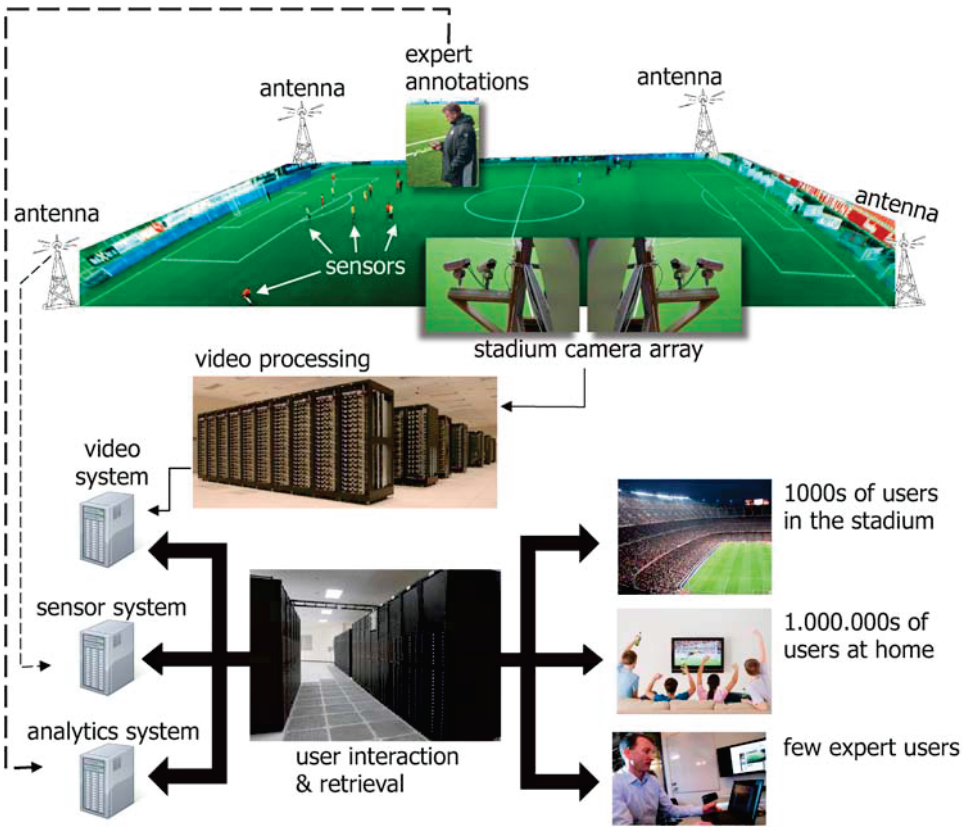


Fig. 1. Overall system architecture.

However, new requirements like real-time performance and better visual quality have resulted in a new and improved pipeline. Using our new real-time pipeline, such systems can be used during the game. A brief overview of the architecture and interaction of the different components is given in Fig. 1. In this paper we will focus on the details of the whole pipeline from capturing images from the cameras via various corrections steps for panorama generation to encoding and storage of both the panorama video and the individual camera streams on disks. We describe how we have evaluated different implementation alternatives (both algorithms and implementation options), and we benchmark the performance with and without using graphics processing units (GPUs) as an co-processors. We evaluate each individual component, and, we show how the entire pipeline is able to run in real-time on a low-cost 6-core machine with a GPU, i.e. moving the 1 frame per second (fps) system to 30 fps enabling game analysis during the ongoing event.

The remainder of the paper is structured as follows. We give a brief overview of the basic idea of our system in Sec. 2, and then we analyze the state of the art in Sec. 3 to see if systems exist that meet our requirements. Then, we describe and evaluate our real-time panorama video pipeline in Sec. 4. Various aspect of the system are discussed in Sec. 5 before we finally conclude the paper in Sec. 6.

2. Our Sports Analysis Systems

Today, a large number of (elite) sports clubs spend a large amount of resources to analyze their game performance, either manually or using one of the many existing analytics tools. For example, in the area of soccer, several systems enable trainers and coaches to analyze the gameplay in order to improve the performance. For instance, Interplay-sports, ProZone, STATS SportVU Tracking Technology and Camargus provide very nice video technology infrastructures. These systems can present player statistics, including speed profiles, accumulated distances, fatigue, fitness graphs and coverage maps using different charts, 3D graphics and animations. Thus, there exist several tools for soccer analysis. However, to the best of our knowledge, there does not exist a system that fully integrates all desired features in real-time, and existing systems still require manual work moving data between different components. In this respect, we have presented Bagadus [11, 26], which integrates a camera array video capture system with a sensor-based sport tracking system for player statistics and a system for human expert annotations. Our system allows the game analytics to automatically playout a tagged game event or to extract a video of events extracted from the statistical player data. This means that we for example can query for all sprints faster than X or all situations where a player is in the center circle. Using the exact player position provided by sensors, a trainer can also follow individuals or groups of players, where the videos are presented either using a stitched panorama view of the entire field or by (manually or automatically) switching between the different camera views. Our prototype is currently deployed at an elite club stadium. We use a dataset captured at a premier league game to experiment and to perform

benchmarks on our system. In previous versions of the system, the panorama video had to be generated offline, and it had static seams [11]. For comparison with the new pipeline presented in Sec. 4, we next present the camera setup and the old pipeline.

2.1. *Camera setup*

To record the high resolution video of the entire soccer field, we have installed a camera array consisting of four Basler industry cameras with a 1/3-inch image sensors supporting 30 fps at a resolution of 1280×960. The cameras are synchronized by an external trigger signal in order to enable a video stitching process that produces a panorama video picture. The cameras are mounted close to the middle line (see Fig. 2), i.e. under the roof of the stadium covering the spectator area approximately 10 meters from the side line and 10 meters above the ground. With a 3.5 mm wide-angle lens, each camera covers a field-of-view of about 68 degrees, and the full field with sufficient overlap to identify common features necessary for camera calibration and stitching, is achieved using the four cameras. Calibration is done via a classic chessboard pattern [33].

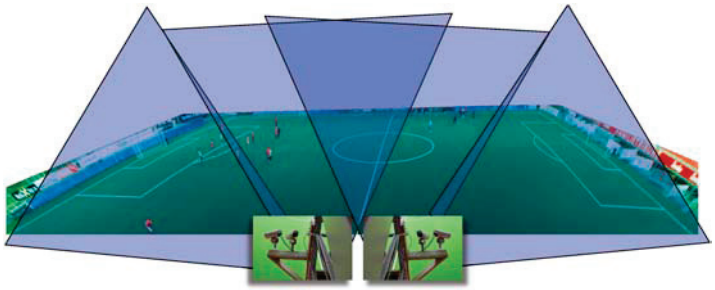


Fig. 2. Camera setup at the stadium.

2.2. *The offline, static stitching pipeline*

Our first prototype focused on integrating the different subsystems. We therefore did not put large efforts into real-time performance resulting in an unoptimized, offline panorama video pipeline that combined images from multiple, trigger-synchronized cameras as described above. The general steps in this stitching pipeline are: (1) correct the images for lens distortion in the outer parts of the frame due to a wide-angle fish-eye lens; (2) rotate and morph the images into the panorama perspective caused by different positions covering different areas of the field; (3) rotate and stitch the video images into a panorama image; and (4) encode and store the stitched video to persistent storage. Several implementations were tested for the stitching operation such as the OpenCV planar projection, cylindrical projection and spherical projection algorithms, but due to the processing performance and quality of the output image, the used solution is a homography based algorithm.

Fig. 3. The homography-based panorama image stitched from four cameras.

The first step before executing the pipeline, is to find corresponding pixel points in order to compute the homography between the camera planes [12], i.e. the head camera plane and the remaining camera planes. When the homography is calculated, the image can be warped (step 2) in order to fit the plane of the second image. The images must be padded to have the same size, and the seams for the stitching must be found in the overlapping regions (our first pipeline used static seams). Figure 3 shows the four rotated, wrapped and stitched images. The whole process of stitching the images is described in [28]. We also calculate the homography between the sensor data plane and the camera planes to find the mapping between sensor data coordinates and pixel positions.

As can be seen in the figure, the picture is not perfect, but the main challenge is the high execution time. On an Intel Core i7-2600 @ 3.4 GHz and 8 GB memory machine, the stitching operation consumed 974 ms of CPU time to generate each 7000×960 pixel panorama image [11]. Taking into account that the target display rate is 30 fps, i.e. requiring a new panorama image every 33 ms, there are large performance issues that must be addressed in order to bring the panorama pipeline from a 1 fps system to a 30 fps system. However, the stitching operations can be parallelized and parts of it offloaded to external devices such as GPUs, which, as we will see in Sec. 4, results in a performance good enough for real-time, online processing and generation of a panorama video.

3. Related Work

Real-time panorama image stitching is becoming common. For example, many have proposed systems for panorama image stitching (e.g. [6, 14, 19–21]), and modern operating systems for smart phones like Apple iOS and Google Android support generation of panorama pictures in real-time. However, the definition of real-time is not necessarily the same for all applications, and in this case, real-time is similar to "within a second or two". For video, real-time has another meaning, and a panorama picture must be generated in the same speed as the display frame rate, e.g. every 33 ms for a 30 frame-per-second (fps) video.

One of these existing systems is Camargus [1]. The people developing this system claim to deliver high definition panorama video in real-time from a setup consisting of 16 cameras (ordered in an array), but since this is a commercial system, we have no insights to the details. Another example is Immersive Cockpit [29] which aims to generate a panorama for tele-immersive applications. They generate a stitched video

which capture a large field-of-view, but their main goal is not to give output with high visual quality. Although they are able to generate video at a frame rate of about 25 fps for 4 cameras, there are visual limitations to the system, which makes the system not well suited for our scenario.

Moreover, Baudisch *et al.* [5] present an application for creating panoramic images, but the system is highly dependent on user input. Their definition of real time is "panorama construction that offers a real-time preview of the panorama while shooting", but they are only able to produce about 4 fps (far below our 30 fps requirement). A system similar to ours is presented in [4], which computes stitch-maps on a GPU, but the presented system produces low resolution images (and is limited to two cameras). The performance is within our real-time requirement, but the timings are based on the assumption that the user accepts a lower quality image than the cameras can produce.

Haynes [3] describes a system by the Content Interface Corporation that creates ultra high resolution videos. The Omnicam system from the Fascinate [2, 27] project also produces high resolution videos. However, both these systems use expensive and specialized hardware. The system described in [3] also makes use of static stitching. A system for creating panoramic videos from already existing video clips is presented in [7], but it does not manage to create panorama videos within our definition of real-time. As far as we know, the same issue of real-time is also present in [5, 13, 23, 31].

In summary, existing systems (e.g. [7, 13, 23, 29, 31]) do not meet our demand of being able to generate the video in real-time, and commercial systems (e.g. [1, 3]) as well as the systems presented in [2, 27] do often not fit into our goal to create a system with limited resource demands. The system presented in [4] is similar to our system, but we require high quality results from processing a minimum of four cameras streams at 30 fps. Thus, due to the lack of a low-cost implementations fulfilling our demands, we have implemented our own panorama video processing pipeline which utilize processing resources on both the CPU and GPU.

4. A Real-Time Panorama Stitcher

In this paper, we do not focus on selecting the best algorithms etc., as this is mostly covered in [11]. The focus here is to describe the panorama pipeline and how the different components in the pipeline are implemented in order to run in real-time. We will also point out various performance trade-offs.

As depicted in Fig. 4, the new and improved panorama stitcher pipeline is separated into two main parts: one part running on the CPU, and the other running on a GPU using the CUDA framework. The decision of using a GPU as part of the pipeline was due to the potential high performance and the parallel nature of the workload. The decision of using the GPU for the pipeline has affected the architecture to a large degree. Unless otherwise stated (we have tested several CPUs and GPUs), our test machine for the new pipeline is an Intel Core i7-3930K, i.e. a 6-core

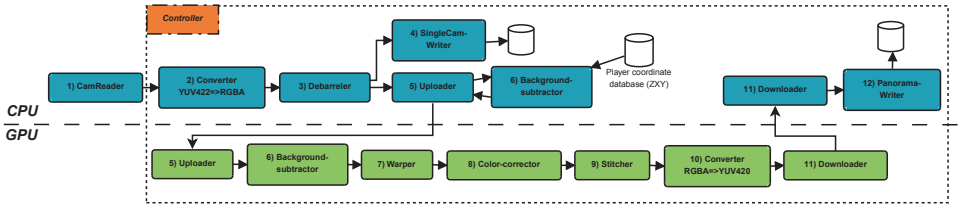


Fig. 4. Panorama stitcher pipeline architecture. The orange and blue components run in the CPU and the green components run on the GPU.

processor based on the Sandy Bridge-E architecture, with 32 GB RAM and an Nvidia GeForce GTX 680 GPU based on the GK104 Kepler architecture.

4.1. *The Controller module*

The single-threaded Controller is responsible for initializing the pipeline, synchronizing the different modules, handling global errors and frame drops, and transferring data between the different modules. After initialization, it will wait for and get the next set of frames from the camera reader (CamReader) module (see below). Next, it will control the transfers of data from the output buffers of module N to the input buffers of module $N + 1$. This is done primarily by pointer swapping, and with memory copies as an alternative. It then signals all modules to process the new input and waits for them to finish processing. Next, the controller continues looping by waiting for the next set of frames from the reader. Another important task of the Controller is to check the execution speed. If an earlier step in the pipeline runs too slow, and one or more frames has been lost from the cameras, the controller will tell the modules in the pipeline to skip the delayed or dropped frame, and reuse the previous frame.

4.2. *The CamReader module*

The CamReader module is responsible for retrieving frames from the cameras. It consists of one dedicated reader thread per camera. Each of the threads will wait for the next frame, and then write the retrieved frame to a output buffer, overwriting the previous frame. The cameras provide a single frame in YUV 4:2:2 format, and the retrieval rate of frames in the CamReader is what determines the real time threshold for the rest of the pipeline. As described above, the camera shutter synchronization is controlled by an external trigger box, and in our current configuration, the cameras deliver a frame rate of 30 fps, i.e. the real-time threshold and the CamReader processing time are thus 33 ms.

4.3. *The Converter module*

The CamReader module outputs frames in YUV 4:2:2 format. However, the stitching pipeline requires RGBA internally for processing, and the system therefore converts frames from YUV 4:2:2 to RGBA. This is handled by the Converter module using

ffmpeg and *swscale*. The processing time for these conversions on the CPU, as seen later in Fig. 11, is well below the real-time requirement, so this operation can run as a single thread.

4.4. *The Debarreler module*

Due to the wide angle lenses used with our cameras in order to capture the entire field, the images delivered are suffering from barrel distortion which needs to be corrected. We found the performance of the existing debarreling implementation in the old stitching pipeline to perform fast enough. The Debarreler module is therefore still based on OpenCVs debarreling function, using nearest neighbor interpolation, and is executing as a dedicated thread per camera.

4.5. *The SingleCamWriter module*

In addition to storing the stitched panorama video, we also want to store the video from the separate cameras. This storage operation is done by the SingleCamWriter, which is running as a dedicated thread per camera. As we can see in [11], storing the videos as raw data proved to be impractical due to the size of uncompressed raw data. The different CamWriter modules (here SingleCamWriter) therefore encode and compress frames into 3 seconds long H.264 files, which proved to be very efficient. Due to the use of H.264, every SingleCamWriter thread starts by converting from RGBA to YUV 4:2:0, which is the required input format by the x264 encoder. The threads then encode the frames and write the results to disk.

4.6. *The Uploader module*

Due to the large potential of parallelizing the panorama workload and the high computing power of modern GPUs, large parts of our pipeline run on a GPU. We therefore need to transfer data from the CPU to the GPU, i.e. a task performed by the Uploader module. In addition, the Uploader is also responsible for executing the CPU part of the BackgroundSubtractor (BGS) module (see Sec. 4.7). The Uploader consists of a single CPU thread, that first runs the player pixel lookup creation needed by the BGS. Next, it transfers the current RGBA frames and the corresponding player pixel maps from the CPU to the GPU. This is done by use of double buffering and asynchronous transfers.

4.7. *The BackgroundSubtractor module*

Background subtraction is the process of determining which pixels of a video that belong to the foreground and which pixels that belong to the background. The BackgroundSubtractor module, running on the GPU, generates a foreground mask (for moving objects like players) that is later used in the Stitcher module later to avoid seams in the players. Our BackgroundSubtractor can run like traditional systems searching the entire image for foreground objects. However, we can also

exploit information gained by the tight integration with the player sensor system. In this respect, through the sensor system, we know the player coordinates which can be used to improve both performance and precision of the module. By first retrieving player coordinates for a frame, we can then create a player pixel lookup map, where we only set the players pixels, including a safety margin, to 1. The creation of these lookup maps are executed on the CPU as part of the Uploader. The BGS on GPU then uses this lookup map to only process pixels close to a player, which reduces the GPU kernel processing times, from 811.793 microseconds to 327.576 microseconds on average on a GeForce GTX 680. When run in a pipelined fashion, the processing delay caused by the lookup map creation is also eliminated. The sensor system coordinates are retrieved by a dedicated slave thread that continuously polls the sensor system database for new samples.

Even though we enhance the background subtraction with sensor data input, there are several implementation alternatives. When determining which algorithm to implement, we evaluated two alternatives: Zivkovic [34, 35] and KaewTraKulPong [18]. Even though the CPU implementation was slower (see Fig. 5), Zivkovic provided the best visual results, and was therefore selected for further modification. Furthermore, the Zivkovic algorithm proved to be fast enough when modified with input from the sensor system data. The GPU implementation, based on [25], proved to be even faster, and the final performance numbers for a single camera stream can be seen in Fig. 5. A visual comparison of the unmodified Zivkovic implementation and the sensor system-modified version is seen in Fig. 6 where the sensor coordinate modification reduce the noise as seen in the upper parts of the pictures.

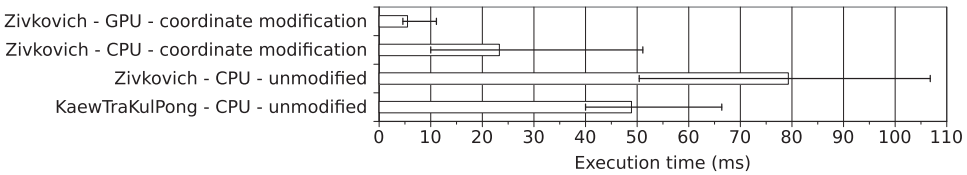
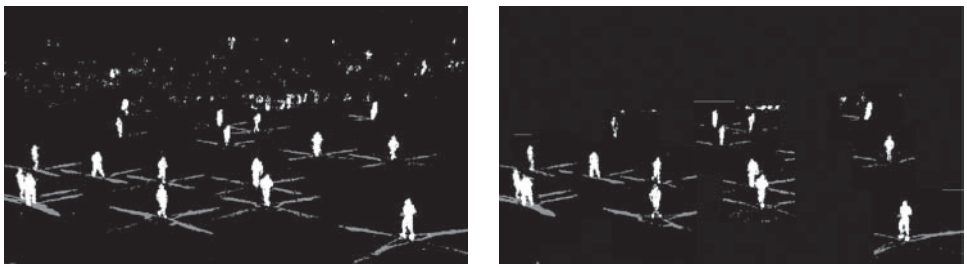


Fig. 5. Execution time of alternative algorithms for the BackgroundSubtractor module (1 camera stream).



(a) Unmodified Zivkovic (b) Player sensor data modification of Zivkovic

Fig. 6. Background subtraction comparison.

4.8. *The Warper module*

The Warper module is responsible for warping the camera frames to fit the stitched panorama image. By warping we mean twisting, rotating and skewing the images to fit the common panorama plane. Like we have seen from the old pipeline, this is necessary because the stitcher assumes that its input images are perfectly warped and aligned to be stitched to a large panorama. Executing on the GPU, the Warper also warps the foreground masks provided by the BGS module. This is because the Stitcher module at a later point will use the masks and therefore expects the masks to fit perfectly to the corresponding warped camera frames. Here, we use the Nvidia Performance Primitives library (NPP) for an optimized implementation.

4.9. *The Color-corrector module*

When recording frames from several different cameras pointing in different direction, it is nearly impossible to calibrate the cameras to output the exact same colors due to the different lighting conditions. This means that, to generate the best panorama videos, we need to correct the colors of all the frames to remove color disparities. In our panorama pipeline, this is done by the Color-corrector module running on the GPU.

We choose to do the color correction after warping the images. The reason for this is that locating the overlapping regions is easier with aligned images, and the overlap is also needed when stitching the images together. This algorithm is executed on the GPU, enabling fast color correction within our pipeline. The implementation is based on the algorithm presented in [32], but have some minor modifications. We calculate the color differences between the images for every single set of frames delivered from the cameras. Currently, we color-correct each image in a sequence, meaning that each image is corrected according to the overlapping frame to the left. The algorithm implemented is easy to parallelize and does not make use of pixel to pixel mapping which makes it well suited for our scenario. Figure 7 shows a comparison between running the algorithm on the CPU and on a GPU.

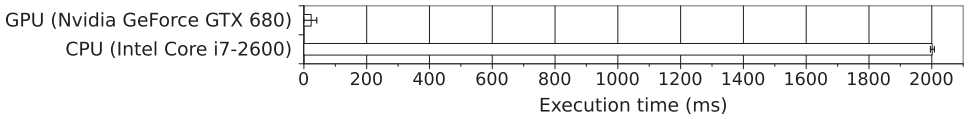


Fig. 7. Execution time of color correction.

4.10. *The Stitcher module*

Like in the old pipeline, we use a homography based stitcher where we simply create seams between the overlapping camera frames, and then copy pixels from the images based on these seams. These frames need to follow the same homography, which is

why they have to be warped. Our old pipeline used static cuts for seams, which means that a fixed rectangular area from each frame is copied directly to the output frame. Static cut panoramas are faster, but can introduce graphical errors in the seam area, especially when there are movement in the scene (illustrated in Fig. 8).

To make a better seam with a better visual result, we therefore introduce a dynamic cut stitcher instead of the old static cut. The dynamic cut stitcher creates seams by first creating a rectangle of adjustable width over the static seam area. Then, it treats all pixels within the seam area as graph nodes. The graph is directed from the bottom to the top in such a way that each pixel points to the three adjacent ones above (left and right-most pixels only point to the two available). Each of these edge's weight are calculated by using a custom function that compares the absolute color difference between the corresponding pixel in each of the two frames we are trying to stitch. The weight function also checks the foreground masks from the BGS module to see if any player is in the pixel, and if so it adds a large weight to the node. In effect, both these steps will make edges between nodes where the colors differs and players are present have much larger weights. We then run the Dijkstra graph

Fig. 8. Stitcher comparison — improving the visual quality with dynamic seams and color correction. The first image shows the original stitch [11] with a fixed cut stitch with a straight vertical seam. The middle image shows a dynamic stitch with no color correction. The embedded thumbnail shows the seam. The bottom image shows a dynamic stitch with color correction, i.e. resulting in that the seam is no longer visible.

algorithm on the graph to create a minimal cost route from the start of the offset at the bottom of the image to the end at the top. Since our path is directed upwards, we can only move up or diagonally from each node, and we will only get one node per horizontal position. By looping through the path, we therefore get our new cut offsets by adding the node's horizontal position to the base offset.

An illustration of how the final seam looks can be seen in bottom image in Fig. 8, where the seams without and with color correction are shown in the embedded thumbnails. Timings for the dynamic stitching module can be seen in Fig. 9. The CPU version is currently slightly faster than our GPU version (as searches and branches often are more efficient on traditional CPUs), but further optimization of the CUDA code will likely improve this GPU performance. Note that the min and max numbers for the GPU are skewed by frames dropping (no processing), and the initial run being slower.

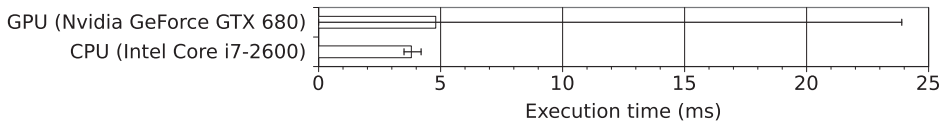


Fig. 9. Execution time for dynamic stitching.

4.11. *The YuvConverter module*

Before storing the stitched panorama frames, we need to convert back from RGBA to YUV 4:2:0 for the H.264 encoder, just as in the SingleCamWriter module. However, due to the size of the output panorama, this conversion is not fast enough on the CPU, even with the highly optimize *swscale*. This module is therefore implemented on the GPU. In Fig. 10, we can see the performance of the CPU based implementation versus the optimized GPU based version.

Nvidia NPP contains several conversion primitives, but not a direct conversion from RGBA to YUV 4:2:0. The GPU based version is therefore first using NPP to convert from RGBA to YUV 4:4:4, and then a self written CUDA code to convert from YUV 4:4:4 to YUV 4:2:0.

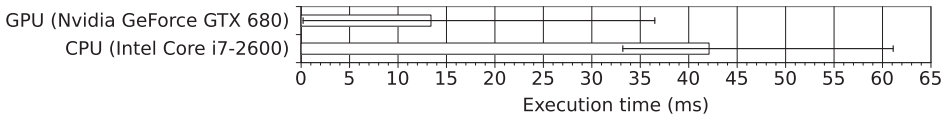


Fig. 10. Execution time for RGBA to YUV 4:2:0 conversion.

4.12. *The Downloader module*

Before we can write the stitched panorama frames to disk, we need to transfer it back to the CPU, which is done by the Downloader module. It runs as a single CPU thread

that copies a frame synchronously to the CPU. We could have implemented the Downloader as an asynchronous transfer with double buffering like the Uploader, but since the performance as seen in Fig. 11 is very good, this is left as future work.

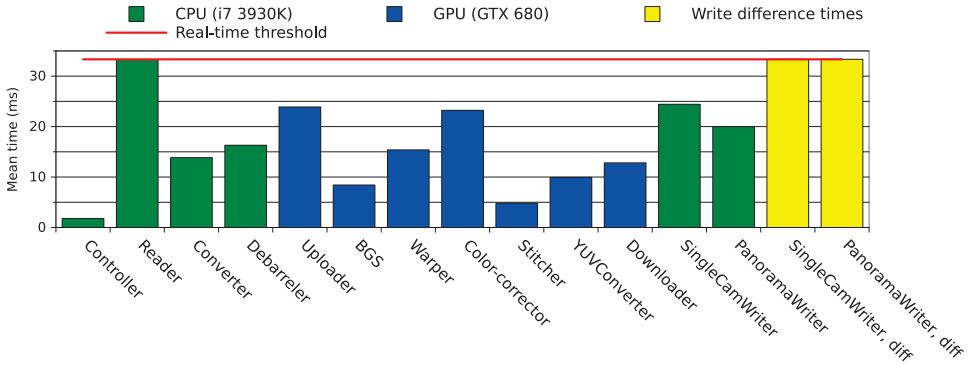


Fig. 11. Improved stitching pipeline performance, module overview (Nvidia GeForce GTX 680 and Intel Core i7-3930K).

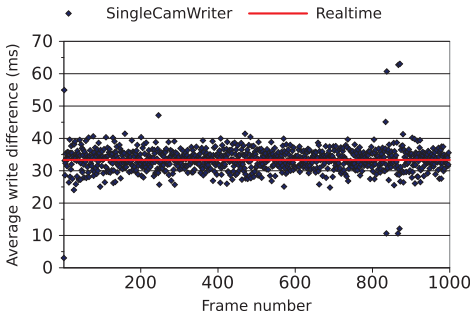
4.13. *The PanoramaWriter module*

The last module, executing on the CPU, is the Writer that writes the panorama frames to disk. The conversion from RGBA to YUV has already been done on the GPU, so the only steps the PanoramaWriter needs to follow, is to first encode the input frame to H.264, and then write the result to disk as three second H.264 video files.

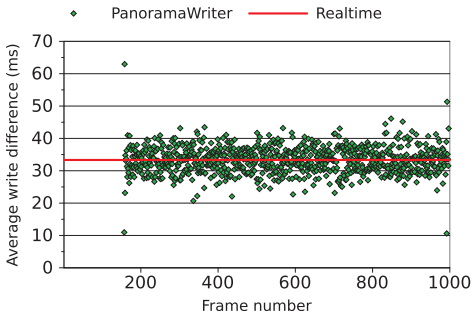
4.14. *Pipeline performance*

In order to evaluate the performance of our pipeline, we used an off-the-shelf PC with an Intel Core i7-3930K processor and an nVidia GeForce GTX 680 GPU. We have benchmarked each individual component and the pipeline as a whole capturing, processing and storing 1000 frames from the cameras.

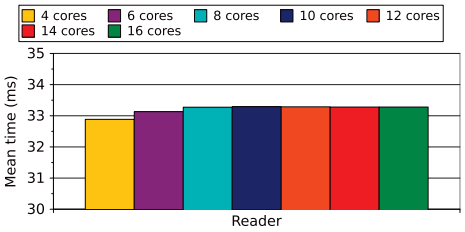
In the initial pipeline [11], the main bottleneck was the panorama creation (warping and stitching). This operation alone used *974 ms per frame*. As shown by the breakdown into individual components' performance in Fig. 11, the new pipeline has been greatly improved. Note that all individual components run in real-time running concurrently on the same set of hardware. Adding all these, however, gives times far larger than 33 ms. The reason why the pipeline is still running in real-time is because several frames are processed in parallel. Note here that all CUDA kernels are executing at the same time on a single GPU, so the performance of all GPU modules are affected by the performance of the other GPU modules. On earlier GPUs like the GTX 280, this was not allowed, but concurrent CUDA kernel execution was introduced in the Fermi architecture [24] (GTX 480 and above). Thus, since the



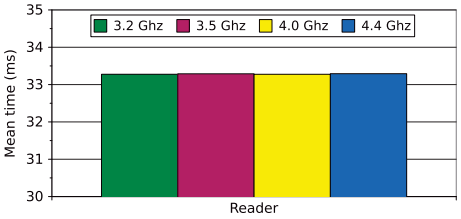
(a) SingleCamWriter inter-departure time



(b) Inter-departure time of PanoramaWriter. The inter-departure frames delayed by five seconds due to the two second safety buffer in CPU/GPU transfer and the three second delay of the sensor data



(c) Core count scalability



(d) Core frequency scalability

Fig. 12. Inter-departure time of frames when running the entire pipeline. In a real-time scenario, the output rate should follow the input rate (given here by the trigger box) at 30 fps (33 ms).

Controller module schedules the other modules according to the input rate of 30 fps, the amount of resources are sufficient for real-time execution.

For the pipeline to be real-time, the output rate should follow the input rate, i.e. deliver all output frames (both 4 single camera and 1 panorama) at 30 fps. Thus, to give an idea of how often a frame is written to file, Fig. 12 shows individual and average frame inter-departure rates. The figures show the time difference between consecutive writes for the generated panorama as well as for the individual camera streams. Operating system calls, interrupts and disk accesses will most likely cause small spikes in the write times (as seen in the scatter plot in Figs. 12(a) and 12(b)), but as long as the average times are equal to the real-time threshold, the pipeline can be considered real-time. As we can see in Figs. 11, 12(c) and 12(d), the average frame inter-arrival time (Reader) is equal to the average frame inter-departure time (both SingleCamWriter and PanoramaWriter). This is also the case testing other CPU frequencies and number of available cores. Thus, the pipeline runs in real-time.

As said above and seen in Figs. 12(a) and 12(b), there is a small latency in the panorama pipeline compared to writing the single cameras immediately. The total panorama pipeline latency, i.e. the end to end frame delay from read from the camera

until written to disk, is equal to 33 ms per sequential module (as long as the modules perform fast enough) plus a 5 second input buffer (the input buffer is because the sensor system has at least 3 second latency before the data is ready for use, and we have added a 2 second buffer for GPU processing). The 33 ms are caused by the camera frame rate of 30 fps, meaning that even though a module may finish before the threshold, the Controller will make it wait until the next set of frames arrive before it is signaled to re-execute. This means that the pipeline latency is 5.33 seconds per frame on average.

5. Discussion

Our soccer analysis application integrates a sensor system, soccer analytics annotations and video processing of a video camera array. There already exist several components that can be used, and we have investigated several alternatives in our research. Our first prototype aimed at full integration at the system level, rather than being optimized for performance. In this paper, however, our challenge has been of an order of magnitude harder by making the system run in real-time on low-cost, off-the-shelf hardware.

The new real-time capability also enables future enhancements with respect to functionality. For example, several systems have already shown the ability to serve available panorama video to the masses [13, 23], and by also generating the panorama video live, the audience can mark and follow particular players and events. Furthermore, ongoing work also include machine learning of sensor and video data to extract player and team statistics for evaluation of physical and tactical performance. We can also use this information to make video playlists [15] automatically giving a video summary of extracted events. Due to limited availability of resources, we have not been able to test our system with more cameras or higher resolution cameras. However, to still get an impression of the scalability capabilities of our pipeline, we have performed several benchmarks changing the number of available cores, the processor clock frequency and GPUs with different architecture and compute resources. Figure 13[a] shows the results changing the number of available cores that can process the many concurrent threads in the CPU-part of pipeline (Fig. 12(c) shows that the pipeline is still in real-time). As we can observe from the figure, every component runs in real-time using more than 4 cores, and the pipeline as a whole using 8 or more cores. Furthermore, the CPU pipeline contains a large, but configurable number of threads (86 in the current setup), and due to the many threads of the embarrassingly parallel workload, the pipeline seems to scale well with the number of available cores.

Similar conclusions can be drawn from Fig. 14 where the processing time is reduced with a higher processor clock frequency, i.e. the pipeline runs in real-time already at 3.2 GHz, and there is almost a linear scaling with CPU frequency (Fig. 12(d)

[a]Note that this experiment was run on a machine with more available cores (16), each at a lower clock frequency (2.0 GHz) compared to the machine installed at the stadium which was used for all other tests.

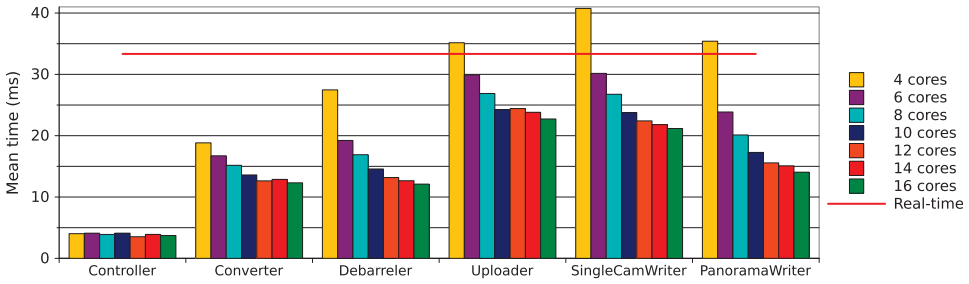


Fig. 13. Core count scalability.

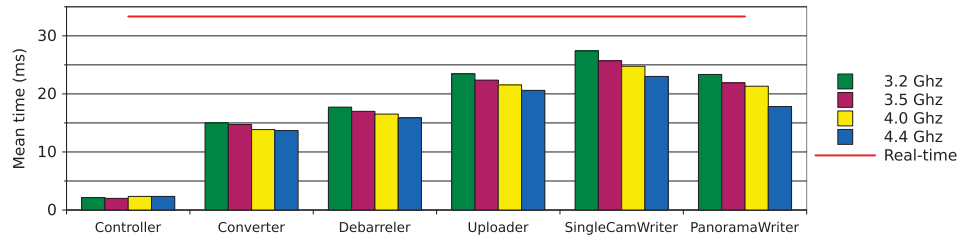


Fig. 14. CPU frequency scalability.

shows that the pipeline is still in real-time). Especially the H.264 encoder scales very good when scaling the CPU frequency. With respect to the GPU-part of the pipeline, Fig. 15 plots the processing times using different GPUs.

The high-end GPUs GTX 480 and above (Compute 2.x and higher) all achieve real-time performance on the current setup. The GTX 280 is only compute 1.3 which does not support the concurrent CUDA kernel execution in the Fermi architecture [24], and the performance is therefore slower than real-time. As expected, more powerful GPUs reduce the processing time. For now, one GPU fulfills our real-time requirement, we did therefore not experiment with multiple GPUs, but the GPU processing power can easily be increased by adding multiple cards. Thus, based on these results, we believe that our pipeline easily can be scaled up to both higher numbers of cameras and higher resolution cameras.

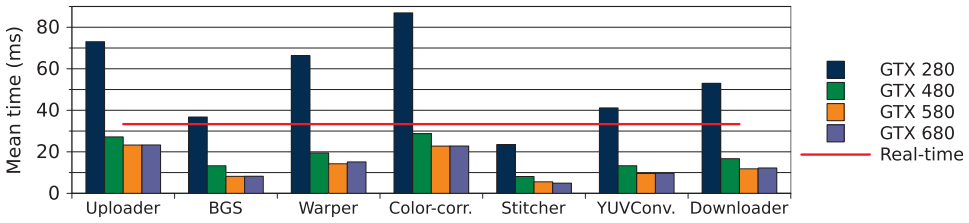


Fig. 15. GPU comparison.

6. Conclusions

In this paper, we have presented a prototype of a real-time panorama video processing system. The panorama prototype is used as a sub-component in a real sport analysis system where the target is automatic processing and retrieval of events at a sports arena. We have described the pipeline in detail, where we use both the CPU and a GPU for offloading. Furthermore, we have provided experimental results which prove the real-time properties of the pipeline on a low-cost 6-core machine with a commodity GPU, both for each component and the combination of the different components forming the entire pipeline.

The entire system is under constant development, and new functionality is added all the time, e.g. camera-array-wide synchronized automatic exposure [8], interactive zoom and panning [9, 10], extended search functionality [22] and scaling the panorama system up to a higher number of cameras and to higher resolution cameras [9]. So far, the pipeline scales nicely with the CPU frequencies, number of cores and GPU resources. We plan to use PCI Express-based interconnect technology from Dolphin Interconnect Solutions for low latency and fast data transfers between machines. Experimental results in this respect is though ongoing work and out of scope in this paper.

Acknowledgments

This work has been performed in the context of the *iAD* centre for Research-based Innovation (project number 174867) funded by the Norwegian Research Council. Furthermore, the authors also acknowledge the support given by Kai-Even Nilssen for practical assistance with respect to the installation at Alfheim stadium.

References

[1] Camargus — Premium Stadium Video Technology Infrastructure, http://www.camargus.com/. [Online; accessed 01-March-2013.]

[2] Live ultra-high resolution panoramic video, http://www.fascinate-project.eu/index.php/tech-section/hi-res-video/. [Online; accessed 04-March-2012.]

[3] Software stitches 5k videos into huge panoramic video walls, in real time, http://www.sixteen-nine.net/2012/10/22/software-stitches-5k-videos-huge-panoramic-video-walls-real-time/, 2012. [Online; accessed 05-March-2012.]

[4] M. Adam, C. Jung, S. Roth and G. Brunnett, Real-time stereo-image stitching using GPU-based belief propagation, 2009, pp. 215–224.

[5] P. Baudisch, D. Tan, D. Steedly, E. Rudolph, M. Uyttendaele, C. Pal and R. Szeliski, An exploration of user interface designs for real-time panoramic photography, *Australasian Journal of Information Systems* **13**(2) (2006) 151.

[6] M. Brown and D. G. Lowe, Automatic panoramic image stitching using invariant features, *International Journal of Computer Vision* **74**(1) (2007) 59–73.

[7] D.-Y. Chen, M.-C. Ho and M. Ouhyoung, Videovr: A real-time system for automatically constructing panoramic images from video clips, in *Proc. CAPTECH*, 1998, pp. 140–143.

[8] V. R. Gaddam, C. Griwodz and P. Halvorsen, Automatic exposure for panoramic systems in uncontrolled lighting conditions: A football stadium case study, in *Proc. SPIE/IS&T Electronic Imaging — the Engineering Reality of Virtual Reality*, 2014, pp. 90120C–90120C-9.

[9] V. R. Gaddam, R. Langseth, S. Ljødal, P. Gurdjos, V. Charvillat, C. Griwodz and P. Halvorsen, Interactive zoom and panning from live panoramic video, in *Proc. NOSSDAV*, 2014, pp. 19:19–19:24.

[10] V. R. Gaddam, R. Langseth, H. K. Stensland, P. Gurdjos, V. Charvillat, C. Griwodz, D. Johansen and P. Halvorsen, Be your own cameraman: Real-time support for zooming and panning into stored and live panoramic video, in *Proc. MMSys*, 2014, pp. 168–171.

[11] P. Halvorsen, S. Sægrov, A. Mortensen, D. K. C. Kristensen, A. Eichhorn, M. Stenhaug, S. Dahl, H. K. Stensland, V. R. Gaddam, C. Griwodz and D. Johansen, Bagadus: An integrated system for arena sports analytics — a soccer case study, in *Proc. MMSys*, 2013, pp. 48–59.

[12] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision* (Cambridge University Press, 2003).

[13] K. Huguenin, A.-M. Kermarrec, K. Kloudas and F. Taiani, Content and geographical locality in user-generated content sharing systems, in *Proc. NOSSDAV*, 2012, pp. 77–82.

[14] J. Jia and C.-K. Tang, Image stitching using structure deformation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **30**(4) (2008) 617–631.

[15] D. Johansen, H. Johansen, T. Aarflot, J. Hurley, Å. Kvalnes, C. Gurrin, S. Sav, B. Olstad, E. Aaberg, T. Endestad, H. Riiser, C. Griwodz and P. Halvorsen, DAVVI: A prototype for the next generation multimedia entertainment platform, in *Proc. ACM MM*, 2009, pp. 989–990.

[16] D. Johansen, M. Stenhaug, R. B. A. Hansen, A. Christensen and P.-M. Høgmo, Muithu: Smaller footprint, potentially larger imprint, in *Proceedings of the IEEE International Conference on Digital Information Management*, 2012, pp. 205–214.

[17] H. D. Johansen, S. A. Pettersen, P. Halvorsen and D. Johansen, Combining video and player telemetry for evidence-based decisions in soccer, in *Proceedings of the International Congress on Sports Science Research and Technology Support*, 2013, pp. 197–205.

[18] P. KaewTraKulPong and R. Bowden, An improved adaptive background mixture model for real-time tracking with shadow detection, in *Video-Based Surveillance Systems* (Springer, 2002), pp. 135–144.

[19] A. Levin, A. Zomet, S. Peleg and Y. Weiss, Seamless image stitching in the gradient domain, *Computer Vision — ECCV 2004*, 2004, pp. 377–389.

[20] Y. Li and L. Ma, A fast and robust image stitching algorithm, in *Proc. WCICA* **2** (2006) 9604–9608.

[21] A. Mills and G. Dudek, Image stitching with dynamic elements, *Image and Vision Computing* **27**(10) (2009) 1593–1602.

[22] A. Mortensen, V. R. Gaddam, H. K. Stensland, C. Griwodz, D. Johansen and P. Halvorsen, Automatic event extraction and video summaries from soccer games, in *Proc. MMSys*, 2014, pp. 176–179.

[23] O. A. Niamut, R. Kaiser, G. Kienast, A. Kochale, J. Spille, O. Schreer, J. R. Hidalgo, J.-F. Macq and B. Shirley, Towards a format-agnostic approach for production, delivery and rendering of immersive media, in *Proc. MMSys*, 2013, pp. 249–260.

[24] nVIDIA. Nvidia's next generation CUDA compute architecture: Fermi. http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf, 2010. [Online; accessed 08-March-2013].

[25] V. Pham, P. Vo, V. T. Hung *et al.*, GPU implementation of extended gaussian mixture model for background subtraction, in *IEEE International Conference on Computing*

and Communication Technologies, Research, Innovation and Vision for the Future, 2010, pp. 1–4.

[26] S. Sægrov, A. Eichhorn, J. Emerslund, H. K. Stensland, C. Griwodz, D. Johansen and P. Halvorsen, Bagadus: An integrated system for soccer analysis (demo), in *Proc. ICDSC*, 2012, pp. 1–2.

[27] O. Schreer, I. Feldmann, C. Weissig, P. Kauff and R. Schafer, Ultrahigh-resolution panoramic imaging for format-agnostic video production, in *Proceedings of the IEEE* **101**(1) (2013) 99–114.

[28] H. K. Stensland, V. R. Gaddam, M. Tennøe, E. Helgedagsrud, M. Næss, H. K. Alstad, A. Mortensen, R. Langseth, S. Ljødal, Ø. Landsverk, C. Griwodz, P. Halvorsen, M. Stenhaug and D. Johansen, Bagadus: An integrated real-time system for soccer analytics, *Transactions on Multimedia Computing, Communications and Applications* **10**(1s) (2014) 14:1–14:21.

[29] W.-K. Tang, T.-T. Wong and P.-A. Heng, A system for real-time panorama generation and display in tele-immersive applications, *IEEE Transactions on Multimedia* **7**(2) (2005) 280–292.

[30] M. Tennøe, E. Helgedagsrud, M. Næss, H. K. Alstad, V. R. Gaddam, H. K. Stensland, C. Griwodz, D. Johansen and P. Halvorsen, Efficient implementation and processing of a real-time panorama video pipeline, in *Proc. ISM*, 2013, pp. 76–83.

[31] C. Weissig, O. Schreer, P. Eisert and P. Kauff, The ultimate immersive experience: Panoramic 3d video acquisition, *Advances in Multimedia Modeling*, LNCS Vol. 7131 (Springer, 2012), pp. 671–681.

[32] Y. Xiong and K. Pulli, Color correction for mobile panorama imaging, in *Proc. ICIMCS*, 2009, pp. 219–226.

[33] Z. Zhang, Flexible camera calibration by viewing a plane from unknown orientations, in *Proceedings of the IEEE International Conference on Computer Vision*, 1999, pp. 666–673.

[34] Z. Zivkovic, Improved adaptive gaussian mixture model for background subtraction, in *Proc. ICPR* **2** (2004) 28–31.

[35] Z. Zivkovic and F. van der Heijden, Efficient adaptive density estimation per image pixel for the task of background subtraction, *Pattern Recognition Letters* **27**(7) (2006) 773–780.

Appendix D

[Journal] The Cameraman Operating My Virtual Camera is Artificial: Can the Machine Be as Good as a Human?

[Authors:] **V. R. Gaddam**, R. Eg, C. Griwodz, and P. Halvorsen
[Published:] ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP), 2015

The Cameraman Operating My Virtual Camera is Artificial: Can the Machine Be as Good as a Human?

VAMSIDHAR REDDY GADDAM, RAGNHILD EG, RAGNAR LANGSETH, CARSTEN GRIWODZ, and PÅL HALVORSEN, Simula Research Laboratory and University of Oslo

In this article, we argue that the energy spent in designing autonomous camera control systems is not spent in vain. We present a real-time virtual camera system that can create *smooth* camera motion. Similar systems are frequently benchmarked with the human operator as the best possible reference; however, we avoid a priori assumptions in our evaluations. Our main question is simply whether we can design algorithms to steer a virtual camera that can compete with the user experience for recordings from an expert operator with several years of experience? In this respect, we present two low-complexity servoing methods that are explored in two user studies. The results from the user studies give a promising answer to the question pursued. Furthermore, all components of the system meet the real-time requirements on commodity hardware. The growing capabilities of both hardware and network in mobile devices give us hope that this system can be deployed to mobile users in the near future. Moreover, the design of the presented system takes into account that services to concurrent users must be supported.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Video*

General Terms: Experimentation, Measurement, Performance

Additional Key Words and Phrases: Interactive immersion, panorama video, zoom, panning, real-time, visual servoing, virtual camera, quality of experience, user studies

ACM Reference Format:
Vamsidhar Reddy Gaddam, Ragnhild Eg, Ragnar Langseth, Carsten Griwodz, and Pål Halvorsen. 2015. The cameraman operating my virtual camera is artificial: Can the machine be as good as a human? ACM Trans. Multimedia Comput. Commun. Appl. 11, 4, Article 56 (April 2015), 20 pages.
DOI: http://dx.doi.org/10.1145/2744411

1. INTRODUCTION

Improvements in communications and processing power provide opportunities to explore groundbreaking systems in interactive immersive applications. For example, in scenarios like surveillance and sports, high-resolution wide field-of-view panoramic video has become popular. Stitched panorama videos generated from a camera array that covers an entire field can be used to support virtual views through zoom and pan operations. In turn, individual users can interactively control their own virtual camera. We have created a prototype system that provides an immersive experience using a soccer stadium as a case study. The strength of the system lies in complete automation of several steps that are currently considered to be superior when operated by a human. In this respect, a user can function as his or her own cameraman. However, in

This work has been performed in the context of the iAD center for Research-based Innovation (project number 174867) funded by the Norwegian Research Council.
Author's address: V. R. Gaddam; email: vamsidhg@ifi.uio.no.

ACM 1551-6857/2015/04-ART56 $15.00
DOI: http://dx.doi.org/10.1145/2744411

many situations it may be desirable to allow the system to automatically operate the cameras, for instance when following the ball or a particular player in a soccer scenario.

Most commonly seen broadcast is usually of high profile games. In such games, the financial resources can make several simultaneous high quality capture systems possible. However, most soccer games at different levels of profession do not qualify for such high production qualities. In such cases, there are typically 2-3 simultaneous video streams that are switched. Considering a few games,[1] it can be observed that the camera operator majorly works with pan-tilt-zoom operations for capturing the game from a single viewpoint. Thus, our main research question in this article is whether a machine-generated virtual camera can provide a viewing experience that is at least as good as a human-generated one, under similar conditions?

Many researchers have looked at similar challenges [Ariki et al. 2006; Carr et al. 2013; Chen and De Vleeschouwer 2010] from a broad point of view, but focussing on different details of such systems. Our overall goal is to facilitate personal interaction rather than consider the users as passive observers, which is typically the case in traditional uni-stream broadcasts. This personal interaction could involve the manual control of the virtual camera or the decision to allow automatic tracking of objects. One way to provide an interactive presence in the stadium is to deliver video in which a user can pan, tilt and zoom from a given viewpoint, the position of the cameraman. Ideally, if a user was present at the stadium, these camera movements are the degrees of freedom he or she should have without moving. However, when scaling such a system to several users, the delivery part is constrained to be independent from the capturing part (no physically moving cameras based on user needs). In this article, we briefly show how the panorama is generated and how the virtual view is extracted from the panorama. Then, we present different ways to automatically control the zoom, pan and tilt. Finally, we perform a 2-step user study where the first step focuses on comparing different algorithms for servoing the virtual camera and the second step evaluates the machine generated movements against those generted by human operators. We analyse the user studies aiming to answer the question whether a machine has the potential to be as good as a human operator, and the answer is promising.

The remainder of the article is organized as follows. Section 2 briefly outlines some of the many related works in relevant intersection areas. Then, Section 3 introduces our system, before we look into the details of the automatic camera control algorithms in Section 4. In Section 5, we present experimental results on the technical implementation, and Section 6 encompasses the user studies that evaluate manual and automatic camera controls. Finally, we discuss the results and implications in Section 7 before we conclude the article in Section 8.

2. RELATED WORK

Our system contains many integrated components. In this section, we briefly describe the ones we found most similar and closely relate to our approach.

Ren et al. [2010] provide details about an 8-camera system that is able to keep track of players using hypothesis from the multiple views. However, the scope of their paper is limited to extracting the position information. Several free-viewpoint systems [Carranza et al. 2003; Debevec et al. 1996; Grau et al. 2004; Kanade et al. 1997] exist to provide the viewer with the power to change the view point to the desired one smoothly. However, all those have limited challenges due to the fact that they are made indoors. Outdoor sports provide ample number of challenges to reuse the same techniques in terms of space, illumination changes and uncontrolled conditions. Thus

[1]http://www.youtube.com/watch?v=E8jSOv8Ch5s - [10:00 - 11:00]
http://www.youtube.com/watch?v=FEM0dY8c0co.

the depth based image rendering techniques [Papadakis et al. 2010; Grau et al. 2007] still widely suffer to achieve the production quality. This can be seen in the recent work by Goorts et al. [2014]. However, impressive the functionality is to a researcher, the visual quality is still far from delivery to general audience. Hence we looked at single view-point approach.

First, generating a panorama video has by itself several challenges, and extensive amounts of literature is available for panorama creation. A well-known example is the panoramic capture system that was used to record a world cup match during FIFA World Cup 2006 [Fehn et al. 2006]. Similar works include [Xu and Mulligan 2013; Gaddam et al. 2014a; Carr and Hartley 2009] where the authors emphasize on engineering challenges related to a system for recording panoramic videos. Nevertheless, recent approaches prove that panorama video can be generated in real time [Gaddam et al. 2014b], then the challenge remains to extract virtual views from the panorama.

Using a panoramic texture for an immersive feeling is also a researched topic [Jenkin et al. 1998; Chen 1995], also in different applications areas (e.g., lecture videos [Yokoi and Fujiyoshi 2005; Sun et al. 2005; Ahmed and Eades 2005]). Some works also describe manually controlled virtual cameras generated from panoramic videos with emphasis on human-centered interfaces [Foote et al. 2013], network performance [Mavlankar and Girod 2010] and over-all system aspects [Gaddam et al. 2014b].

Automation of camera control at several levels has also been explored before. For example, Wang et al. [2004] provided a multilevel framework to automatically generate replays from just one camera. Dearden et al. [2007] provide an evolving system that learns from the movement of a trained camera operator. Several works focused solely on control theory of virtual cameras from multiple cameras [Lipski et al. 2009; Christie et al. 2005; Hutchinson et al. 1996]. Though these are interesting approaches, we want to build an entire system that extracts a virtual view from a high-resolution panorama controlled either by the user or a machine operated cameraman in real time. The idea is to put together these different components and bridge the gap so as to make these components function in coherence.

We are definitely not the first to explore such ideas. For example, Ariki et al. [2006] provided a prototype where they used clipping on a portion of an HD recording as a means of creating a virtual camera. They generate the virtual camera motion automatically based on situation recognition from the game. Their user study focuses more on learning the effect of various evaluation criteria like naturality in zooming, panning, shot size, duration, video quality and intelligibility on the audience preferences. This is probably the closest and a simpler version of our work. Furthermore, Carr et al. [2013] presented a hybrid system using both a robotic PTZ camera and a virtual camera generated from panorama. They evaluated their system comparing it to a human operated one as benchmark. Their motivation is to get as close to the human operator as possible. Even though a really thorough work dealing with automatic virtual cameras, they fixed the focal length and the tilt angle subjecting to less exposure to scrutiny by the viewers about the short-comings from changing these variables. Similarly, Chen and De Vleeschouwer [2010] performed an automatic production planning over multiple cameras. They employed an individual stimulus rating based evaluation system which cannot be directly used for comparing different variables. Both of these works focus on basketball as their case study, which has a much more limited field size compared to a soccer field, giving smaller panning requirements.

However, the main focus of these investigations is whether the perceived experience from a automatically controlled virtual view can match the one generated by a human. We try to learn from the existing approaches and design a system for automatic control of a virtual view extracted from a high-resolution, wide field-of-view panorama video

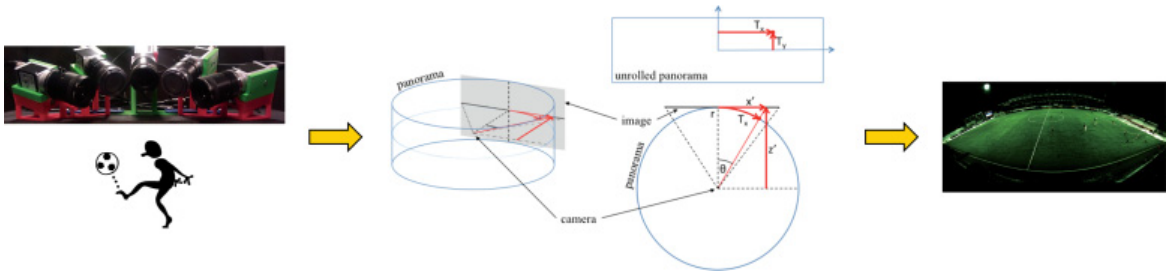


Fig. 1. The camera array captures videos from the individual cameras, each with its own spatial coverage. The idea is to have the cameras in the center of a virtual cylinder where every source image is considered a (cropped) plane, tangential to the cylinder and orthogonal to the camera's viewing axis. Each pixel of the cylinder is then computed as the (interpolated) pixel value of the ray from the camera center through the pixel intersecting the image plane.

and evaluate it. But we do not use the human operator as a benchmark, instead as a competitor in our evaluations.

3. SYSTEM SETUP

Our solution is composed of a sensor system to capture player positions, a mobile-phone-based expert annotation system to tag events live, and a camera-array-based video system to generate a wide field-of-view panorama video in real time [Halvorsen et al. 2013]. The current prototype is deployed and running in an elite club stadium.

Because the main focus of this article is the video system, we first describe the generation of the panorama video before we describe how the system enables individual users to control their own virtual camera.

3.1. Panorama Recording System

To capture the visual sequences and generate a real-time panorama video [Tennøe et al. 2013], we use a distributed recording system. Five Basler cameras capture videos at a 1086×2046 pixel resolution, arranged as illustrated in Figure 1. Recording machines thereafter transfer the captured raw video data over a PCIe-based high-speed interconnect network to a panorama processing machine. To give a higher vertical resolution, the cameras are rotated 90 degrees. Moreover, the frequent changes in the outdoor light conditions require auto-exposure to be performed on the center camera at fixed intervals, with the resulting exposure parameters broadcasted to the other camera readers. The model in Figure 1 portrays how the processing machines warps the frames onto a cylinder and then stitches the cylindrical panorama, with the seams calculated dynamically for each frame in order to avoid the ghosting artifact of the moving objects (the players and the ball). The resulting stitched videos have output frames of 4096×1680 pixel resolution, as seen on the right of Figure 1. Finally, the processing machines encode the panoramic video in H.264, making it available for HTTP streaming.

3.2. Virtual View Generation

A pinhole camera model is applied to generate a virtual view [Gaddam et al. 2014b] of the field of interest, with pixels fetched from the panoramic texture. The reprojection onto the virtual camera plane preserves a perspective nature of the output view, where the straight lines remain straight, irrespective of any distortions introduced during stitching. The perspective nature is accomplished using the pin-hole-based point projection from a 3D point P to an image point q, which can be written as follows:

Fig. 2. Virtual view generated from re-projection. The panorama video with the marked region of interest is shown together with the generated virtual camera, emphasizing that the extracted area is not a simple crop from the high-resolution panorama video.

$$\lambda q = [K|0_3] \begin{bmatrix} R & 0 \\ 0_3 & 1 \end{bmatrix} \begin{bmatrix} 0_3^T & -C \\ 0 & 1 \end{bmatrix} P, \tag{1}$$

where R is the general (3×3) 3D rotation matrix as a function of θ_x, θ_y and θ_z, the rotation angles around the x, y and z axes, respectively. Moreover, K is the camera intrinsic matrix built with focal length (f). Then, if p is the current pixel, we need to find the ray (s) that passes from the camera center C to the pixel p:

$$s = \lambda R^{-1} K^{-1} p. \tag{2}$$

Then, the intersection point of this ray with the unit cylinder gives us the exact position on the cylindrical texture:

$$T_x = \left(\frac{W_p}{FOV}\right) \left\{ arctan\left(\frac{-s(1)}{s(3)}\right) \right\} + \frac{W_p}{2} \tag{3}$$

$$T_y = \left(\frac{1}{2} - \frac{s(2)}{\sqrt{s(1)^2 + s(3)^2}}\right) H_p. \tag{4}$$

Here, the point (T_x, T_y) represents the coordinates on the unrolled cylindrical texture as described before, and W_p, H_p and FOV correspond to the width, height and field-of-view of the panoramic texture, respectively. When these calculations are performed with subpixel accuracy, the intersection will not necessarily land at one pixel. Consequently, an interpolation may be required from the surrounding pixels, which we manage using bicubic interpolation [Gaddam et al. 2014b]. Depending on the requested output resolution, the entire virtual camera frame is generated in about 10 ms. An example of a generated view is included in Figure 2.

In the current setup, the client controls and generates the virtual view using a client program. This program fetches the decoded video segments, before the final virtual

view is controlled either manually or guided by the orchestrating program. The latter is achieved using position data, in this scenario, the position of the ball or the selected players, with either object tracking [Kaiser et al. 2011; Xu et al. 2005; Yu et al. 2003] or available sensor data.

In the current context, we have applied automatic tracking and we describe in this article key approaches for virtual camera movements. By doing this, we seek to minimize computational expenses, thus enabling the client to run on devices with different capabilities, ranging from high-capacity desktop machines to mobile phones.

4. APPROACHES FOR AUTOMATIC CAMERA CONTROL

To operate the virtual camera automatically, we are operating in 3D space with the origin on the axis of the cylinder for all the movements. Here, we let θ_x be the angle along the pan direction, θ_y be the angle in the tilt direction, and f be the focal length, that is, these three variables are used and changed to control the virtual camera. A ray pointing at $(\theta_x, \theta_y) = (0, 0)$ meets the panorama image at the center.

Furthermore, let the feature point on the panorama be $s_p = (\theta_x^p, \theta_y^p)$, and let the current state of the camera be $c^i = (\theta_x^i, \theta_y^i, f^i)$ where previous states are denoted $c^{i-1}, c^{i-2}, \ldots$. Then, the problem of operating the virtual camera can be formulated as

$$c^i = F\big(s_p^{i+l}, s_p^{i+l-1}, s_p^{i+l-2}, \ldots, c^{i-1}, c^{i-2}, \ldots\big), \tag{5}$$

where l is the future data fetched by simply delaying l units of time. The broadcast can be slightly delayed from real-time and this is quite a common phenomenon with delays attributing to delays in channel, direction process etc. However, the processing in our system is automatic and strictly real time, we can introduce an artificial delay to provide some future data to the servoing algorithms. This helps us keep the causality of the system, because in reality, the future data has already been captured. The models that we developed for controlling the virtual camera handle the state variables independently. There are two models for controlling the angles and the focal length is controlled depending on the current position of the center of the virtual camera on the panorama.

4.1. Models for Pan and Tilt

We have used two different models for the pan/tilt operations, that is, a Schmitt trigger and an Adaptive trigger. The pan and tilt angle movements are assumed to be independent. However, the changes in tilt angles are penalized more than the pan angles because panning is usually more natural than tilting a camera in wide field of view situations.

4.1.1. Schmitt Trigger. The concept of a schmitt trigger is to stabilize noisy input data. We modified it so as to provide a smoother movement by adding an acceleration α. For the schmitt trigger to function, we define an imaginary window[characterized by θ^t] inside the virtual view. When the target point is inside the imaginary window, the system is brought quickly, yet smoothly, to rest by using an acceleration α_{stop}. Once the target point goes outside the window, we provide an acceleration α, to the view so that we reach the target. The sign of α depends on the current velocity of the feature point and the virtual camera. The acceleration is added only when the velocity is less than the maximum velocity $\delta\theta_max$. Algorithm 1 presents this approach. The velocity and acceleration of a variable θ are written as $\delta\theta$ and $\delta^2\theta$, respectively.

4.1.2. Adaptive Trigger. The adaptive trigger is designed to adaptively estimate the required velocity of the virtual camera. We smooth the movement of the camera in a

ALGORITHM 1: Schmitt Trigger

```
if θ^p is outside θ^t then
    if δθ^p > δθ^{i−1} then
        δ²θ ← α
    else
        δ²θ ← −α
    end if
else
    δ²θ ← α_stop
end if
```

two step smoothing process. We use a running weighted mean smoothing at both steps. Another key difference in this model is the use of future data. By delaying the system by 1 second, we have future data for about 1 second. The windows for the regression are smaller than the fetched future data because of the second level smoothing. For a given variable x let $S(x)$ be the smoothed value. Algorithm 2 describes this approach. When computing the target velocities, the gradient is taken over smoothed feature positions because the noise get amplified with a gradient. τ is a threshold for removing small variations in position that are caused by small jerky motions. These jerky motions create a small average velocity over multiple frames. We preferred to keep the camera static rather than subjecting it to a really slow movement.

ALGORITHM 2: Adaptive Trigger

```
(θ_x^0, θ_y^0) ← (θ_x^p, θ_y^p)
while running do
    δθ^s = δ(S(θ))
    if δθ^s > τ then
        δθ^st = δθ^s
    else
        δθ^st = 0
    end if
    δθ = S(δθ^st)
end while
```

4.2. Models for Zoom

The zoom is controlled by modifying f accordingly, the virtual view is zoomed by increasing f. In the current system, we developed two models to change f depending on where the virtual view is looking at. With our knowledge from different broadcasts, we wanted to find out the preference of audience for these two commonly used zoom mechanisms.

4.2.1. Smooth Zoom. This is to imitate the nature of the physical zoom that is obtained by smoothly controlling the zoom ring on the recording camera. We modelled a quadratic function in the current camera position coordinates such that f increases when the position approaches the goal posts or the other end of the field from the camera setup.

$$f^i = \lambda_0 + \lambda_1\left(\theta_x^i - \theta_{x0}\right)^2 + \lambda_2\left(\theta_y^i - \theta_{y0}\right)^2, \tag{6}$$

where λ_1 and λ_2 are the parameters that control the effect of pan and tilt angles respectively. θ_{y0} is used to offset the curve so that the function is an increasing one over all the tilt angles. θ_{x0} is set to 0, because the function should be increasing from the

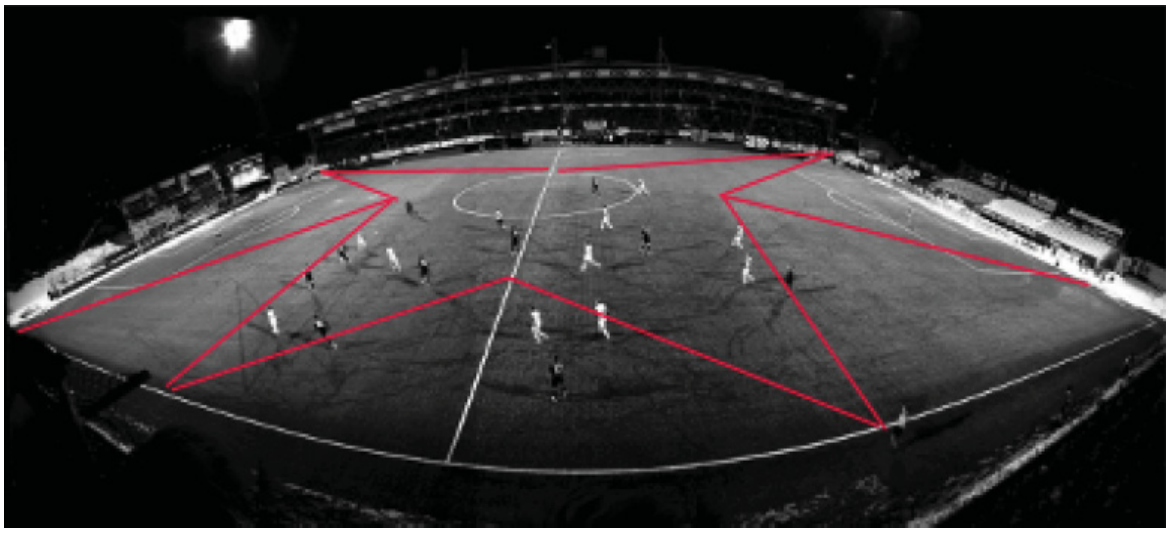


Fig. 3. Toggle zoom assignment.

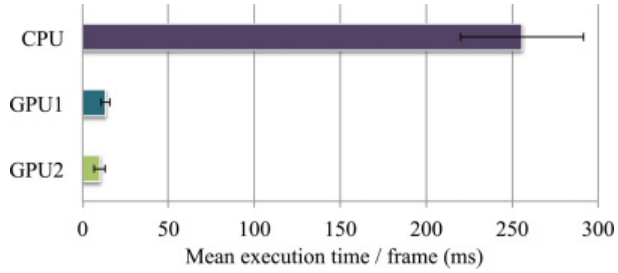


Fig. 4. Total execution time per frame in milliseconds.

center of the field as we move towards the goals. λ_0 is the zero order offset. All these parameters are empirically selected.

4.2.2. Toggle Zoom. This mode was developed to imitate the immediate switch in zoom levels. We picked a rather simple model for creating this effect. The panorama is partitioned into several zones and a focal length is assigned per zone. The zones can be seen in Figure 3.

5. EXPERIMENTAL RESULTS

To make an objective evaluation of the system, we have performed different sets of tests. The first set of tests evaluates the real-time properties of the virtual view generation. The rest of the experiments evaluate camera movements.

5.1. Execution Overhead

We have earlier proved that the given system can provide panorama video in real-time [Gaddam et al. 2014b], and in the context of generating a virtual view, we have tested three different implementations: 1) a CPU version just looping through all the pixels per frame; 2) a straight forward GPU port; and 3) an optimized GPU implementation where the system renders OpenGL textures written by an NVidia CUDA kernel directly from the GPU to the screen. The total per frame execution times for a virtual camera with full HD resolution on an Intel i7-2600 CPU with an Nvidia GeForce GTX 460 GPU is shown in Figure 4. Both GPU versions easily reach the

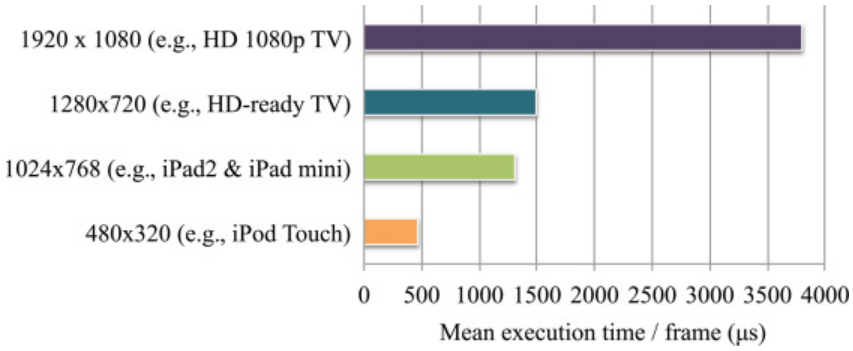


Fig. 5. Core execution times for various resolutions.

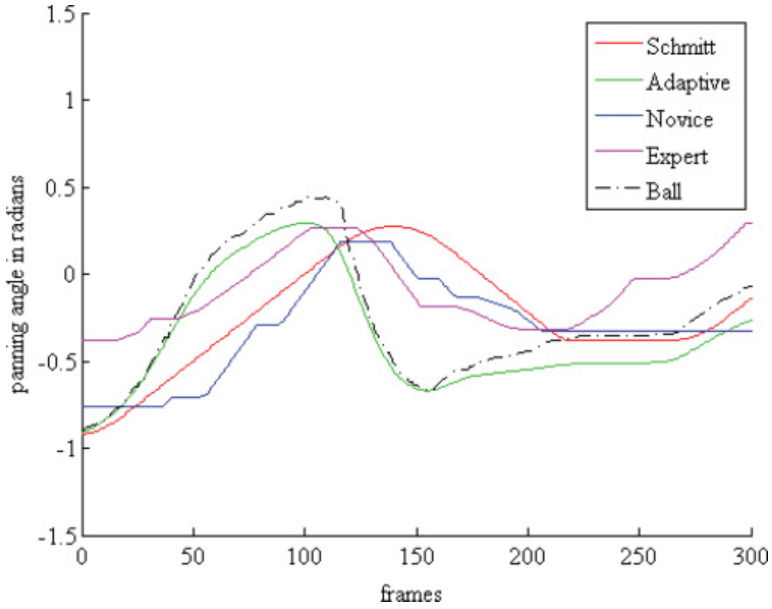


Fig. 6. Schmitt trigger and adaptive trigger plots for 300 frame segment along with the plots from human operated camera.

real-time requirements of 25 frames per second with an average of 13 and 10 milliseconds, respectively.

Now, not all receivers require a full HD video. In this respect, we have also evaluated the impact of interpolation and ray intersection costs depending on the size, that is, the only parts that varies with the output resolution. The results are shown in Figure 5. In short, the size of the output is negligible.

5.2. Pan/Tilt Models

The execution times for the Schmitt trigger and adaptive trigger are around $2\mu s$ and $30\mu s$, respectively. Even though they differ by several orders, the absolute values are still negligible. Figure 6 provides a 300 frames segment for camera movements and ball position, where we see the pan/tilt angle (in radians) of the virtual view generated by both machine and human operations. In other words, if the curves are close, they capture more or less the same view. The causal nature of the Schmitt trigger and the human operators can be observed in the figure owing to the fact that, they get the ball position as it happens. On the other hand, the adaptive model has access to a small future data.

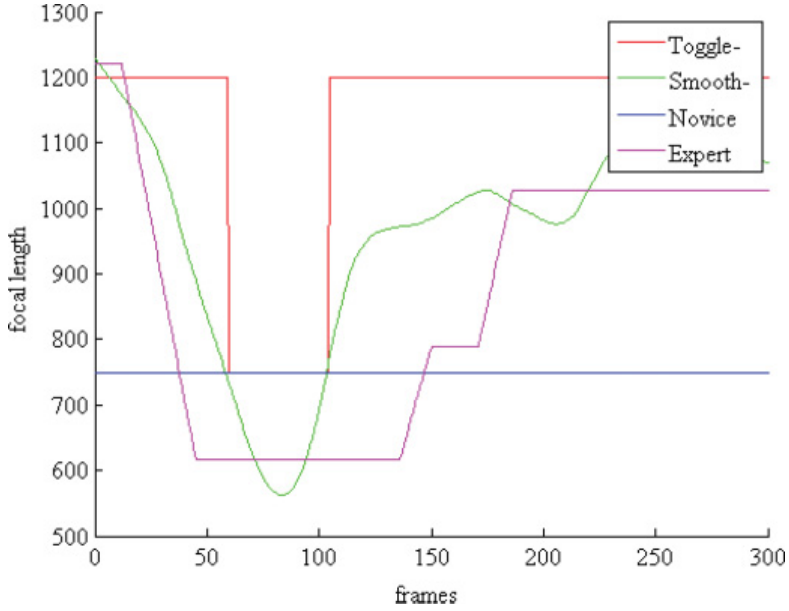


Fig. 7. Smooth zoom and toggle zoom plots along with the plots from human operated camera for 300 frame segment.

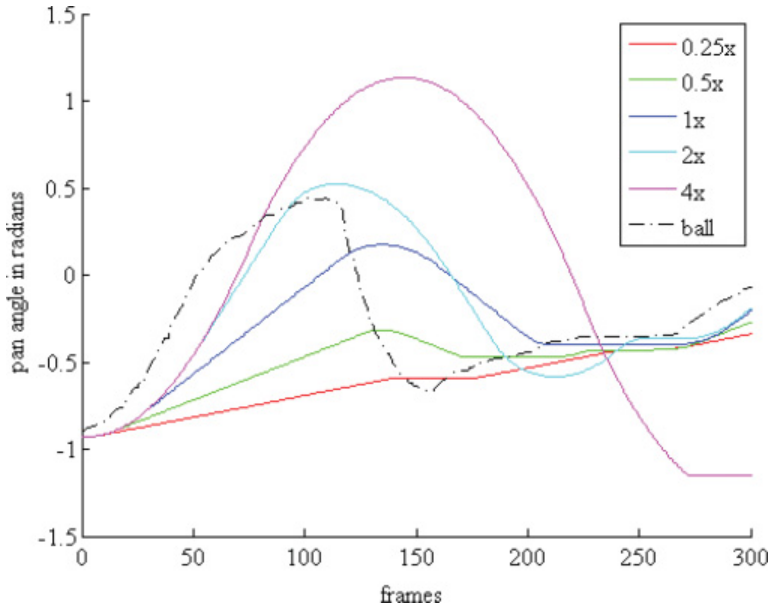


Fig. 8. The calculated trajectories for various acceleration values in the Schmitt trigger case.

5.3. Zoom Models

Since the calculation of zoom is a closed form expression over the current viewing position, the execution time is really low. Figure 7 provides plots from the different zoom models and the human operators over a 300 frames segment. Since the position is dependent on the pan/tilt model chosen, both curves are calculated using the adaptive trigger. It can be observed that the machine generated zoom curves show noticeable similarity to the expert controlled camera, irrespective of the simplicity in the models.

5.4. Schmitt Trigger—Analysis

There are three control parameters in the Schmitt trigger case. The acceleration, maximum velocity and the stopping-acceleration. Figure 8 displays the curves angle curves

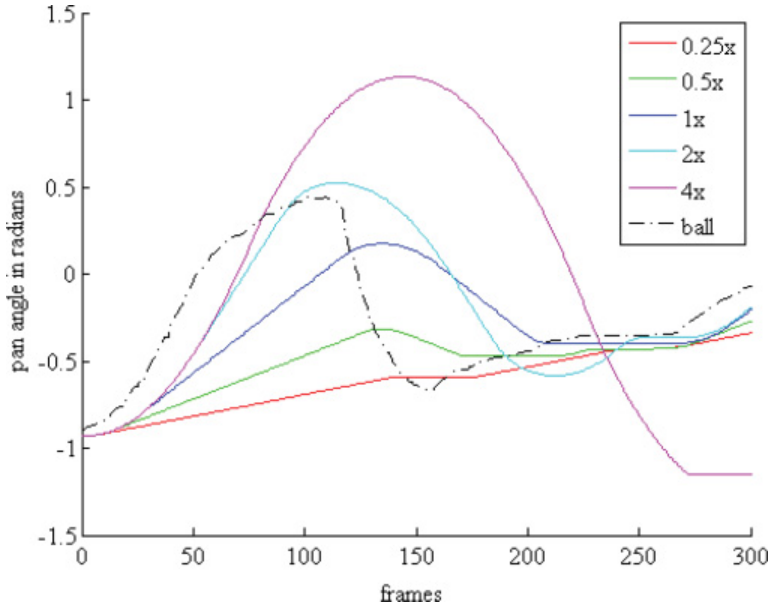


Fig. 9. The trajectories for various max velocities in the Schmitt trigger case.

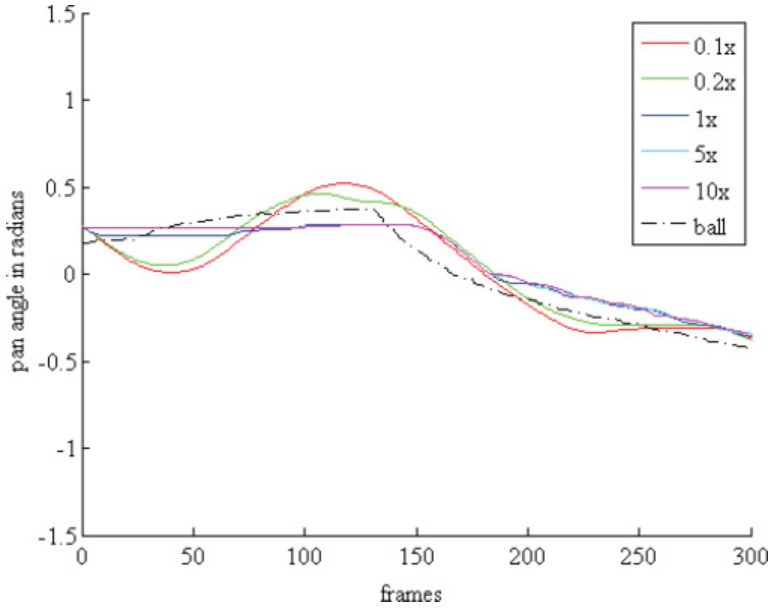


Fig. 10. Effect of varying stop-acceleration on the trajectories in Schmitt trigger case.

for different accelerations over 300 frames. It can be observed that higher acceleration tends to get the camera center closer to ball position quickly, but a problem is that it also introduces uneasiness in watching.

Figure 9 demonstrates the effect of varying the maximum velocity over 300 frames. When the ball moves really quickly, the curves in the plot show that the higher the maximum velocity, the closer they get to the slope required. However, this creates an undesired effect of overshooting irrespective of the quick deceleration.

Moreover, Figure 10 demonstrates the effect of the stop acceleration on the virtual camera movement. The trade-off here is between an appearance of a mechanical stop to a swinging effect. Both the velocity and acceleration effects can be seen in the plots.

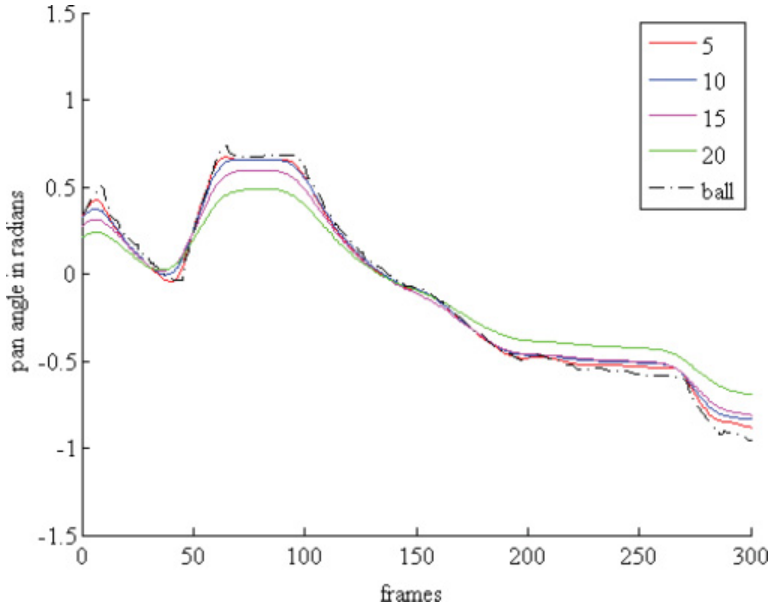


Fig. 11. Effect of window size selected for smoothing on the trajectories using the adaptive trigger.

5.5. Adaptive Trigger—Analysis

In the adaptive trigger, we have two control parameters. One is the window size and the other is thresholding for clipping. The thresholding for clipping only eliminates small jerky movements, so it is empirically chosen and it's plots do not provide great variation. Figure 11 demonstrates the effect on window size on the panning variable. The window size is varied between 5, 10, 15, and 20 frames. A scene of 300 frames where there are enough changes in the ball direction is picked. The exact field of view depends on the current focal length. However, as a rule of thumb, anything inside 0.2–0.5 radians from the center of the virtual camera can be assumed to be inside the field of view.

6. USER STUDIES

In the development of a user-centered system like ours, subjective feedback is essential to select the approaches giving the best quality of experience (QoE). Several experimental approaches [ITU-T 1998; ITU-R 2002] have been adapted and extended by researchers in the field of multimedia. Such QoE studies aim to assess, for example, behavioural responses to different aspects of multimedia systems [Wu et al. 2009], and particular attention has been devoted to the perception of video quality and the detection of visual artifacts [Farias et al. 2007; Goldmann et al. 2010; Ni et al. 2011]. To perform our assessment experiments, we have taken advantage of the flexibility of online tests which lately have become common [Chen et al. 2009]. Furthermore, since the user experience with the system is dependent on many factors outside video quality, we decided to introduce a pairwise comparison test [Lee et al. 2012; Ni et al. 2011] to contrast the different combinations of camera movements, two by two. When asked to select one of two versions of the same sequence, participants are presented with a task that is comparatively simpler than subjective ratings of sequences. Seeing how pairwise comparisons only require decisions on one's preference, this test is a good alternative when exposing participants to unfamiliar stimuli and situations [Lee et al. 2012].

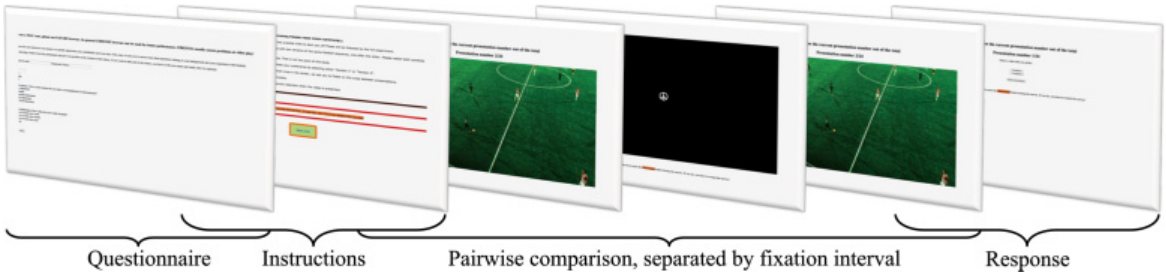


Fig. 12. Visual outline of the steps presented in the user study. Participants started with the questionnaire and instructions, before moving on to the soccer sequences. These were introduced by two practice trials, followed by the full study. Each pairwise comparison was separated by a 2-second fixation interval and terminated in a response session.

6.1. Study 1: Camera Controls

Because the system aims to provide users with the best possible experience, we need user feedback in order to establish the most preferable parameters for camera movements and zooming. We therefore conducted a user study to compare center trigger and adaptive camera movements, as well as toggle and smooth camera zooms.

6.1.1. Method. User preference for transient variables, such as camera movement and zoom, is deemed to be highly subjective and to depend on the presented sequence. To avoid subjective ratings that may vary more between presentations than between our experimental variables, we decided to use pairwise comparisons, as recommended by the ITU [ITU-T 1998]. Hence, each sequence was presented twice in a row, with only our variables of interest changing between presentations as shown in Figure 12.

Participants. A total of 49 users, 42 men and 7 women, participated in the first study. They were aged between 20 and 40 years, with an average of 27 years. Participants were presented with the opportunity to enter a lottery for a chance to win a small prize.

Stimuli and Procedure. All soccer sequences[2] were derived from the same international league match, recorded in 2013. While the ITU [ITU-T 1998] recommends a duration of approximately 10 seconds for pairwise comparisons of video presentations, we placed higher priority in ensuring that the soccer sequences contained more than one example of pan, zoom and tilt movements. Due to this, we extended the set sequence duration to 15 seconds. Automated camera movements were implemented subsequently, making sure that each movement and zoom contrast was presented four times. Each soccer sequence was therefore presented twice, separated by a two-second interval showing a fixation point on a black background. Stimuli contrasts were paired up so that either the camera movement or the camera zoom approach differed between the first and the second presentation. Although each paired contrast was presented four times, new soccer sequences were included for every pairwise comparison. Thus, participants watched 16 unique sequences, selected as the most suitable excerpts from the entire soccer match.

As stated before, we conducted the study using an online web-form so participants could complete it at their convenience. The paired video presentations were grouped in two stimuli blocks, with every contrast repeated twice within a block. Stimuli were counterbalanced with reverse-order for half of the contrasts, before they were randomised within each block. We created four randomised versions of the study, so that

[2]For a visual appreciation of the different approaches, two video sequences are included as examples for each of the different automatic and manual camera modes. These are attached with the submission.

Table I.
Parametric and non-parametric statistics for the number of times a stimulus combination was preferred over its contrasts, averaged across participants and sorted according to the Friedman rank score. Wilcoxon signed-rank test indicates statistically significant differences between stimuli, these are reported in relation to the lower ranked stimulus (the row above). Nonsignificant contrasts are labelled ns, while non-applicable comparisons are marked with a hyphen.

Results from Study 1								
Stimulus	**Parametric statistics**		**Percentiles**			**Friedman**	**Wilcoxon**	**Effect**
combination	**Mean**	**Std. dev.**	**25th**	**50th**	**75th**	**rank score**	**signed-rank test**	**size**
Schmitt/Toggle	1.77	1.40	1	2	3	1.32	–	–
Schmitt/Smooth	4.00	1.25	3	4	5	2.47	$<.001$	-0.51
Adaptive/Toggle	4.36	1.10	3	4	5	2.74	ns	-0.14
Adaptive/Smooth	5.87	1.56	5	6	7	3.47	$<.001$	-0.39
Results from Study 2								
Stimulus	**Parametric statistics**		**Percentiles**			**Friedman**	**Wilcoxon**	**Effect**
combination	**Mean**	**Std. dev.**	**25th**	**50th**	**75th**	**rank score**	**signed-rank test**	**size**
Novice	2.60	2.06	1	2	4	1.31	–	–
Expert	5.37	1.68	4	5	7	2.24	$<.001$	-0.52
Adaptive/Toggle	7.43	1.67	6	7	8	2.99	$<.001$	-0.41
Adaptive/Smooth	8.60	2.09	7	9	10	3.46	$<.021$	-0.28

the random order varied between participant groups. In order to control whether subjective preferences depended on soccer viewing experience, we introduced the study with two questions to assess soccer interest and dedication; we also collected details on age and gender. Participants received no information on the camera implementations, instead they received instructions to select the version they preferred. Following the questionnaire and instructions, we included two practice trials to get participants acquainted with the task, these were succeeded by the 16 pairwise comparisons.

6.1.2. Results. With every contrast repeated four times, the preference scores for the different conditions were added up for every participant. This resulted in individual counts the four combinations of camera movements and camera zooms, ranging from 0 to 4. In order to identify and weed out outlying preference counts, we also calculated the difference in scores between paired stimuli. This resulted in four mean differences, and we used the average of these to identify any scores that fell more than two standard deviations from the mean. Accordingly, we identified and excluded data from two participants, whose mean difference scores of zero indicated that they were unable to distinguish between stimuli. For the main analysis, we collapsed preference scores across stimulus combinations to obtain the overall number of times each camera mode was preferred by an individual. With two contrasts repeated four times for every camera mode, the highest possible preference count comes to 8. A Friedman rank test was used to analyse the preference counts from the remaining 47 participants, revealing a significant effect of our camera implementations ($\chi^2(3) = 72.73$). To further explore the difference between stimulus combinations, we also ran three Wilcoxon signed-rank tests and calculated effect sizes from these. In addition to the non-parametric tests, parametric means and standard deviations are included to better highlight the distribution of scores. Results from the analyses are presented in Table I. Furthermore, we also explored the individual contrasts with a Friedman rank test, again revealing a significant overall effect ($\chi^2(7) = 162.33$). These results are illustrated in Figure 13, listed according to their Friedman rank scores.

From the collapsed preference counts and the ranking scores presented in the first part of Table I, the adaptive trigger movement combined with the smooth focal zoom

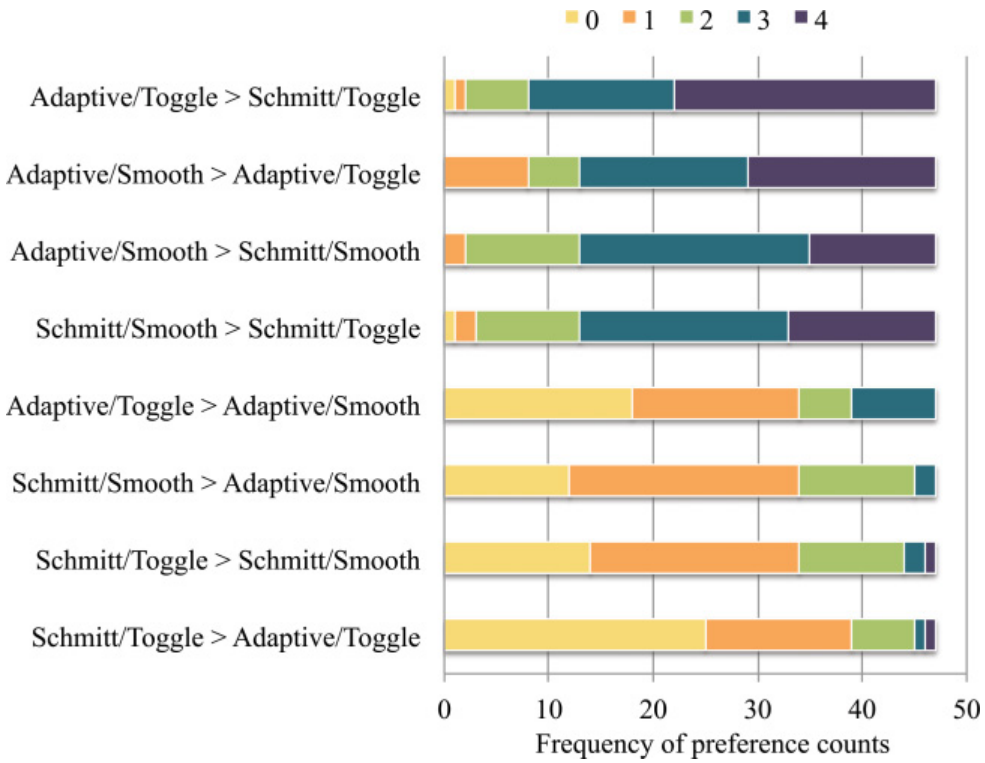


Fig. 13. Frequency distribution portraying the number of times one stimulus was preferred over its contrast, accumulated across users. For example, in the first line, we see that 25 persons have preferred the Adaptive/Toggle over Schmitt/Toggle in all four repetitions. The maximum count of 4 corresponds to the number of repetitions for each pair of videos. Stimulus contrasts are sorted according to Friedman rank scores and plotted symmetrically.

emerges as the preferred camera implementation. Although not significantly different from the third rank, the adaptive trigger remained the preferred choice over the Schmitt trigger, ranking second when combined with the toggle focal zoom. These trends are also evident when looking at the ranked individual contrasts in Figure 13. The adaptive trigger movement is preferred over Schmitt alternative for the vast majority of presentations, just as the smooth focal is the predominantly preferred zoom option over the toggle focal. In short, the opinions of 47 users clearly demonstrate the preference for the adaptive trigger and smooth focal camera implementation.

6.2. Study 2: Man vs. Machine

Following the results from Study 1, we established that users prefer the camera movement combination with adaptive trigger pan and smooth focal zoom. However, an important challenge for such an automated system is to provide a viewing experience that can compete with a soccer match filmed by a manually operated camera. Hence, the second user study compares user preferences for the two highest ranked automated camera implementations with that of two human operators.

6.2.1. Method. The second user study was conducted as a pairwise comparisons test, with the same setup used for Study 1.

Participants. With 14 females and 23 males, we collected data from 37 participants, none of whom had taken part in Study 1. Their ages spanned from 21 to 71 years, with an average of 29 years. Every participant was provided with the opportunity to sign up for a lottery that offered small prizes to be won.

Stimuli and Procedure. To compare automated camera movements with manual camera operations, we selected the two best-preferred stimulus combinations from Study 1. In so doing, we re-used half of the stimuli from the first user study and compared these to sequences with recorded camera movements. To record the camera movements, we invited an expert and a novice camera operator to watch the same soccer match. The expert was an experienced camera operator from a Scandinavian broadcaster, whereas the novice had experience with camera-view operations within games. After receiving instructions on how to move and zoom with the virtual camera using a joystick, the operators embarked upon the task of following the match by keeping the ball and action in focus. From their recordings, we selected 20 expert and 20 novice 15-second excerpts to contrast with the automated sequences. For further verification of the preference ratings from Study 1, we also contrasted the automated sequences with each other. Moreover, we contrasted the expert and novice recordings to see whether preferences differed between the two.

Study 2 proceeded in the same manner as Study 1, described in Section 6.1.1. The only procedural distinction between the two studies is the inclusion of more stimuli, resulting in 24 pairwise comparisons.

6.2.2. Results. Response data from Study 2 were restructured and analysed the same way as described for Study 1 in Section 6.1.2, again with 2 outliers detected and excluded. With the Friedman rank test indicating significant differences between the collapsed preference counts ($\chi^2(3) = 56.73$), we again followed up with Wilcoxon signed-rank tests. Results from these analyses are included in Table I. A second Friedman rank test revealed significant differences also between the individual contrasts ($\chi^2(11) = 177.15$), the ranked preference counts for these are portrayed in Figure 14.

First and foremost, the results from Study 2 reveal that users clearly prefer automated over manual camera movements. Of course, the quality of manual controls is only as good as the operator. We considered this possible limitation and took precautions by including two camera operators, one expert and one novice. The higher ranking of the expert over the novice operator exemplifies the importance of the camera man's expertise. Despite our precautions, we cannot ascertain that users will prefer the automatic camera operations over any camera operator. However, considering the significant differences and the magnitudes of effect sizes for the presented conditions, the results show that our system outperforms the two human operators. Specifically, a consistent trend can be observed for both the collapsed preference counts (Table I) and the individual contrasts (Figure 14), where the automated camera movements are chosen over the manual operations in the majority of presentations. Furthermore, the higher rank for the adaptive/smooth over the adaptive/toggle combination reflects the results from Study 1.

7. DISCUSSION

The motivation behind designing and developing such a system lies in providing an interaction to the user. In cases where manual control of the virtual camera is desired, the system simplifies significantly. On the other hand, a viewer following a game might be interested in interaction but at a higher level. The viewer might place a request to the client to follow the ball/a single player or a collection of players. In such a case, the client has to provide an aesthetically pleasing virtual camera based on the position data from the ball and players. Even a coach is greatly advantaged by such a system, he/she can instantly request multiple virtual cameras focussing on different features. For example, one for the ball, one for a recently injured player, one for a recently exchanged player and one for the defense. So, building the entire system and a subjective evaluation of the results proved to be mandatory.

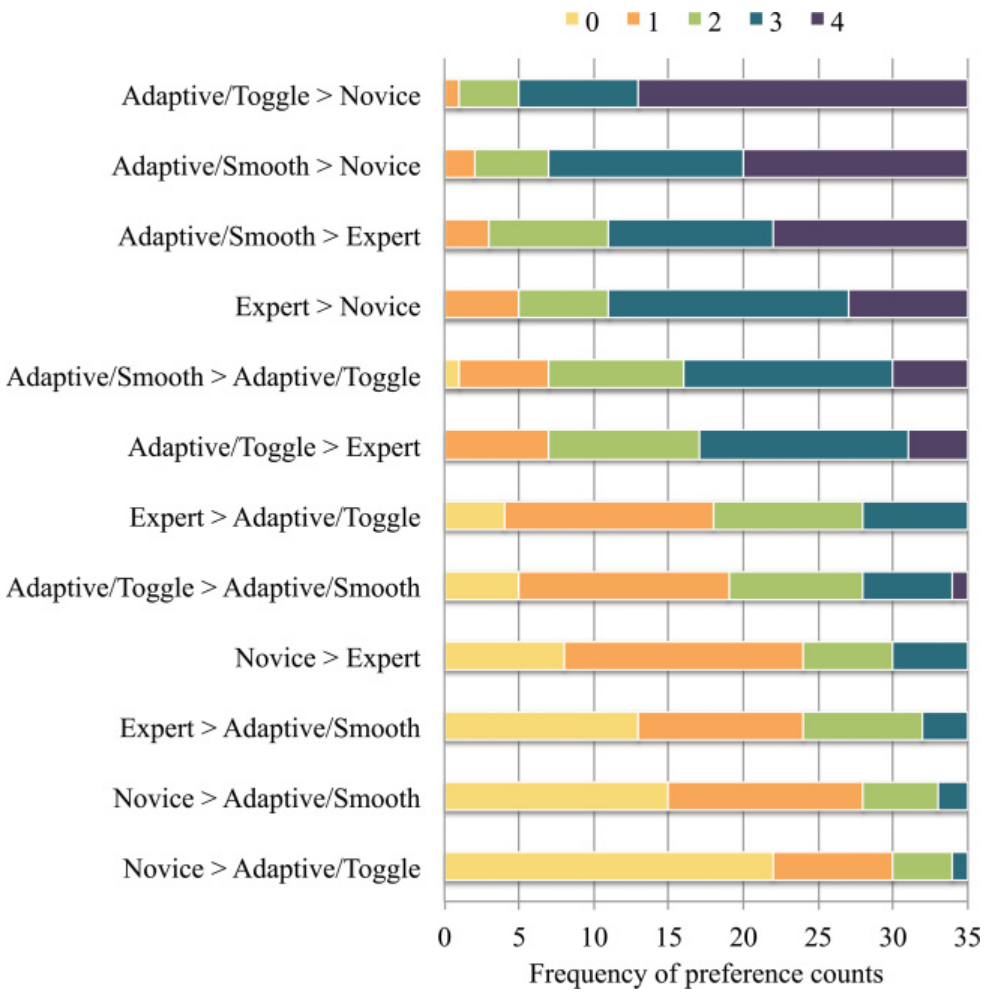


Fig. 14. Frequency distribution portraying the number of times one stimulus was preferred over its contrast, accumulated across users. The maximum count of 4 corresponds to the number of repetitions. Stimulus contrasts are sorted according to Friedman rank scores and plotted symmetrically.

In the two user studies, we have explored and analysed user preferences for automated and manual camera movements. The first study established that the average user prefers the adaptive trigger movement over the Schmitt trigger and the smooth focal zoom over the toggle; implications of these findings are discussed further on. From the second user study, we found that the average user maintains the same preference for the adaptive trigger and the smooth focal zoom when compared to a human-operated camera. While this finding is specific to the current context and may not reflect the performance of all camera operators, the subjective preference for automated camera movements suggests a positive user experience with our system. Overall, the presented results are promising for the future acceptance and use of our system.

The user preferences between the toggle and smooth zoom is slightly ambiguous. From the user study, it is clear that the smooth zoom is preferred, but toggle zoom provides the advantage to switching to an overview immediately. This when combined with smooth zoom for smaller ball changes can provide a nice aesthetic, yet functional camera motion that can keep the ball in field of view. Moreover, the zoom model currently is based only on the position of the ball on the panorama. This can be significantly improved by incorporating game context into the model. Some of the things can be velocity of the ball, player arrangement and special events (penalty, corner or throw-in).

Furthermore, it must be noted that this study focuses on one of the several points from where the action is captured on the soccer field. When it comes to capturing from one point in live, the camera man has little freedom in the grammar of the video. In an actual broadcast, the producer mixes several streams together and this is where the grammar come into place.

In future, we are aiming to improve several components of the system. We are currently working on capturing High Dynamic Range (HDR) panoramic videos to handle the loss of details in shadows on sunny days. We are also investing our energy into developing the client on a mobile platform, which has its own challenges concerning the bandwidth and power consumption.

Moreover current day's visual tracking algorithms' recall is not practically applicable to real-life scenarios. Owing to this, we still have a large manual component when it comes to estimating the ball position. We are currently exploring algorithms based on multi-sensor data to track the ball with a high recall rate. When we track the ball successfully, we will be able to provide a complete system functional in real time. However, we do have an accurate tracking of the player positions, meaning that the system easily can follow a single player or a group of players.

8. CONCLUSION

In our research, we have shown that a single camera-array generated panorama video can support an arbitrary number of virtual views, which are generated locally on the client device. In many scenarios, users will want to control and interact with their own virtual camera, whereas other situations require higher levels of abstraction. In order to generate video streams that incorporate automated camera movements while satisfying user expectations, we have explored machine-controlled camera modes versus human camera operators in two separate user studies. In the first, we explored automatic movement approaches and established that the best-preferred mode combines smooth focal zoom with adaptive trigger movements. The second study compared machine and human generated camera movements, with results that promise well for future acceptance of a machine controlled cameraman. However, It must be noted that we are not claiming a system that is capable of exceeding a human operator. The research outcomes here do not guarantee an assertion that a machine can beat a human. The study merely points at the one of the several possible positive futures in the current context and scenario. There are several many variables and a long way to generalize and extend this to the entire broadcasting paradigm. Our ongoing work include both improved object tracking and further parallelization; most importantly though, we aim to further improve the automated camera movements.

REFERENCES

Adel Ahmed and Peter Eades. 2005. Automatic camera path generation for graph navigation in 3D. In *Proceedings of the Asia-Pacific Symposium on Information Visualisation*. 27–32. http://dl.acm.org/citation.cfm?id=1082315.1082320

Y. Ariki, S. Kubota, and M. Kumano. 2006. Automatic production system of soccer sports video by digital camera work based on situation recognition. In *Proceedings of the IEEE International Symposium on Multimedia*. 851–860. DOI:http://dx.doi.org/10.1109/ISM.2006.37

Peter Carr and Richard Hartley. 2009. Portable multi-megapixel camera with real-time recording and playback. In *Proceedings of the Conference on Digital Image Computing: Techniques and Applications*. 74–80. DOI:http://dx.doi.org/10.1109/DICTA.2009.62

Peter Carr, Michael Mistry, and Iain Matthews. 2013. Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording. In *Proceedings of the ACM Multimedia Conference*. 193–202.

Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. 2003. Free viewpoint video of human actors. *ACM Trans. Graph.* 22, 3, 569–577. DOI:http://dx.doi.org/10.1145/882262.882309

Fan Chen and Christophe De Vleeschouwer. 2010. Personalized production of basketball videos from multi-sensored data under limited display resolution. *Computer Vision Image Understanding* 114, 6, 667–680. DOI:http://dx.doi.org/10.1016/j.cviu.2010.01.005

Kuan-Ta Chen, Chen-Chi Wu, Yu-Chun Chang, and Chin-Laung Lei. 2009. A crowd-sourceable QoE evaluation framework for multimedia content. In *Proceedings of the ACM Multimedia Conference*. 491–500. DOI:http://dx.doi.org/10.1145/1631272.1631339

Shenchang Eric Chen. 1995. QuickTime VR: An image-based approach to virtual environment navigation. In *Proceedings of the ACM SIGGRAPH International Conference on Computer Graphics and Interactive Techniques*. 29–38. DOI:http://dx.doi.org/10.1145/218380.218395

Marc Christie, Rumesh Machap, Jean-Marie Normand, Patrick Olivier, and Jonathan Pickering. 2005. Virtual camera planning: A survey. In *Smart Graphics*, Lecture Notes in Computer Science, vol. 3638, 40–52. DOI:http://dx.doi.org/10.1007/11536482 4

A Dearden, Y Demiris, and O Grau. 2007. Learning models of camera control for imitation in football matches. In *Proceedings of the Artificial and Ambient Intelligence Symposium*. 227–231.

Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. 1996. Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'96)*. ACM, New York, 11–20. DOI:http://dx.doi.org/10.1145/237170.237191

Mylène C. Q. Farias, John M. Foley, and Sanjit K. Mitra. 2007. Detectability and annoyance of synthetic blocky, blurry, noisy, and ringing artifacts. *IEEE Trans. Signal Process.* 55, 6, 2954–2964. DOI:http://dx.doi.org/10.1109/TSP.2007.893963

Christoph Fehn, Christian Weissig, Ingo Feldmann, Markus Muller, Peter Eisert, Peter Kauff, and Hans Bloss. 2006. Creation of high-resolution video panoramas of sport events. In *Proceedings of the IEEE International Symposium on Multimedia*. 291–298. DOI:http://dx.doi.org/10.1109/ISM.2006.55

Eric Foote, Peter Carr, Patrick Lucey, Yaser Sheikh, and Iain Matthews. 2013. One-man-band: A touch screen interface for producing live multi-camera sports broadcasts. In *Proceedings of the ACM Multimedia Conference*. 163–172. DOI:http://dx.doi.org/10.1145/2502081.2502092

Vamsidhar Reddy Gaddam, Carsten Griwodz, and Pål Halvorsen. 2014a. Automatic exposure for panoramic systems in uncontrolled lighting conditions: a football stadium case study. In *Proceedings of SPIE: The Engineering Reality of Virtual Reality*. 90120C–90120C–9. DOI:http://dx.doi.org/10.1117/12.2040145

Vamsidhar Reddy Gaddam, Ragnar Langseth, Sigurd Ljødal, Pierre Gurdjos, Vincent Charvillat, Carsten Griwodz, and Pål Halvorsen. 2014b. Interactive Zoom and Panning from Live Panoramic Video. In *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video*. Article 19. DOI:http://dx.doi.org/10.1145/2578260.2578264

Lutz Goldmann, Francesca De Simone, Frederic Dufaux, Touradj Ebrahimi, Rudolf Tanner, and Mauro Lattuada. 2010. Impact of video transcoding artifacts on the subjective quality. In *Proceedings of the International Workshop on Quality of Multimedia Experience*. 52–57.

Patrik Goorts, Steven Maesen, Maarten Dumont, Sammy Rogmans, and Philippe Bekaert. 2014. Free viewpoint video for soccer using histogram-based validity maps in plane sweeping. In *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. 378–386.

O. Grau, T. Pullen, and G. A. Thomas. 2004. A combined studio production system for 3-D capturing of live action and immersive actor feedback. *IEEE Trans. Circuits Syst. Video Technol.* 14, 3, 370–380. DOI:http://dx.doi.org/10.1109/TCSVT.2004.823397

O. Grau, G. A. Thomas, A. Hilton, J. Kilner, and J. Starck. 2007. A robust free-viewpoint video system for sport scenes. In *Proceedings of the 3DTV Conference*. 1–4. DOI:http://dx.doi.org/10.1109/3DTV.2007.4379384

Pål Halvorsen, Simen Sægrov, Asgeir Mortensen, David K. C. Kristensen, Alexander Eichhorn, Magnus Stenhaug, Stian Dahl, Håkon Kvale Stensland, Vamsidhar Reddy Gaddam, Carsten Griwodz, and Dag Johansen. 2013. BAGADUS: An Integrated system for arena sports analytics – A soccer case study. In *Proceedings of the ACM Multimedia Conference*. 48–59.

S. Hutchinson, G. D. Hager, and P. I. Corke. 1996. A tutorial on visual servo control. *IEEE Trans. Rob. Automation* 12, 5, 651–670. DOI:http://dx.doi.org/10.1109/70.538972

ITU-R. 2002. BT.500-11. Methodology for the subjective assessment of the quality of television pictures. https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.500-11-200206-SIIPDF-E.pdf.

ITU-T. 1998. P.911. Subjective audiovisual quality assessment methods for multimedia applications. https://www.itu.int/rec/T-REC-P.911-199812-1/en.

Michael Jenkin, James Elder, and Greg Pintilie. 1998. Loosely-coupled telepresence through the panoramic image server. In *Vision Interface: Real World Applications of Computer Vision*.

R. Kaiser, M. Thaler, A. Kriechbaum, H. Fassold, W. Bailer, and J. Rosner. 2011. Real-time person tracking in high-resolution panoramic video for automated broadcast production. In *Proceedings of the European Conference on Visual Media Production*. 21–29. DOI:http://dx.doi.org/10.1109/CVMP.2011.9

Takeo Kanade, Peter Rander, and P. J. Narayanan. 1997. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE MultiMedia* 4, 1, 34–47. DOI:http://dx.doi.org/10.1109/93.580394

Jong-Seok Lee, Lutz Goldmann, and Touradj Ebrahimi. 2012. Paired comparison-based subjective quality assessment of stereoscopic images. *Multimedia Tools Appl.* 67, 1, 31–48. DOI:http://dx.doi.org/10.1007/s11042-012-1011-6

Christian Lipski, Christian Linz, Kai Berger, and Marcus Magnor. 2009. Virtual video camera: Image-based viewpoint navigation through space and time. In *Proceedings of the ACM SIGGRAPH International Conference on Computer Graphics and Interactive Techniques*. Article 93. DOI:http://dx.doi.org/10.1145/1599301.1599394

Aditya Mavlankar and Bernd Girod. 2010. Video streaming with interactive pan/tilt/zoom. In *High-Quality Visual Experience*, Marta Mrak, Mislav Grgic, and Murat Kunt (Eds.), 431–455. DOI:http://dx.doi.org/10.1007/978-3-642-12802-8 19

Pengpeng Ni, Ragnhild Eg, Alexander Eichhorn, Carsten Griwodz, and Pål Halvorsen. 2011. Flicker effects in adaptive video streaming to handheld devices. In *Proceedings of the ACM Multimedia Conference*. 463–472.

N. Papadakis, A. Baeza, I. Rius, X. Armangue, A. Bugeau, O. D'Hondt, P. Gargallo, V. Caselles, and S. Sagas. 2010. Virtual camera synthesis for soccer game replays. In *Proceedings of the Conference on Visual Media Production*. 97–106. DOI:http://dx.doi.org/10.1109/CVMP.2010.20

Jinchang Ren, Ming Xu, James Orwell, and GraemeA. Jones. 2010. Multi-camera video surveillance for real-time analysis and reconstruction of soccer games. *Machine Vision Appl.* 21, 6, 855–863. DOI:http://dx.doi.org/10.1007/s00138-009-0212-0

Xinding Sun, J. Foote, D. Kimber, and B. S. Manjunath. 2005. Region of interest extraction and virtual camera control based on panoramic video capturing. *IEEE Trans. Multimedia* 7, 5, 981–990. DOI:http://dx.doi.org/10.1109/TMM.2005.854388

Marius Tennøe, Espen Helgedagsrud, Mikkel Næss, Henrik Kjus Alstad, Håkon Kvale Stensland, Vamsidhar Reddy Gaddam, Dag Johansen, Carsten Griwodz, and Pål Halvorsen. 2013. Efficient implementation and processing of a real-time panorama video pipeline. In *Proceedings of the IEEE International Symposium on Multimedia*.

Jinjun Wang, Changsheng Xu, Engsiong Chng, Kongwah Wah, and Qi Tian. 2004. Automatic replay generation for soccer video broadcasting. In *Proceedings of the ACM Multimedia Conference*. 32–39. DOI:http://dx.doi.org/10.1145/1027527.1027535

Wanmin Wu, Ahsan Arefin, Raoul Rivas, Klara Nahrstedt, Renata M. Sheppard, and Zhenyu Yang. 2009. Quality of experience in distributed interactive multimedia environments: Toward a theoretical framework. In *Proceedings of the ACM Multimedia Conference*. 481–490.

M. Xu, J. Orwell, L. Lowey, and D. Thirde. 2005. Architecture and algorithms for tracking football players with multiple cameras. In *IEE Proc. Vision Image Signal Process*. 152, 2, 232–241. DOI:http://dx.doi.org/10.1049/ip-vis:20041257

Wei Xu and Jane Mulligan. 2013. Panoramic video stitching from commodity HDTV cameras. *Multimedia Systems* 19, 5, 407–426. DOI:http://dx.doi.org/10.1007/s00530-013-0316-2

T. Yokoi and H. Fujiyoshi. 2005. Virtual camerawork for generating lecture video from high resolution images. In *Proceedings of the IEEE International Conference on Multimedia and Expo*. DOI:http://dx.doi.org/10.1109/ICME.2005.1521532

Xinguo Yu, Changsheng Xu, Hon Wai Leong, Qi Tian, Qing Tang, and Kong Wah Wan. 2003. Trajectory-based ball detection and tracking with applications to semantic analysis of broadcast soccer video. In *Proceedings of the ACM Multimedia Conference*. 11–20. DOI:http://dx.doi.org/10.1145/957013.957018

Received July 2014; revised November 2014; accepted February 2015

Appendix E

[Journal] Tiling in Interactive Panoramic Video: Approaches and Evaluation (Under Review)

[Authors:] **V. R. Gaddam**, M. Riegler, R. Eg, C. Griwodz, and P. Halvorsen
[In Review:] IEEE Transactions on Multimedia (T-MM), 2016

Tiling in Interactive Panoramic Video: Approaches and Evaluation

Vamsidhar Reddy Gaddam, Michael Riegler, Ragnhild Eg, Carsten Griwodz, Pål Halvorsen

***Abstract*—Interactive panoramic systems are currently on the rise. However, one of the major challenges involved in such a system is the overhead to transfer a full quality panorama to the client where only a part of the panorama is used to extract a virtual view. Thus, a system should maximize the user experience and at the same time minimize the bandwidth required. In this paper, we apply tiling to deliver different qualities of different parts of the panorama. Tiling has traditionally been applied to delivery of very high-resolution content to clients, and here, we apply similar ideas in a real-time interactive panoramic video system. A major challenge is movement of such a virtual view, where clients' regions of interest change dynamically and independently from each other. We show that our algorithms, which progressively increases quality towards the point of the view, manages to (i) reduce the bandwidth requirement and (ii) provide a similar QoE compared to a full panorama system.**

***Index Terms*—Multimedia system, tiling, user studies, video, panorama.**

I. Introduction

The role of videos in the Internet got more and more important in the last years. YouTube and Netflix are alternately mentioned as the sources of most Internet traffic [1], and also other big companies like Facebook integrate videos and sharing of them in their services [2]. The commercial use of video streaming in the Internet has not only led to a proliferation, but also to the user expectation of high-quality videos, and the service providers fulfil them. YouTube users can already watch videos in 4k. The adoption of these high resolutions means that the classical video streaming challenge, the availability of bandwidth, persists in spite of growing capacities [3].

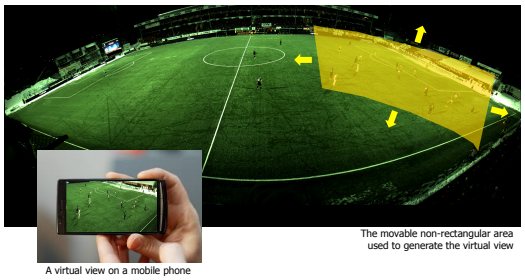


Fig. 1: The re-projected virtual view, and the panorama video with the marked region of interest.

A special case among high-resolution videos are panoramic videos. They have uses in video surveillance, sports analysis, robotics, etc. They differ from other video application in their user interaction, because most of the time, users watch only a sub-section of the entire video. A large number of panorama solutions exist, including research prototypes and commercial products, but their potential is still largely unexplored and delivery techniques not yet perfected like for example in [4–7].

Panoramic video is usually created from multiple cameras that cover a wide field of view and are stitched into one high-resolution frame. Users are then commonly given the opportunity to access narrower views extracted from the panorama using pan-tilt-zoom (PTZ) operations. This means that each user controls one virtual camera (or more) interactively to create a view. Since panoramas created by stitching multiple camera view are typically cylindrical panoramas like in figure 1, which can provide a roughly uniform distribution of pixels to all angle recorded in the panorama, even panning operations amount of more complex operation than a plain cropping.

Furthermore, in systems with a high number of users, the virtual view is usually generated on the client side due to allow the system to scale and keep interaction latency low. The downside of this approach is that the entire panorama must be delivered to the client at all times because the user can perform PTZ operations at any time. Obviously, the delivery of a full-quality high-resolution panorama video is (excessively) costly in terms of bandwidth. In an example installation where we generate a 4096x1680 x264-encoded panorama video, the average bandwidth requirement was about 9.5 Mbps. Moreover, since only parts of the panorama are used for the virtual view extraction (see figure 1), sending the full-quality panorama at all times wastes bandwidth.

To reduce the waste, but keep the user's quality of experience (QoE) high, we present an analysis of options and propose a solution that combines tiling and HTTP adaptive streaming (HAS). This is done in three main steps. (i) Using ideas from region-of-interest streaming and retrieving higher quality in the areas of the full panorama currently used for the virtual view, we analyze and discuss the trade-offs. (ii) We conduct a subjective study to validate objective quality metrics for our scenario and then use the latter to investigate the trade-off between video quality and bandwidth for several adaptation strategies. Finally, (iii) we present a possible solution for transferring a high quality panoramic video cost effective and in a high quality to the user. Our experimental results show that our approach reduces the bandwidth requirement and provides a similar Quality of Experience (QoE) compared to a full panorama system.

The rest of the paper is organized as follows. First, the state-of-the-art is discussed in the section II. In the following

Manuscript received April 19, 1988; revised February 31, 1982.

section III, a detailed overview of the system is given. After that, quality selection approaches are discussed in section IV. This is followed by presenting the evaluation methods in section V and the results of the evaluation in section VI. The discussion about the outcome of the paper can be found in section VII. Finally, conclusions are given in section VIII.

II. Related Work

Panoramas can be divided broadly into two groups, i.e., complete panoramas that span 360^o around at least one axis and partial panoramas with less angular coverage. When it comes to complete panoramas, a cube-panorama is a standard format. They consist of 4-6 images with 90^o field of view put next to each other. The format is convenient because it allows the creation of virtual views with only one linear transformation (homography) for each side of the cube. Its deficiency is that the number of pixels representing a viewing direction varies strongly between the center and the corner of a cube side. A cylindrical (spherical) panorama can also be used for a complete panorama. It reduces the pixel density problem, but requires the computation a homography for every column of pixels (every pixel) to project from the panorama onto a virtual view. For both cases, a compact representation is crucial wherever many users interact with the panorama video at the same time over the network. The affect storage space and memory usage as well, which may also be a performance concern.

Panorama Systems. Panorama systems that allows a user PTZ operations on a virtual camera have been developed both in research [4–12] and industry [13, 14].

PTZ cameras existed for a long time on static panoramic images and are quite commonly used in services like street maps and photosynth. However, the afore-mentioned systems deliver video experiences with similar interface. This adds a lot of challenges to the systems and several other dimensions to the problem of interactive experience.

[8] and [4] present a system for live/real time production of broadcast video using PTZ cameras. [8] specifically focus on efficient interfaces to create a live virtual camera for a single producer. [10] discuss virtual PTZ cameras to control/steer a robotic PTZ camera. The advantage of using a robotic PTZ camera is that it can use optical zoom and hence, provide high resolutions even at higher focal lengths. However, using a robotic PTZ camera exclusively limits the number of users that can control the camera to *one*.

[11] used virtual PTZ cameras to follow the speaker in lecture recording. Also [15] focus specifically on lecture videos. At an abstract level, indoor applications are similar to outdoor ones where the interactive experiences is concerned. The photometric challenges in outdoor applications are, however, not comparable to those of indoor scenes due to variable lighting and greater depths, and can affect the user experience drastically. [16] provide a good way of recording outdoor panoramas using HDTV cameras. Some works [17, 18] look at the system aspects of panorama capture systems.

Only a few works [18, 19] discuss the distribution of the live panorama video and even those lack a complete evaluation. Most industry projects transfer the entire panorama before starting the interaction, thus leading to a not *true-live* component. However, YouTube 360 has just released the first 360^o videos that deliver 4-sided cube-panorama videos stitched into a single video stream and allow pan and tilt operations (no zoom) in the Chrome browser.

Streaming Options. Tiled video can be processed into an individual stream for each viewer on the server side [20], but this approach does not scale to a large number of concurrent viewers who can chose individual views. In the latter case, the tiles that cover the user's view are delivered and processing occurs on the receiver side. We are not aware of a discussion of the options for this.

All distributed tiling systems face the challenge of user interaction that changes the user's view rapidly, requiring new covering tiles between two consecutive frames. Users can notice a delayed reaction to their interaction within a few milliseconds [21]. To avoid this latency, tiling systems that extract views on the receiver side choose to retrieve all tiles (within interaction range) at all times, but at a less than perfect quality to save bandwidth.

HAS is well-suited for this multi-quality delivery because it can delivery multiple quality levels to large audiences with the help of standard Web caches to increase scalability. However, retrieval decisions can only been made on segment boundaries, which means that visual quality can be reduced for several frames after user interaction affects the required tiles.

Faster quality improvement could be achieved by downloading a higher quality version of a segment that comes into visual range, decode it, skip frames that have already been played out at low quality, and continue with high-quality frames. The technique puts sudden high demands on download bandwidth and decoding. Alternatively, Scalable Video Coding (SVC) Mid Grain Scalability (MGS) could be combined with HAS [22]. Quality could be increased by retrieving an enhancement layer, which puts less load on bandwidth, and allows the receiver to improve frame quality immediately after skipping to the correct frame in the enhancement layer. However, an H.264 SVC-encoded video has 10% bandwidth overhead per enhancement layer compared to a non-scale video of the same quality [23].

Push-based streaming systems are an alternative because they can encode each tile as a continuous stream. Solutions that require multicast [24, 25] cannot be used on a large scale due to the lack of IP multicast. But also in a unicast solution, a push server can respond to a receiver's request for higher quality within one RTT of a user request. One method works by updating SDP [26], which can switch the unicast delivery of layers on and off, but of course, the SVC overhead mentioned above applies here as well. An even faster method is based on RTP [27], which can send a bit-rate request and instruct the server to send new Intra frame as soon as possible. This option is interesting, as it works either with SVC (suffering the mentioned overhead), with non-layered codecs but live encoding (or transcoding) at the sender, or a set of parallel streams where switching is supported through SI/SP frames [28]. The overhead of the SI/SP method lies between the other two approaches. All of these RTP-based methods

have in common that packet loss can occur, and it is therefore today usual to use MPEG 2-TS packaging [29], but this in itself incurs a 20% bandwidth overhead [30].

Considering that all of the approaches demand that the base-layer quality of all reachable tiles is streamed at all times, the bandwidth overhead of the various alternatives to HAS seemed too large for our scenario. We have therefore chosen a HAS with 1-second segments and discuss the quality implications of the qualities switching delay below.

Tiling Approaches Using HAS. Even though not directly related to the cylindrical/spherical panorama systems that provide free PTZ camera movement, there are some works [20, 31–33] that provide an approximate interaction. [31] discuss tiling in interactive panorama video. However, their panorama is a perspective one and the virtual camera performs merely cropping, which is identical to cropping from a high-resolution video. Similarly, [20] provide zoomable playout on mobile devices for bigger resolution videos. [32] present an approach for the zoomable video where the tiles are optimally selected and sent from the server side. [34] performed a user study to determine the effect of tiling on the zoomable video presentation. Except for [31], these works do not support a completely random PTZ camera. [33] present a similar system where the tiles are encoded at multiple qualities and retrieved depending on the current view, however they do not discuss smooth random movement. Their interface is similar to that of a zoomable video, where you can pick a portion of the entire video presented in a thumbnail and that part is cropped from the full resolution and presented. Hence, to our knowledge, our work is the first to handle the problem of tiling and discuss its trade-offs in the context of a random PTZ camera on a cylindrical panorama texture.

III. System Overview

Our panorama system is currently running live at two different locations. The tiling generation and retrieval operations are highlighted in figure 2. All components run in real-time, and the user can thus control the virtual camera during a live stream.

Server Side. Cylindrical panorama images are generated from five 2K cameras whose the shutters and exposures are perfectly synchronized. The seams are calculated dynamically for every frame. The frames are divided into 64 tiles (8x8), and one video stream is generated for each tile. Each video tile is encoded into 1-second segments at multiple qualities (and bit rates) using *libav* and *x264*. Each tile can then be requested individually by client using HAS.

Client Side. Once the different quality tiles are made available on the server, the client fetch tiles and generate the virtual view from the retrieved panorama. The task of the client is to retrieve high quality tiles for the virtual view and lower quality tiles for the surrounding tiles. Thus the client is able to supply the user with a high quality virtual view video, while at the same time trying to save bandwidth compared to the full quality panorama retrieval approaches discussed in the previous section.

However the system must fetch, spatially, every tile in the panorama video, whatever might be the quality. This way, the system can still provide data if the user interactively moves the virtual camera compared to presenting black areas or a static image if none of the surrounding tiles are retrieved at all. To accomplish this, the client is designed as shown in figure 3. There are four major components in the client system, (i) a File Manager, (ii) a Decoder, (iii) a Renderer and (iv) a Tile Selector.

File Manager. The File Manager component is responsible for requesting appropriate tiles in a given quality from the server (determined by the Tile Selector described below). The

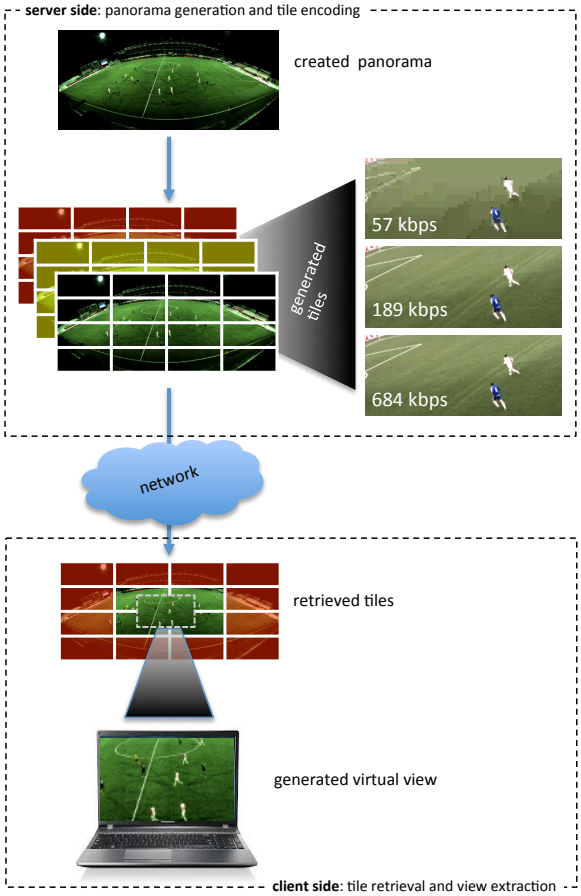


Fig. 2: At the server side, we divide the generated panorama video into 8x8 tiles, and then encode each tile in different qualities. The client retrieves tiles in certain qualities based on the current position of the virtual camera (full quality tiles for the virtual view and low quality (red) tiles outside the field of view).

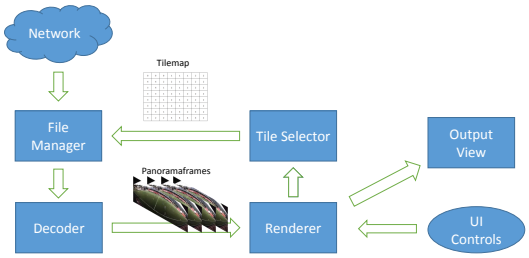


Fig. 3: The architecture of a client supporting tiling.

byte stream is transferred as fetched and forwarded straight to the Decoder module instead of saving it as a local file, thus bypassing the disk.

Fig. 4: Sample output frame from the decoder module.

Decoder. Once the tiles are available from the File Manager, the Decoder module starts decoding frames and pushing it to a common panorama texture. Since the tiles are spatially independent of each other, this process is heavily parallelizable. The operations are, frame-synchronized to avoid placing a frame from two different tiles at different time instants into the same panorama frame. Figure 4 shows an example frame[1] where one can observe that the panorama frame is reconstructed from different quality tiles.

Renderer. As soon as a panorama frame is decoded, it is pushed to the rendering module. This module is responsible for creating the virtual views using the PTZ parameters. In addition, it provides User Interaction or virtual view controls. In most interactive systems, the functionality of the Renderer is limited to this. However, in order to support tiling, we need to save the information of the panorama parts that are currently being viewed. This information is transferred to the Tile Selector module which again uses the information to select the tile qualities for the next iteration.

Tile Selector. Once a frame is displayed, the panorama location from where the current view is extracted is transferred to the Tile Selector from the Renderer. This information plays a crucial role in selecting the next tile set. Finally it important to point out that all these modules need to perform in real-time to provide a smooth interactive experience to the user while keeping the bandwidth consumption at a minimum required level. One can observe that this can be challenging task at the Decoder module, where several videos are expected to be decoded concurrently in real-time and also frame-synchronized.

IV. Quality Selection Approaches

As described before, the Tile Selector is responsible for determining appropriate qualities (and bitrates) for the different tiles, and adapt according to the viewer movement. Let $Q = \{q_0, q_1, \ldots, q_{n-1}\}$ be the set of n available quality levels and T_i be the tile quality at tile i, then the problem can be written as a simple labelling problem in equation 1. The qualities are in a decreasing order where q_0 is the highest quality tile.

$$T_i = q \quad \text{where} \quad q \in Q \tag{1}$$

[1]Due to possibly limited resolution of printers, it is recommended to analyze the images on screen.

There are several ways to perform this labelling, which will ultimately influence the bandwidth consumed and the user experience of the system. A *binary tile occupancy map*, containing information on which tiles are currently used to generate the virtual view, is used in the labelling process. The binary occupancy map has $B_i = 1$ at tile i when the view needs pixels from the tile i on the panorama. Even using the same binary occupancy map, there are several ways to select a tile quality, and below we briefly outline some of the algorithms evaluated in this study. The three first algorithms make a binary decision between a predefined, yet configurable, high or low quality. The last approach allows for a gradual (multi-level) decrease of quality depending on the importance of a tile.

A. Binary

The binary approach is a simple approach where high quality is assigned to the required tiles and low quality to the ones that are not required (figure 5). Using the binary occupancy map described above, this becomes rather trivial. Hence, the binary approach can be formulated as following:

Fig. 5: Tiled binary.

$$T_i = \begin{cases} q_h & \text{if } B_i = 1 \\ q_l & \text{else} \end{cases} \tag{2}$$

The only requirement in this case is to have $l > h$. However, the choice of exact quality levels can be considered as tuning.

B. Rescaled

A commonly used approach in research in terms of tiling is to send a low quality base thumbnail video and provide only the required high quality tiles [31, 32] (figure 6). To create the thumbnail video, the source video is down-scaled and stored. During the process of virtual view generation, the pixels from the available high quality tiles are used. For the pixels where the high quality data is missing, the thumbnail video is up-scaled and used, which can be considered as low quality tiles.

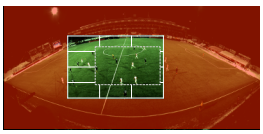


Fig. 6: Thumbnail.

C. Prediction

When a user moves the virtual camera, there is a chance that the view will be generated by some low quality tiles since the tile quality is only changed at the segment boundary. In order to lower the probability that this occurs, it is beneficial to try to predict future movements and retrieve a higher

Fig. 7: Predicted.

quality tile if there is a high probability that the user moves the view into this tile (figure 7, i.e., similar to the tiled binary, but where the high quality area is enlarged according to the prediction). In this respect, it is beneficial to predict the path across several frames in future. There are several models available for prediction. However, to keep the comparison to the state-of-the-art consistent, we used the Auto Regressive Moving Average (ARMA) prediction [31]. Here, let θ_t be the position and $\delta\theta_t$ be the velocity of the view at time instant t. The velocity at the current instant is estimated as,

$$\delta\theta_t = \alpha\delta\theta_{t-1} + (1-\alpha)(\theta - \theta_{t-1}) \tag{3}$$

Then, the future position at $t+f$ is estimated as

$$\hat{\theta}_{t+f} = \theta_t + f\delta\theta_t \tag{4}$$

where f is the number of frames predicted in future. This can be used straight away to figure out a future binary occupancy map. This map can be used in any of the approaches mentioned here. But for the sake of comparison, we use the Predictive approach along with the Rescaled approach [31].

D. Pyramid

The pyramid is a complex scheme where we chose qualities intelligently with a gradually decreasing quality according to the distance from the virtual camera (figure 8). Here, we introduce the term *priority* (p_i) that varies in $\{0,1\}$, where 0 being highly important to 1 being least important.

Fig. 8: Pyramid.

Depending on the importance, we fetch the corresponding quality. But there is another catch. If we just decide on the importance, we might end up fetching high quality tiles for a lot of tiles for a zoomed-out virtual view. Here, the maximum quality level (q_{max}) comes into picture. This quantity depends on the number of high priority tiles. We select q_H as the quality level to be used when all the tiles are being used for the virtual view.

$$q_{max} = (\frac{\sum_{i\in T} b_i}{N})q_H \tag{5}$$

$$T_i = \begin{cases} q_{max} & \text{if } b_i = 1 \\ q_{max} + p_i(n - q_{max} - 1) & \text{else} \end{cases} \tag{6}$$

After q_{max} is calculated, we count the occupancy of the neighbourhood and then assign that as its p_i as shown in equation 7. As one can observe there are several tuning parameters. One is the q_H, which determines the quality at a certain zoom level. The second is the selection of the neighbourhood itself which can be determined by the weights α_j. We can either make the weights to be isotropic or anisotropic. Given the fact that one is more prone to pan than to tilt, anisotropic weights can lead to similar performance as the isotropic one while consuming less bandwidth.

$$p_i = 1 - \frac{\sum_{j\in\mathcal{N}} \alpha_j b_{ij}}{\sum_{j\in\mathcal{N}} \alpha_j} \tag{7}$$

V. Evaluation Methods

The problem of bandwidth reduction is a strict trade-off of two conflicting constraints. One constraint is the bandwidth itself, which can be measured straight away as the rate of data transferred. The second constraint is the quality of experience, which is not trivial to measure. When developing approaches, we need to consider how well the approaches are performing with respect to these constraints and which approaches provide the best trade-off. We compare the two different pipelines in figure 9, and we use the final output (the rendered virtual view) for comparison.

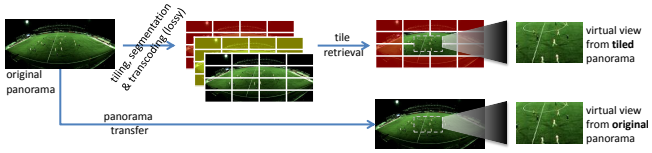


Fig. 9: Pipeline differences: original vs. tiled panorama.

We assessed the quality selection schemes described in section IV by comparing their performance on the entire first half of a soccer game (approximately 47 minutes). We used five quality levels with increasing Constant Rate Factor (CRF) values, the levels along with the CRF values are $q_0(21)$, $q_1(24)$, $q_2(30)$, $q_3(36)$ and $q_5(48)$. For the investigation, we compared the quality of four pre-generated sequences of PTZ operations, called *paths*. We created four paths whose pan/tilt operations follow the general soccer game flow, but whose zoom varies as described and labelled in table I. The quality selection methods were labelled as shown in table II.

Label	Path
s1	The virtual camera is severely zoomed-in
s2	The zoom is at a medium level
s3	An overview video where the view is zoomed-out
s4	A dynamic zoom factor depending on the situation

TABLE I: Paths: sequences of PTZ operations

Label	Approach
l1	Binary with q_0 and q_4
l2	Binary with q_1 and q_3
l3	Rescaled with no prediction
l4	Rescaled with 100 frames prediction
l5	Pyramidal with isotropic weights
l6	Pyramidal with anisotropic weights
l7	Pyramidal with isotropic weights (different parameters)
l8	Full Panorama input (no tiling)

TABLE II: Labelling of approaches for analysis

A. Quality metric

The challenge for objective video quality metrics has so far been to match subjective viewing experiences for videos of finite duration (8-12 seconds). Objective methods that try to solve this challenge and that have undergone rigorous independent testing [35] are meant for constant-quality videos (with uniform disturbances). They can estimate QoE if degradation in a video spans several frames and work well for individual HAS segments. However, they may not be suitable when the user is presented with a view that is stitched from several independently adapting HAS video tile. In this scenario, only parts of the video suffer from distortion and there are updates that can instantly change the degradation.

To compare the quality selection schemes in our paper, we have therefore conducted a user study to assess whether the image similarity metric SSIM [36] or OpenVQ, an independent implementation of a perceptual video quality metric described in ITU-T J.247 [37, Annex B], provide good estimates of subjective quality assessment. All user study experiments are performed using 12-second excerpts[2] from the 4 sequences mentioned in table I.

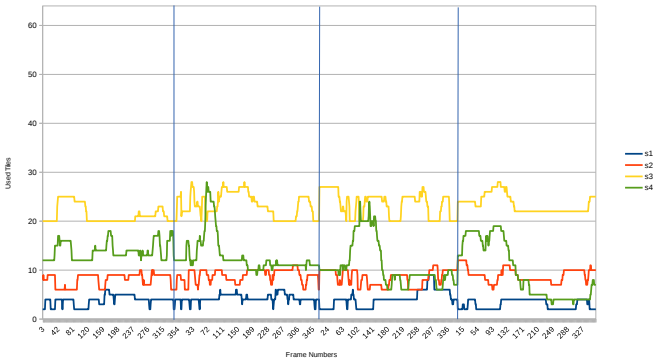


Fig. 10: Total number of tiles used, out of 64 possible, per frame in sequences of 12-second durations at 4 pre-determined time instants.

OpenVQ: Perceptual Evaluation of Video Quality (PEVQ) is a full-reference algorithm that outputs mean opinion scores (MOS) as an objective video quality metric. After evaluation by the Video Quality Experts Group (VQEG), it has become part of the standard ITU-T J.247 [37]. Out of all candidates, PEVQ achieved the best error rate with respect to subjective studies conducted by two independent institutes.

Unfortunately, PEVQ is not freely available for researchers, so we used OpenVQ, which is based on J.247 Annex B, but not a one-to-one implementation of PEVQ. The *patented* temporal alignment has been dropped, because neither HAS nor RTP-based streaming suffer from temporal misalignment. Furthermore, flaws in the formulas in J.247 Annex B force a rather loose interpretation. The dataset used for evaluating J.247 candidates is not publicly available, but with a ground truth of ICCRyN datasets [38–40], OpenVQ achieves results close to those reported for PEVQ in J.247.

SSIM: The Structured Similarity Index (SSIM) [36] is a metric for assessing differences between images. It is supposed to model human subjective experience quite well, but [41] have demonstrated that this fails for a variety of possible image degradations. In spite of this, SSIM is even used for estimation the quality of video. $x264$ makes encoding decisions based on it, and [42] construct a video quality assessment tool based on it.

PSNR: A commonly used measure in evaluating video qualities is PSNR. As [43] explains, it is solely a pixel difference metric, and quite unrelated to subjective experience. Already [44] explained its limits, while [45] have clarified that it can predict human preference in one particular case: when the same content has been encoded with different compression strengths.

[2]The same sequences are attached along with the paper, however compressed due to the limitation on space.

B. Assessing objective metrics

The measure that we require differs from the ground truths that have been used in assessing current video quality metrics. As mentioned in Section V-A, we have tiled videos following a HAS model. An adaptation decision for each tile is made once a second. We do not aim at generating a single quality value for an entire 47-minute test case, because we have not found any valid basis for doing so in the literature. Instead, we verify how well the above objective metrics describe user experience on a second-by-second basis.

1) Study design: We compared the results of the objective metrics with subjective evaluations across a range of tiling approaches. The user study was designed to investigate two aspects of the subjective perception of quality. We consider the noticeability of quality distortions and the experienced annoyance related, but distinct. We ran two consecutive experiments, one to address the detection of tiling distortions, and one to address the annoyance resulting from the distortions. In addition, we included five-point absolute category rating scales for subjective assessments of overall video quality, adhering to ITU-T P.911 [46].

In both experiments, participants watched sequences with duration of 12 seconds extracted from the sequences described at the beginning of section V. These were originally chosen as representations of different football scenarios and hence provided variety to participants and served to increase generality. Since all sequences included pre-recorded camera panning and zooming movements, our final stimulus collection contained sequences with frequent tile shifts and varying changes in compression rate and video quality. In the detection experiment, we instructed participants to pay attention to the the quality of the presented sequences and to push down the spacebar the moment they noticed a change for the worse, holding it down for the entire duration of the quality drop. The annoyance experiment followed the same procedure, only changing the instructions to ask participants to push down the spacebar while they experienced annoyance due to low video quality. At the end of each sequence, participants rated the overall video quality on a 5-point scale ranging from ”bad” to ”excellent”.

In order to secure a sufficient number of participants, we recruited the help of crowdsourcing workers. This approach requires some extra methodological considerations due to challenges that concern lack of task adherence and comprehension, and in turn, reduced data consistency [47, 48]. Thus, we initially conducted 3 pilot studies to ensure that the experiments were presented in a succinct, but understandable, format. The first was completed by colleagues and students, the following two on crowdsourcing platforms Microworkers and Crowdflowers. Following each pilot, we adapted the experiments according to the received feedback. For the final study, we used Microworkers and collected data from a total of 200 different participants. Although we implemented quality measures such as gold samples and majority votes, the highly subjective nature of the task did not allow more than the most basic automatic filtering to exclude non-complying individuals. We excluded only participants who failed to complete the

experiment, and on manual inspection removed participants who had obviously attempted to circumvent the experimental tasks, altogether 15%. All other potential exclusion criteria were found to potentially exclude valid human perceptions as well.

We then calculated the agreement between participants' quality ratings for each sequence using the Fleiss Kappa statistic [49]. Because this statistic depends on the number of raters and comparisons [50], we consider it in the context of the possible minimum and maximum values, which are established at -0.80 and 1. For the detection experiment, inter-rater agreements varied between 0.22 and 0.37 across the different sequences and quality conditions. The annoyance experiment yielded values between 0.24 and 0.39. With respect to the arguably subjective and variable measures of detection and annoyance, we judge these positive agreement scores as indications that participants adhered to the task at hand.

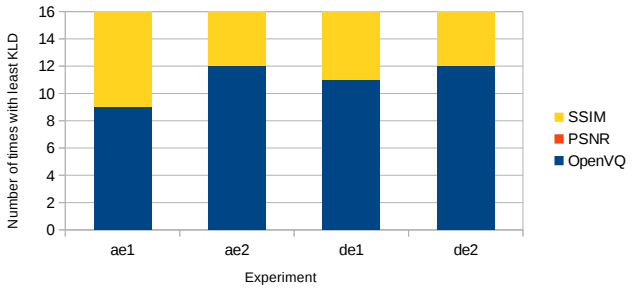


Fig. 12: Number of times a metric had the least divergence from the user input in each task of the experiments among OpenVQ, PSNR and SSIM.

(a) l3s4 - PSNR: 30.39 (b) l7s4 - PSNR: 30.89

Fig. 13: Example of severe differences within a frame (319), leading to similar PSNR values.

2) Performance of Evaluation Metrics: The analysis of user studies for perception is always challenging, especially when the users are expected to provide time-varying input. For example, our study aims at recording perception differences among users, but records also response time differences between them. It may be possible to counteract this by assuming that users' opinions correlate and maximizing cross-correlation by time-shifting all inputs. However, due to the weakness of this assumption, we ignored response times and averaged user inputs across all users.

The results of our user study show a reasonably strong relation between the user input and OpenVQ, but also SSIM. We used Kullback-Leibler-divergence (KLD) [51] to estimate

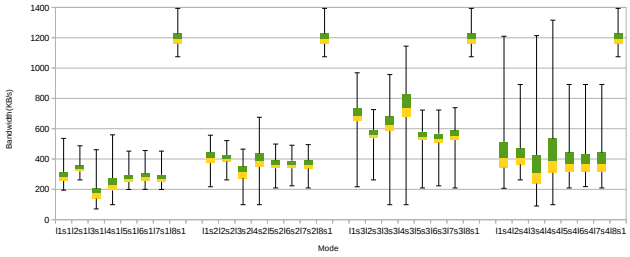


Fig. 14: A boxplot of the bandwidth consumed in (KB/s) for different approaches over 2.834 seconds(47 min) representing the first half of a game.

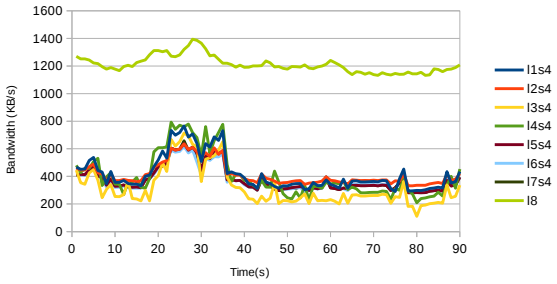


Fig. 15: Bandwidth profile for 90 seconds duration in the middle of $s4$.

the information loss in approximating subjective results with the objective metrics, and KLD stays below 0.05 for path $s1$ and below 0.01 for the other paths. Figure 12 shows that both OpenVQ and SSIM can be closer to the average subjective ratings[3].

Although PSNR is unsuitable as a metric of visual quality (also quite easily shown to fail in the case where high- and low-quality tiles are merged into a single view like in figure 13), we present also PSNR results because they expose unexpected properties of the 1-second video segments. The PSNR results in figure 11a exposed regular severe degradation of the last frame in each 1-second segment. Although this is not noticeable to a human observer even when single-stepping through the video, it is clear evidence of problems in *ffmpeg* or the way in which we use it.

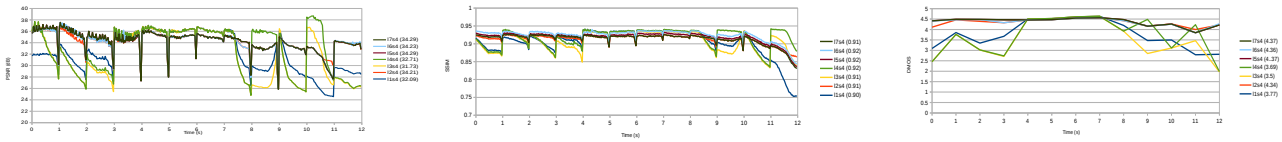
VI. Evaluation Results

In this section, due to space limitations, we mostly discuss about the $s4$ sequence which is representative of the real-life virtual camera operation. However, using the other sequences we can observe zoom specific results. For example, $s1$ consumes least amount of bandwidth due the required low number of tiles. We can also observe from $s3$ that when the user is interested in overview of the field, there is no need to fetch highest quality tiles.

A. Bandwidth

A simple way to determine the cost of network delivery is to measure the bandwidth. The most commonly used measurement is the average bandwidth along the entire run.

[3]Standalone HTML/JS plots for each study are attached to the paper for reference. Any recent web browser can be used to explore them.



(a) PSNR (b) SSIM (c) OpenVQ

Fig. 11: Different variation over the quality metrics across the 12 second clips. For reference, the average of each metric across the 12-second duration is also presented in parenthesis for each approach.

Figure 14 presents a box-plot of each approach using different paths for the first half of the game. However, the interactive services can have different bandwidth requirements at different instances. Therefore, we use the running bandwidth profile to evaluate the performance of the approaches. Figure 15 shows the bandwidth profile for all approaches for a 90-second duration at 1000 seconds into the game. We can observe that there is some correlation with the number of tiles used at that time instant, which can be seen in figure 17.

Tiles	Total	out21	out24	out30	out36	out48
2x2	17G	7.5G	5.0G	2.2G	1.2G	528M
4x4	18G	7.7G	5.3G	2.5G	1.5G	821M
8x8	23G	8.7G	6.4G	3.7G	2.4G	1.8G

TABLE III: Size of the data for a soccer video of 6297 seconds using different tile granularity when compressing each tile with CRF values of 21, 24, 30, 36 and 48 on 1 second segments. In comparison, the size of the non-tiled panorama using the same segment length is 7.3GB.

The methods are tuned to provide similar bandwidth with slight variations depending on the number of tiles used. However, it is quite evident that the approaches using highest quality tiles wherever required will have high bandwidth usage when a lot of tiles are used in the view. This can be seen in the great bandwidth requirement for $l1$, $l3$ and $l4$. However, over the long run of the random zoom sequence ($s4$), which is probably most representative of a real scenario, the bandwidth consumption for all approaches is quite similar. For an estimate of the costs on the server side, we present the total disk space occupied by the tiled segments in table III. Irrespective of the approach, it can be observed that the bandwidth savings are quite high, sometimes reducing requirements to 25% of the full panorama. Hence, it becomes important to evaluate the approaches for quality.

B. Quality

So far, no approach exists that can provide best visual quality and low bandwidth usage at the same time during the entire virtual view operation. However, some approaches, especially the pyramidal ones, can provide decent bandwidth savings and also acceptable quality most of the times. From figure 16, we can observe that all methods suffer from quality degradation at times. The predictive approach is functional and provides improvement only when the actual positions match with the predicted positions. However, with a completely random operation, this can be challenging even with sophisticated algorithms. Moreover, the prediction algorithms seem to be the most expensive on bandwidth. However, not all high quality tiles fetched are used for extraction of the virtual view.

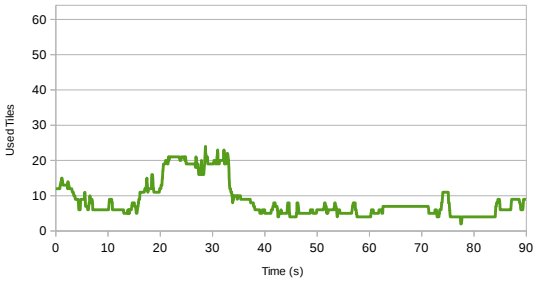
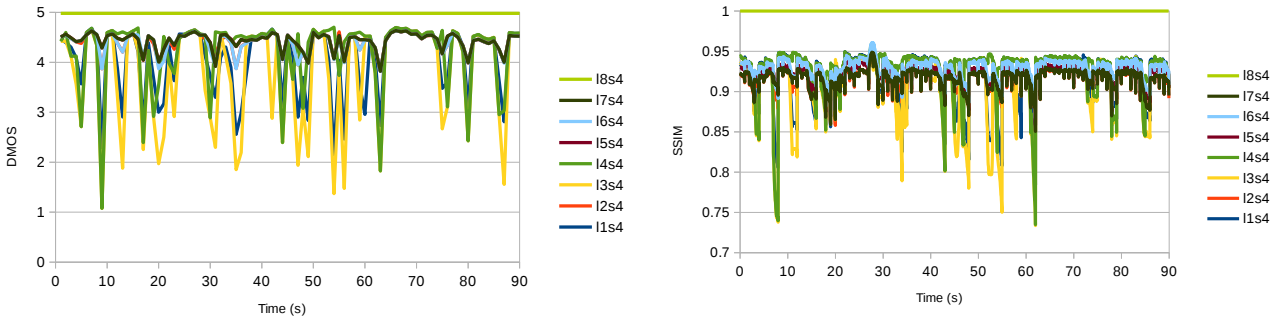


Fig. 17: Number of used tiles across 90 seconds at 1000 seconds into the soccer game for $s4$.

In figure 16, we can see that a value of 0.93 for SSIM and 4.5 for DMOS runs along the time with drops depicting the quality changes during the virtual camera movement. These values imply that the visual quality of the output tiled virtual view is on par with the original. Even in the drops, we observe that the pyramidal approaches perform better than the others. However, SSIM and OpenVQ are full-reference quality measures, which implies that the evaluation can only be carried out with the presence of the high-quality virtual view. However, there are ad-hoc measures that one can collect in the background without much resource consumption and that can provide some insight into the quality of a virtual view.

Furthermore, [31] introduce the notion of missing-pixel percentage to evaluate the accuracy of their prediction and thus the quality of the virtual view. A *missing pixel* is a pixel in the virtual view where the corresponding high quality panorama data is not available for rendering. A percentage of missing pixels can be calculated against the total number of pixels in the virtual view. The average percentage of missing pixels across several seconds is used to evaluate various approaches in [52]. However, in the previous tiling approaches [32, 52], where the selection is mostly a binary process using either a high quality tile or a low quality thumbnail, the missing pixel percentage can contain a lot of information about the quality. But, we also include pyramidal approaches in the evaluation and they use multiple quality levels. Hence, we propose using a *Tile Histogram*. In a frame of the output virtual view, we count the percentage of pixels fetched from each tile.

[52] provide evaluation results also as average percentage of missing pixels for a 480×240 cropped view of 2560×704 pixels panorama. This 6.7% ratio is equivalent to using 4 tiles in a 64 tiled panorama ($s1$ from figure 10), in which



(a) OpenVQ variation (b) SSIM variation

Fig. 16: Measured variation across 90 seconds at 1000 seconds into the soccer game for $s4$

Label/Sequence	s1	s2	s3	s4
l3	20.22	10.20	2.60	9.44
l4	18.20	8.56	1.94	6.53

TABLE IV: Average percentage of Missing pixels measurements over the entire first half of the game

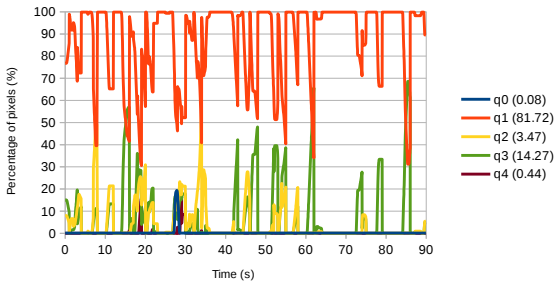


Fig. 18: Pixel histogram across 90 seconds along with the average percentages for each level in the parenthesis

5

case the average percentage of missing pixels of around 20%, from table IV, is coherent with their results. From the table IV, we can also observe that the missing pixel percentage varies depending on the zoom. An example profile of a pixel histogram is plotted in figure 18. One can observe certain correlation between the pixel histogram profile and the variations in quality observed from OpenVQ or SSIM. Ad-hoc metrics like these can be used a reference to check the quality on the fly during the process. However, full-reference metrics provide the most accurate insights into the quality variations.

VII. Discussion

The study presented in this paper uses HAS as the delivery method for tiled panorama video, although this is not the only option. The QoE provided by systems that can adapt visual quality within a single RTT should be explored as well. However, as we discussed in section II, there is a considerable bandwidth penalty associated with push-based unicast solutions. Nevertheless, multicast is not widely available, and we have therefore investigated a HAS-based approach.

From the analysis of quality metrics and bandwidth profiles for different movement path of a virtual camera, we make several observations for the various approaches. We find that *pyramidal* approaches provide a stable quality across different zoom factors and random movements, i.e., it is a good tradeoff between bandwidth savings and perceived video quality. When only a little portion of the panorama is used, we find that the *rescaling* approaches take up the least bandwidth, but the loss in quality is significant. The *prediction* results are not specially impressive when using a general prediction algorithm, and [52] found that even context-based prediction does not lead in much improvement.

The adaptation strategies evaluated in this paper try to adapt the quality of a tile according to the movement of the view in order to have as good quality as possible in the area of the panorama used by the virtual camera to generate the view. There are, however, numerous works that similarly try to optimize the quality of traditional (non-panoramic) HAS streaming according to available resources. Clients of all major HAS variants, i.e., Apple HLS, Microsoft Smooth Streaming and DASH, have algorithms trying to have a high, stable quality. Additionally, researchers have presented approaches trying to optimize the segment retrieval, e.g., according to buffer occupancy [53] and consistent visual quality [54]. However, including this in the current study is out of the scope of this paper, but it is an interesting topic to pursue in the future combining optimal tile quality according to both the virtual view and the available resources.

We have also explored and analysed the effect of different segmented streaming approaches on quality for arena sport scenario where the movement on the field is little compared to the entire field. It would be definitely interesting to explore them in different scenarios like the ones with *details but static* and *details with large movements*. However, these scenarios may require different treatment to achieve a good tradeoff between quality and bandwidth usage.

VIII. Conclusion

We have presented multiple approaches for tiling that can exploit the coding efficiency of H.264 to reduce bandwidth requirements for an interactive live PTZ system. We have evaluated the approaches using several methods and compared these methods for their closeness to subjective perception.

Based on our experimental results, we provide several conclusions. Overall, our results prove that pyramidal approaches

reduce the bandwidth requirement and at the same time provides a similar QoE compared to a full-quality, non-tiled panorama system. Furthermore, utilizing the CRF parameter of H.264 provides better bandwidth savings and better visual quality compared to up-scaling a thumbnail video when the panoramic system is static and the movement in the scene is little compared to the scene itself. This is a rather common scenario for arena sports like Rugby, Soccer, Hockey, Cricket etc. Since a subjective study is a time-consuming and expensive way to evaluate the approaches, there is a rise in objective evaluation. We conclude that traditional evaluation methods will fail to correlate well with the subjective assessment of the experience, and a new metric, OpenVQ, closely capture subjective ratings.

Both the approaches and evaluation methods can be used with other interactive live PTZ camera systems as well. However, the tiling approaches, specially the quality levels, will require some parameter tuning specific to the application to gain optimal performance. Finally, we provide an open-source implementation of the OpenVQ estimation tool for further usage by researchers.

REFERENCES

[1] Statista, "Top 10 Internet Traffic Services," 2015, http://www.statista.com/chart/1620/top-10-traffic-hogs/.

[2] Hypebot, "Video Wars Youtube versus Facebook," 2015, http://www.hypebot.com/hypebot/2015/03/the-video-wars-youtube-vs-facebook.html.

[3] J. Sanchez-Hernandez, J. Garcia-Ortiz, V. Gonzalez-Ruiz, and D. Muller, "Interactive streaming of sequences of high resolution jpeg2000 images," *Multimedia, IEEE Transactions on*, vol. 17, no. 10, pp. 1829–1838, Oct 2015.

[4] W.-K. Tang, T.-T. Wong, and P.-A. Heng, "A system for real-time panorama generation and display in tele-immersive applications," *Multimedia, IEEE Transactions on*, vol. 7, no. 2, pp. 280–292, April 2005.

[5] S. Tzavidas and A. Katsaggelos, "A multicamera setup for generating stereo panoramic video," *Multimedia, IEEE Transactions on*, vol. 7, no. 5, pp. 880–890, Oct 2005.

[6] H.-Y. Shum, K.-T. Ng, and S.-C. Chan, "A virtual reality system using the concentric mosaic: construction, rendering, and data compression," *Multimedia, IEEE Transactions on*, vol. 7, no. 1, pp. 85–95, Feb 2005.

[7] Q. Zhao, L. Wan, W. Feng, J. Zhang, and T.-T. Wong, "Cube2video: Navigate between cubic panoramas in real-time," *Multimedia, IEEE Transactions on*, vol. 15, no. 8, pp. 1745–1754, Dec 2013.

[8] E. Foote, P. Carr, P. Lucey, Y. Sheikh, and I. Matthews, "One-man-band: A touch screen interface for producing live multi-camera sports broadcasts," in *Proc. of ACM MM*, 2013, pp. 163–172.

[9] C. Fehn, C. Weissig, I. Feldmann, M. Muller, P. Eisert, P. Kauff, and H. Bloss, "Creation of high-resolution video panoramas of sport events," in *Proc. of IEEE ISM*, Dec. 2006, pp. 291 –298.

[10] P. Carr, M. Mistry, and I. Matthews, "Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording," in *Proc. of ACM MM*, 2013, pp. 193–202.

[11] X. Sun, J. Foote, D. Kimber, and B. Manjunath, "Region of interest extraction and virtual camera control based on panoramic video capturing," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 981–990, 2005.

[12] W.-K. Tang, T.-T. Wong, and P.-A. Heng, "A system for real-time panorama generation and display in tele-immersive applications," *IEEE Transactions on Multimedia*, vol. 7, no. 2, pp. 280–292, April 2005.

[13] Camargus, "Premium Stadium Video Technology Infrastructure," https://www.youtube.com/watch?v=SO32pEgCeDI.

[14] Teamcoco, "Conan 360: The New Angle on Late Night," 2014, http://teamcoco.com/360.

[15] T. Yokoi and H. Fujiyoshi, "Virtual camerawork for generating lecture video from high resolution images," in *Proc. of IEEE ICME*, July 2005.

[16] W. Xu and J. Mulligan, "Panoramic video stitching from commodity hdtv cameras," *Multimedia Systems*, vol. 19, no. 5, pp. 407–426, 2013.

[17] P. Carr and R. Hartley, "Portable multi-megapixel camera with real-time recording and playback," in *Proc. of DICTA*, 2009, pp. 74–80.

[18] H. Kimata, M. Isogai, H. Noto, M. Inoue, K. Fukazawa, and N. Matsuura, "Interactive panorama video distribution system," in *Proc. of ITU Telecom World*, Oct 2011, pp. 45–50.

[19] O. Niamut, J. Macq, M. Prins, R. Van Brandenburg, N. Verzijp, and P. Alface, "Towards scalable and interactive delivery of immersive media," in *Proc. of NEM Summit*, 2012, pp. 69–74.

[20] F. Liu and W. T. Ooi, "Zoomable video playback on mobile devices by selective decoding," in *Proc. of PCM*, 2012.

[21] K. Raaen, R. Eg, and C. Griwodz, "Can gamers detect cloud delay?" in *Proc. of NetGames*, 2014, pp. 1–3.

[22] I. Unanue, I. Urteaga, R. Husemann, J. D. Ser, V. Roesler, A. Rodriguez, and P. Sanchez, "A tutorial on H.264/SVC scalable video coding and its tradeoff between quality, coding efficiency and performance," in *Recent Advances on Video Coding*. Intech, 2011, pp. 3–26.

[23] C. Kreuzberger, D. Posch, and H. Hellwagner, "A scalable video coding dataset and toolchain for dynamic adaptive streaming over HTTP," in *Proc. of ACM MM-Sys*, 2015, pp. 213–218.

[24] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," *ACM CCR*, vol. 26, pp. 117–130, 1996.

[25] K. A. Hua, Y. Cai, and S. Sheu, "Patching: a multicast technique for true video-on-demand services," in *Proc. of ACM MM*, 1998, pp. 191–200.

[26] J. Rosenberg and H. Schulzrinne, "An offer/answer model with session description protocol (SDP)," RFC 3264 (Proposed Standard), Jun. 2002.

[27] S. Wenger, U. Chandra, M. Westerlund, and B. Burman, "Codec control messages in the RTP audio-visual profile

with feedback (AVPF)," RFC 5104 (Proposed Standard), Feb. 2008.
[28] M. Karczewicz and R. Kurceren, "The SP- and SI-frames design for H.264/AVC," *IEEE TCSVT*, vol. 13, no. 7, pp. 637–644, 2003.
[29] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar, "RTP Payload Format for MPEG1/MPEG2 Video," RFC 2250 (Proposed Standard), Jan. 1998.
[30] H. Riiser, P. Halvorsen, C. Griwodz, and D. Johansen, "Low overhead container format for adaptive streaming," in *Proc. of ACM MMSys*, 2010, pp. 193–198.
[31] A. Mavlankar and B. Girod, "Video streaming with interactive pan/tilt/zoom," in *High-Quality Visual Experience*, ser. Signals and Communication Technology, 2010, pp. 431–455.
[32] R. Guntur and W. T. Ooi, "On tile assignment for region-of-interest video streaming in a wireless LAN," in *Proc. of NOSSDAV*, 2012, pp. 59–64.
[33] H. Kimata, D. Ochi, A. Kameda, H. Noto, K. Fukazawa, and A. Kojima, "Mobile and multi-device interactive panorama video distribution system," in *Proc. of GCCE*, Oct 2012, pp. 574–578.
[34] H. Wang, V.-T. Nguyen, W. T. Ooi, and M. C. Chan, "Mixing tile resolutions in tiled video: A perceptual quality assessment," in *Proc. of NOSSDAV*, 2014, pp. 25:25–25:30.
[35] K. Brunnstrom, D. Hands, F. Speranza, and A. Webster, "VQEG validation and ITU standardization of objective perceptual video quality metrics," *IEEE Signal Processing Magazine*, vol. 26, no. 3, 2009.
[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
[37] ITU-T, "J.247: Objective perceptual multimedia video quality measurement in the presence of a full reference," 2008.
[38] Y. Pitrey, U. Engelke, M. Barkowsky, R. Pépion, and P. L. Callet, "Aligning subjective tests using a low cost common set," in *Proc. of EuroITV QoEMCS*, 2011.
[39] F. Boulos, W. Chen, B. Parrein, and P. Le Callet, "Region-of-interest intra prediction for H.264/AVC error resilience," in *Proc. of ICIP*, 2009, pp. 3109–3112.
[40] S. Pechard, M. Carnec, P. Le Callet, and D. Barba, "From SD to HD television: Effects of H.264 distortions versus display size on quality of experience," in *Proc. of ICIP*, 2006, pp. 409–412.
[41] R. Dosselmann and X. Yang, "A comprehensive assessment of the structural similarity index," *Signal, Image and Video Processing*, vol. 5, no. 1, pp. 81–91, 2011.
[42] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 121–132, Feb. 2004.
[43] S. Winkler, *Digital Video Quality: Vision Models and Metrics*. Wiley, 2005.
[44] J. L. Mannos and D. J. Sakrison, "The effects of a visual fidelity criterion of the encoding of images," *IEEE T. Inform. Theory*, vol. 20, no. 4, pp. 525 – 536, 1974.
[45] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of PSNR in image/video quality assessment," *Electronics Letters*, vol. 44, no. 13, pp. 800–801, 2008.
[46] ITU-T, "P.911: Subjective audiovisual quality assessment methods for multimedia applications," Geneva, pp. 1–46, 1998.
[47] F. Ribeiro, D. Florencio, and V. Nascimento, "Crowdsourcing subjective image quality evaluation," in *Prof. of ICIP*, 2011, pp. 3097–3100.
[48] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with mechanical turk," in *Proc. of CHI*, 2008, pp. 453–456.
[49] J. L. Fleiss, "Measuring nominal scale agreement among many raters." *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.
[50] J. Sim and C. C. Wright, "The kappa statistic in reliability studies: use, interpretation, and sample size requirements," *Physical therapy*, vol. 85, no. 3, pp. 257–268, 2005.
[51] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, pp. 79–86, 1951.
[52] A. Mavlankar and B. Girod, "Pre-fetching based on video analysis for interactive region-of-interest streaming of soccer sequences," in *Proc. of ICIP*, Nov 2009, pp. 3061–3064.
[53] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video streaming using a location-based bandwidth-lookup service for bitrate planning," *ACM TOMCCAP*, vol. 8, no. 3, 2011.
[54] Z. Li, A. C. Begen, J. Gahm, Y. Shan, B. Osler, and D. Oran, "Streaming video over HTTP with consistent quality," in *Proc. of ACM MMSys*, 2014, pp. 248–258.

Appendix F

[Conference] Automatic exposure for panoramic systems in uncontrolled lighting conditions: a football stadium case study.

[Authors:] **V. R. Gaddam**, C. Griwodz, and P. Halvorsen
[Published:] SPIE Electronic Imaging (EI), 2014.

Automatic exposure for panoramic systems in uncontrolled lighting conditions: a football stadium case study.

Vamsidhar Reddy Gaddam, Carsten Griwodz, Pål Halvorsen

University of Oslo and Simula Research Laboratory, Norway

ABSTRACT

One of the most common ways of capturing wide field-of-view scenes is by recording panoramic videos. Using an array of cameras with limited overlapping in the corresponding images, one can generate good panorama images. Using the panorama, several immersive display options can be explored. There is a two fold synchronization problem associated to such a system. One is the temporal synchronization, but this challenge can easily be handled by using a common triggering solution to control the shutters of the cameras. The other synchronization challenge is the automatic exposure synchronization which does not have a straight forward solution, especially in a wide area scenario where the light conditions are uncontrolled like in the case of an open, outdoor football stadium.

In this paper, we present the challenges and approaches for creating a completely automatic real-time panoramic capture system with a particular focus on the camera settings. One of the main challenges in building such a system is that there is not one common area of the pitch that is visible to all the cameras that can be used for metering the light in order to find appropriate camera parameters. One approach we tested is to use the green color of the field grass. Such an approach provided us with acceptable results only in limited light conditions.A second approach was devised where the overlapping areas between adjacent cameras are exploited, thus creating pairs of perfectly matched video streams. However, there still existed some disparity between different pairs. We finally developed an approach where the time between two temporal frames is exploited to communicate the exposures among the cameras where we achieve a perfectly synchronized array. An analysis of the system and some experimental results are presented in this paper. In summary, a pilot-camera approach running in auto-exposure mode and then distributing the used exposure values to the other cameras seems to give best visual results.

Keywords: Automatic exposure, Panoramic video systems, Real-time, Football

1. INTRODUCTION

Capturing panoramic textures has become a popular research area. Using multiple cameras instead of one wide-angle camera provides a higher resolution for large field-of-view textures where such panoramic images or frames again can be used in several different ways. For example, Mavlankar *et al.*[1] used a perspective panorama video as an intermediate representation and crop out regions of interest for an immersive presentation. Carr *et al.*[2] demonstrated a similar system, but using a robotic camera to follow the players in the game aesthetically. More traditional uses include image viewers[3] that can provide detailed presence.

In this respect, several researchers have focused on stitching multiple images together to produce one seamless panorama. However, often they are more focused on aligning the images properly than controlling the capture mechanism. The stitching and alignment tutorial by Szeliski[4] provides an overview of the state-of-the art approaches in stitching. For example, Agarwala *et al.*[5] introduced an interesting approach for stitching panoramic video textures from a single panning video camera and automatic alignment. One later example is given by Brown *et al.*[6] who proposed a fully automatic approach for aligning and stitching panoramic images using SIFT features. All these approaches employ computationally expensive global optimization schemes to estimate and achieve the perfect alignment in any sort of camera movements. But, when the motivation of the system is to record a panoramic video, it is more convenient to have a rigid camera array. In such a configuration, the cameras can be manually calibrated well in advance, eliminating the need for such complicated algorithms. Moreover, when

Contact author: Vamsidhar Reddy Gaddam, E-mail: vamsidhg@ifi.uio.no

a real-time constraint is present, one cannot afford to use the computing resources to align every single frame of the panoramic video.

One of the major things that contribute to the visual quality of a panorama is the color difference between multiple images. Such differences can stem from several factors like inter-camera differences and the amount of light in each image. The ideal case would be to use the same physical camera and the same exposure settings through out the panorama capture. However, when capturing panoramic videos, it is only possible to use the same physical camera if one uses a reflective sphere, but this approach can result in reduced resolution. When an array of multiple cameras produces images that are not captured using similar exposure parameters, there will be visual differences between adjacent camera images. Often this is overcome by color correction approaches, which handle the images post-recording. Xu *et al.*[7] provide a good performance evaluation of several color correction approaches. Xiong *et al.*[8] proposed an elegant color correction approach which applies a color transform that is optimized over all the images to minimize drastic changes per image. Ibrahim *et al.*[9] provide an interesting approach for selecting the reference for color correction.

Nevertheless, even though color correction approaches can provide good results in panorama images, they can introduce artifacts like flicker and unnatural colors when it comes to the stitched videos. This problem can be handled even before the recording of the videos in a constrained space like a sports stadium. In this paper, we therefore propose a novel approach to record panorama videos that do not require any additional color correction step. We developed and implemented several approaches using industrial cameras to record videos of a football stadium. The best results come from an approach where the time between two temporal frames is exploited to communicate the exposure setting from a pilot camera among all the other cameras. An in-depth analysis of our system and the experimental results are presented in this paper.

The rest of the paper is organized as follows: Section 2 provides a brief description of the real-world scenario that we explored and some of the challenges involved. A detailed description of the panorama capture system and various approaches for controlling the exposure are presented in section 3, and some preliminary results are shown in section 4. In section 5, we provide some discussions and conclude the paper in section 6.

2. SCENARIO

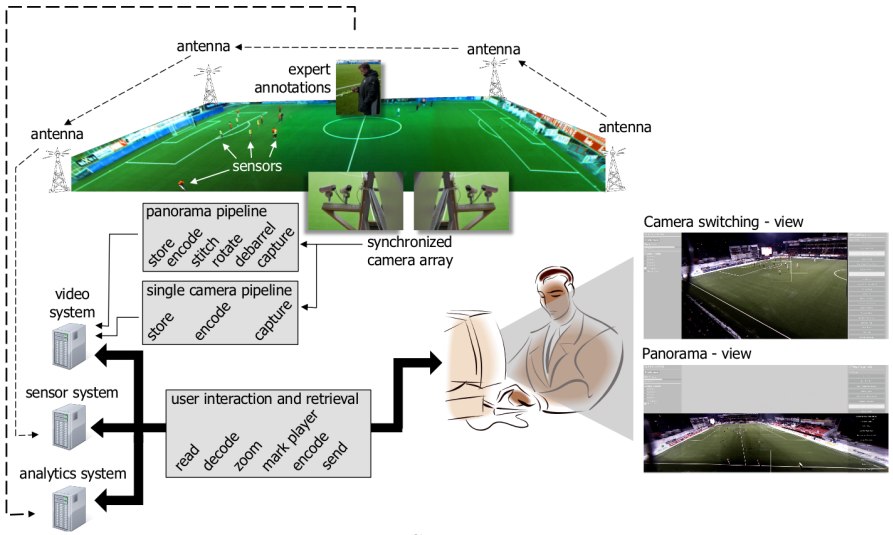
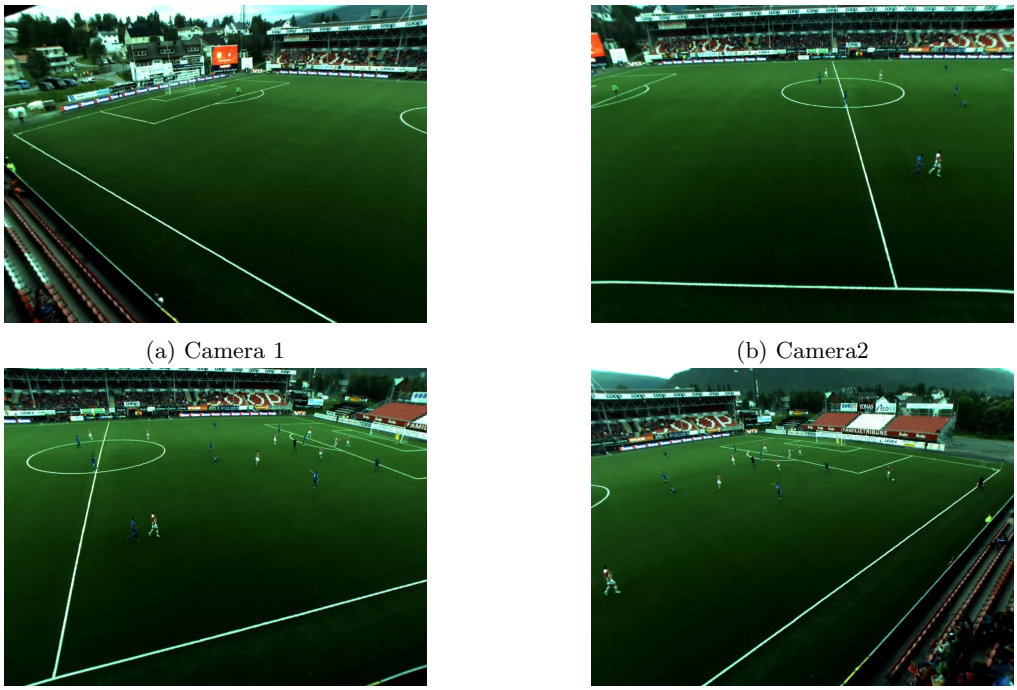


Figure 1: System overview

We have a prototype system[10–12] for capturing panorama videos that is used to display events and videos in an immersive fashion in a football (soccer) analysis scenario. In addition to a sensor subsystem and an event annotation subsystem, we have a panorama video subsystem that consists of four cameras (recently upgraded to five). The whole system is installed at Alfheim stadium in Tromsø, North of Norway. Figure 1 shows an overview of the entire system.

The video subsystem performs two major tasks: Video capture and panorama stitching. The real-time stitching pipeline is described in detail by Tennøe *et al.*[13] The video capture system is responsible for the synchronization of frames, driving the cameras and tranfering of video data from the cameras to the processing system. We use industrial cameras (*acA 1300-30gc*) manufactured by Basler supporting a frame rate upto 30 frames per second and a resolution of 1280×960. The lenses are of 3.5mm and are manufactured by Kowa. Moreover, the cameras are controlled by individual threads which also collect the video data from the cameras, and the time (shutter) synchronization is achieved by building a custom trigger box*. Furthermore, the cameras are placed on one side of the football stadium at an elevation of 10 m from the field. The cameras cover the entire field with sufficient overlap between adjacent cameras, i.e., figure 2 shows the four views captured from the four cameras. It can be observed that there is significant overlap between the cameras that can be used for stitching.



(a) Camera 1 (b) Camera2

(c) Camera 3 (d) Camera 4

Figure 2: Views from each camera for a perfectly synchronized exposure

There are several challenges in designing and implementing such a panorama video system. In the context of the camera setting challenges addressed in this paper, the main problem is that its an outdoor stadium where the light is not controlled. Furthermore, the cameras are looking at the entire field, but there is no place on the entire field that can be used for metering for all the cameras. One general assumption that can be made is that the green grass can be used for metering. However, grass has a directional reflectivity which means that the light seen from different angles can vary quite a bit. The lighting also changes during the game due to frequently changing weather conditions and movement of the sun. Thus, configuring the cameras with the appropriate settings is a big challenge.

In the current scenario using several cameras to generate a panorama video, the aperture must be kept constant to avoid changing depth of field. This means that the only parameters that one can (or should) control are the exposure time and the gain. However,we do not have full freedom in controlling both these parameters. The electric gain introduces noise in the system, so the highest image quality is achieved when the gain is as low as possible. The exposure time has an upper limit both because it can cause motion blur during the game

*Hardware design and software open sourced at `https://bitbucket.org/mpg_code/micro-trigger-box`

and also because there is a hard limit set by the frame rate. So a priority based estimation must be used which changes the exposure time until it reaches the threshold and then modify the gain if the exposure needs more compensation.

Furthermore, due to the difficult lighting conditions, it is not sufficient to set the exposure parameters before the game manually. You cannot select a small patch on the grass for metering because players can pass through it and affect the measurement. To address this challenge, we next present and evaluate three approaches for automatically synchronizing the exposure to the cameras.

3. AUTOMATIC EXPOSURE APPROACHES

A main challenge in a video system generating panorama video from a camera array is to handle an array-wide automatic camera exposure. In this respect, we use the internal metering mechanism to estimate the exposure parameters. The region of interest that is considered for metering can be modified for each camera. We make use of this functionality in the three exposure setting approaches described here.

3.1 Independent metering

This is the most trivial approach for an automatic exposure system. In this approach, we use the fact that the football field provides a nice surface for metering. Since our target space is confined to a football stadium, we can use the green surface of the football field to evaluate the exposure parameters. Initially, a manual selection of metering region is selected per camera, and the cameras are driven to make an automatic exposure. The internal mechanism decides on a specific exposure value and gain to achieve a pre-defined gray value for the average of all the pixels from the metering region. An upper limit can be imposed on the exposure time to force the camera to use a higher gain in case of low light, instead of increasing the exposure time.

3.2 Pairs metering

This approach can be considered as a special case of the Independent metering presented above. In this approach, we exploit the fact that the adjacent cameras have an overlapping region. Therefore, camera pairs are formed which have defined regions of interest that points to the same physical space on the field. The selection of the region of interests are performed manually to minimize the effect from the players or other elements on the field. Then the cameras are run independently to perform automatic exposure but metering based on the selected patches that are overlapped. Since the camera pairs are physically close to each other, the directional reflections will have minimum effect on the exposure. But the first camera pair and the second pair are at a distance of 4m from each other.

3.3 Pilot camera approach

In this approach, there is a pilot camera which functions in auto-exposure mode where the pilot camera's exposure parameters are transfered to the other cameras. Here, let the m cameras be named C_j where $j \in [1, m]$, and C_p be the pilot camera. Let e_j and g_j be the exposure time and gain of camera C_j.

Then, given e_p and g_p from the pilot camera which operates in auto exposure mode, we need to compute e_j and g_j for the rest of the cameras. Furthermore, let T_j be the transformation function from the pilot camera to camera C_j. Then,

$$(e_j, g_j) = T_j(e_p, g_p). \tag{1}$$

The transformation function depends on the relation of camera C_j to the camera C_p. In an ideal situation where the cameras are all identical and have exactly the same settings for aperture and focal length, T_j will be identity function. However, this is not the general case because physically different cameras do not have identical spectral response curves thus leading to difference in exposures. Other factors that can cause differences are the imperfections in adjustment of the aperture size. Generally, the cameras need a prior calibration step to estimate the corresponding transformation functions.

The general processing flow is presented in figure 3. There are two types of threads that are running concurrently, i.e., one is for controlling and communicating with the pilot camera, and the other type is for the rest of

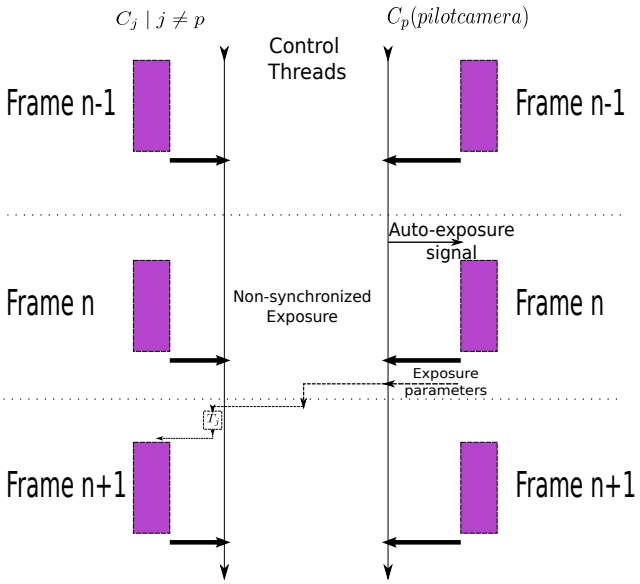


Figure 3: Pilot camera approach

the cameras. All these threads have a synchronization barrier at the end of every frame. Periodically, the pilot camera thread sends a trigger to the pilot camera to make an auto-exposure signal and lock the exposure until the next trigger. In figure 3, this can be seen before acquisition of frame n. After the exposure, the exposure parameters e_p and g_p are transferred back to the controlling machine. These parameters are communicated to other threads which in turn transfer these individually to the other cameras applying the appropriate transformation.

It can be observed that the frames n of the other cameras are not synchronized in exposure with the pilot camera, but we have observed empirically that the light conditions change slowly over the period of the exposure updating trigger. One more important detail is that the frame rate sets a hard upper bound on the execution time and thus on exposure time too. The formulation of transformation function cannot guarantee this because one of the transformations can demand a higher exposure time than the upper limit. Especially, when the cameras have lower response to light than the pilot camera. This problem can be handled in two ways. One way is to embed this property into the transformation function by placing an upper bound. The other way is to handle it in the driver before setting the camera parameters. We found that the driver solution is safer and more robust to further changes in the algorithm.

4. EXPERIMENTS

We will show stitched panorama to demonstrate the visual effect easily. The panorama images presented here for each approach are to emphasize the different lighting conditions. The first sections present images recorded during different lighting conditions that emphasize the differences in the approaches. Section4.4 shows the result using the three approaches from the same match in the same lighting condition for a fair comparison.

4.1 Independent metering

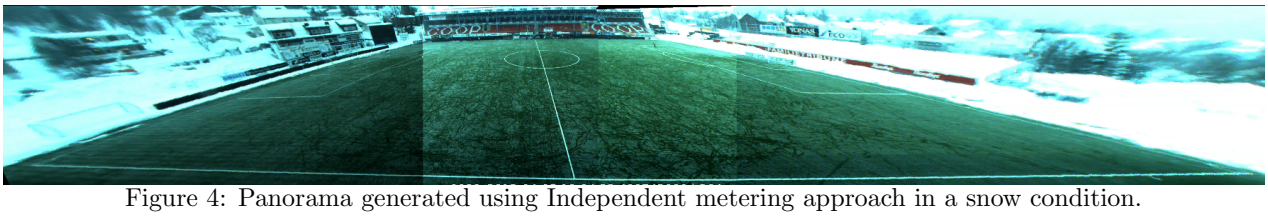


Figure 4: Panorama generated using Independent metering approach in a snow condition.

Figure 4 shows one of the light conditions, where there is snow around the football field. The metering system has to compensate for this and make good choice of exposure values. The influence of snow can be observed in the independent metering approach. But the problem is that the exposures are different in each of the cameras, even though each of the images are well exposed, they are not synchronized.

4.2 Pairs metering

Figure 5: Panorama generated using Pairs metering approach under a partially cloudy sky.

In this approach a clear difference can be seen at the center of the field. But the left two and the right two parts of the panorama are perfectly seamless. Figure 5 shows one of the possible light conditions. It is when the sky is partially cloudy.

4.3 Pilot camera approach

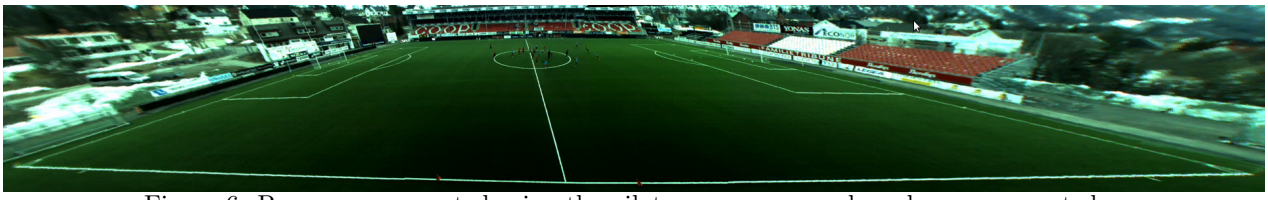


Figure 6: Panorama generated using the pilot camera approach under an overcast sky.

Figure 6 shows another lighting condition where there is an overcast sky. But it can be observed from the figure that the exposure in the whole of the panorama is perfectly synchronized. There is no specific color-correction applied when stitching the panorama.

4.4 Comparison

In this section we present frames using the three approaches during similar time period for comparison. This is also one of the hardest light conditions to handle, when there is direct sun on the stadium and the stands cast a shadow. In such a case, the camera's dynamic range is insufficient to capture variation in the light and dark areas. It can be observed that the first and second approach provide rather similar result where as the third approach provides a seamless result. This similarity between the first two approaches has been observed in different light conditions as well.

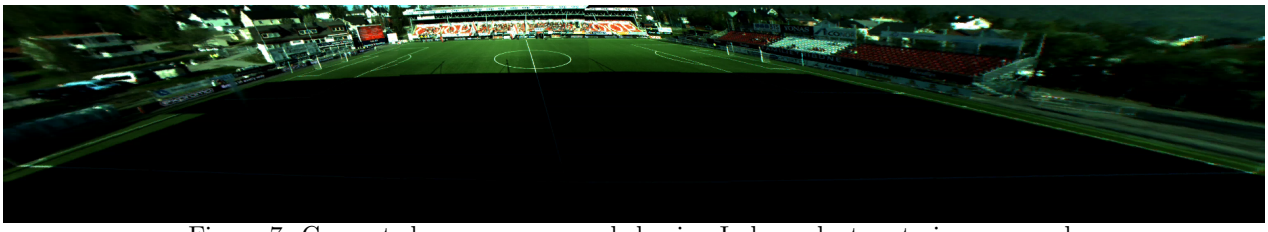


Figure 7: Generated panorama recorded using Independent metering approach

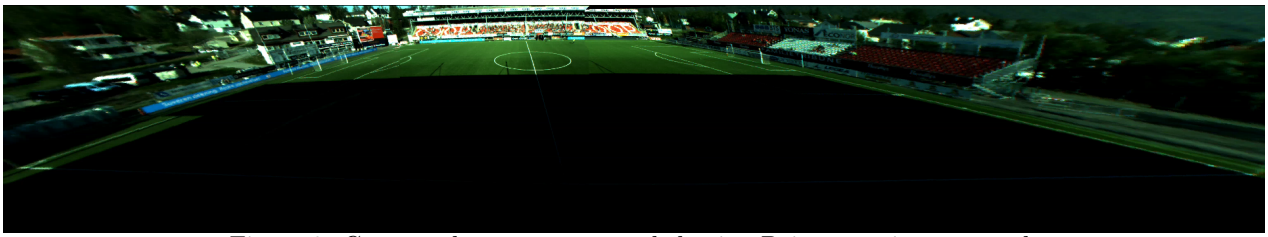


Figure 8: Generated panorama recorded using Pairs metering approach

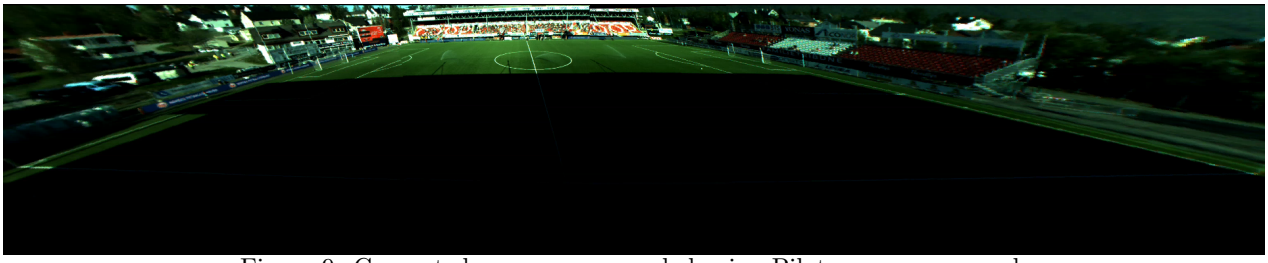


Figure 9: Generated panorama recorded using Pilot camera approach

5. DISCUSSION

The approaches perform varyingly depending on the lighting conditions. In several situations the first two approaches provide similar results owing to the large overlap between adjacent pairs. The third approach always provides seamless panoramic video. But all three approaches fail to provide a visually pleasing output in the case of high contrast. This is more of a lack in the dynamic range of the cameras than the shortcoming of the approach itself.

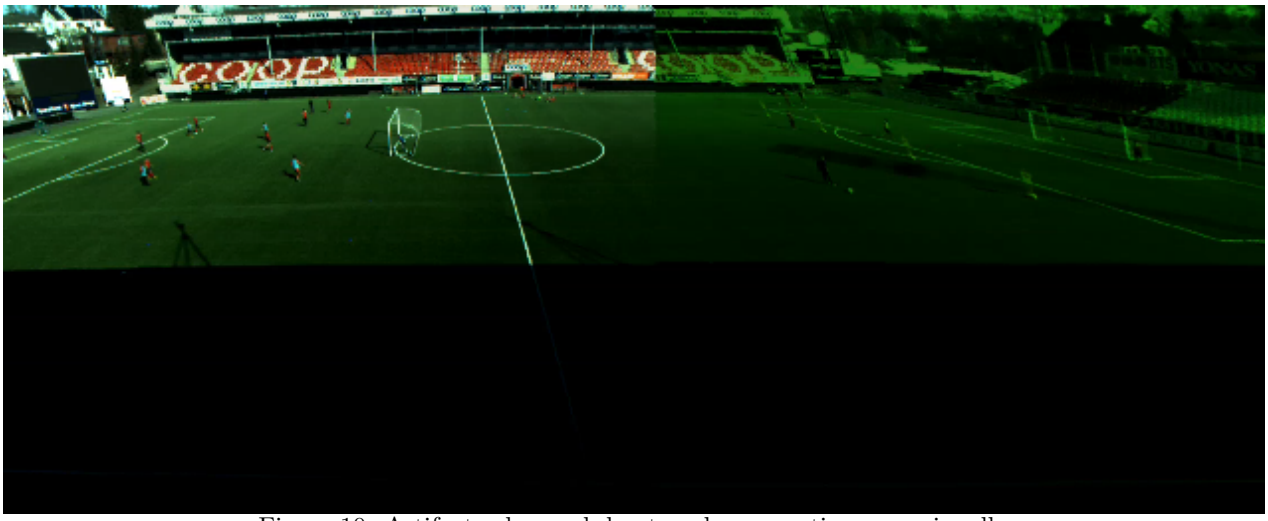


Figure 10: Artifacts observed due to color correction occassionally

We initially employed a color correction component in our stitching pipeline. As mentioned earlier, applying color correction to every frame that is estimated according to the current frames introduces unwanted artifacts. Figure 10 shows one such effect. Here it can be seen that the color correction is trying to achieve a seamless field but instead introduces an unnatural green in the whole of the right part of the panorama. This can be avoided using only correction in the luminance component, but this cannot provide a seamless scene in the target areas[on the field].

The system can be easily extended to accommodate other interesting functionalities like capturing of high dynamic range videos by making multiple exposures. The pilot camera approach can be configured such that two

kinds of exposures are performed every alternating frame. One exposure to capture the details in the highlights region and the other to capture the details in shadow region. Then the video can be formed by fusing these two exposures appropriately.

6. CONCLUSION

In this paper, we present an elegant approach for controlling the exposure of a multi-camera array such that we get a uniform exposure across the captured panorama. This approach can be scaled to several cameras. We also presented results from the experiments performed from a real setup in a football stadium. An outdoor stadium provided us with plenty of challenges, where the light is essentially uncontrolled. We also discuss the applications of panorama to show the motivation for an automated exposure control system.

Later, we provided some examples on how this system can be extended to handle more challenging conditions. The limited dynamic range of the camera need not effect the overall dynamic range of the captured panorama. There are several methods that exist for doing this for an individual camera or one static panoramic image. We provide a way to extend it to panoramic videos.

ACKNOWLEDGMENTS

This work has been performed in the context of the *iAD* center for Research-based Innovation (project number 174867) funded by the Norwegian Research Council.

REFERENCES

[1] Mavlankar, A. and Girod, B., "Video streaming with interactive pan/tilt/zoom," in [*High-Quality Visual Experience*], Mrak, M., Grgic, M., and Kunt, M., eds., *Signals and Communication Technology*, 431–455, Springer Berlin Heidelberg (2010).

[2] Carr, P., Mistry, M., and Matthews, I., "Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording," in [*Proc. of ACM MM*], 193–202 (2013).

[3] Kopf, J., Uyttendaele, M., Deussen, O., and Cohen, M. F., "Capturing and viewing gigapixel images," *ACM Trans. Graph.* **26** (July 2007).

[4] Szeliski, R., "Image alignment and stitching: A tutorial," *Found. Trends. Comput. Graph. Vis.* **2**, 1–104 (Jan. 2006).

[5] Agarwala, A., Zheng, K. C., Pal, C., Agrawala, M., Cohen, M., Curless, B., Salesin, D., and Szeliski, R., "Panoramic video textures," *ACM Trans. Graph.* **24**, 821–827 (July 2005).

[6] Brown, M. and Lowe, D. G., "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision* **74**, 59–73 (Aug. 2007).

[7] Xu, W. and Mulligan, J., "Performance evaluation of color correction approaches for automatic multi-view image and video stitching," in [*Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*], 263–270 (2010).

[8] Xiong, Y. and Pulli, K., "Color correction for mobile panorama imaging," in [*Proc. of ICIMCS*], 219–226 (2009).

[9] Ibrahim, M., Hafiz, R., Khan, M., Cho, Y., and Cha, J., "Automatic reference selection for parametric color correction schemes for panoramic video stitching," in [*Advances in Visual Computing*], *Lecture Notes in Computer Science* **7431**, 492–501, Springer Berlin Heidelberg (2012).

[10] Sægrov, S., Eichhorn, A., Emerslund, J., Stensland, H. K., Griwodz, C., Johansen, D., and Halvorsen, P., "Bagadus: An integrated system for soccer analysis (demo)," in [*Proc. of ICDSC*], (Oct. 2012).

[11] Halvorsen, P., Sægrov, S., Mortensen, A., Kristensen, D. K., Eichhorn, A., Stenhaug, M., Dahl, S., Stensland, H. K., Gaddam, V. R., Griwodz, C., and Johansen, D., "Bagadus: An integrated system for arena sports analytics – a soccer case study," in [*Proc. of ACM MMSys*], 48–59 (Mar. 2013).

[12] Stensland, H. K., Gaddam, V. R., Tennøe, M., Helgedagsrud, E., Næss, M., Alstad, H. K., Mortensen, A., Langseth, R., Ljødal, S., Landsverk, Ø., Griwodz, C., Halvorsen, P., Stenhaug, M., and Johansen, D., "Bagadus: An integrated real-time system for soccer analytics," *ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP)* (2014).

[13] Tennøe, M., Helgedagsrud, E., Næss, M., Alstad, H. K., Stensland, H. K., Gaddam, V. R., Johansen, D., Griwodz, C., and Halvorsen, P., "Efficient implementation and processing of a real-time panorama video pipeline," in [*Proc. of IEEE ISM*], (Dec. 2013).

Appendix G

[Conference] Interactive Zoom and Panning from Live Panoramic Video

[Authors:] **V. R. Gaddam**, R. Langseth, S. Ljødal, P. Gurdjos, V. Charvillat, C. Griwodz, and P. Halvorsen.
[Published:] ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 2014.

Interactive Zoom and Panning from Live Panoramic Video

Vamsidhar Reddy Gaddam[1], Ragnar Langseth[1], Sigurd Ljødal[1], Pierre Gurdjos[2], Vincent Charvillat[2], Carsten Griwodz[1], Pål Halvorsen[1]

[1]Simula Research Laboratory & University of Oslo, Norway
[2]Universite de Toulouse, France

[1]{vamsidhg, ragnarla, sigurdlj, griff, paalh}@ifi.uio.no
[2]{pgurdjos, vincent.charvillat}@enseeiht.fr

ABSTRACT

Panorama video is becoming increasingly popular, and we present an end-to-end real-time system to interactively zoom and pan into high-resolution panoramic videos. Compared to existing systems using perspective panoramas with cropping, our approach creates a cylindrical panorama. Here, the perspective is corrected in real-time, and the result is a better and more natural zoom. Our experimental results also indicate that such zoomed virtual views can be generated far below the frame-rate threshold. Taking into account recent trends in device development, our approach should be able to scale to a large number of concurrent users in the near future.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: [Distributed applications]

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

panorama video, zoom, panning, real-time

1. INTRODUCTION

The growing availability of high-speed Internet access has gone along with a growth in interactive and immersive multimedia applications, and panoramic video is a feature that is becoming more and more popular in various scenarios for its ability to increase immersion. We look at it in the context of arena sports like soccer, which have always provided opportunities for innovation in broadcasting. The challenges in providing interactive experiences in broadcasting to a large audience span several fields. One such challenge is to provide a real-time pannable and zoomable virtual camera to several thousands or even millions of users.

NOSSDAV'14, March 19 – 21 2014, Singapore, Singapore

http://dx.doi.org/10.1145/2578260.2578264

This type of challenges has attracted the attention of several researchers [10, 12, 3, 5, 8, 4, 13]. For example, Carr *et al.* [5] demonstrated recently a hybrid robotic/virtual camera for event recording. It uses a robotic camera to follow features and assist a virtual camera. Such a system can provide a smooth and aesthetic camera, but it is restricted to *one* virtual camera per a robotic camera. So, in case of a multi-user system, all clients receive the same video. Furthermore, Mavlankar *et al.* [8] describe a pan-tilt-zoom streaming system, but the system presented merely crops from a larger video. For a large space like a soccer stadium, such a system introduces widely varying quality from one end to the other end. If the perspective nature is maintained (straight lines remain straight) in the panorama, a large amount of redundant data is transferred on the close end.

To provide a better perceived quality when zooming and panning into a panorama video and keep redundant data transfer low, we present a system that uses a cylindrical panorama as an intermediate representation. The approach followed by us to create the intermediate representation is one of the numerous choices available. Then, a virtual view is generated where the perspective of the delivered video is corrected before it is presented to the client. The client has full freedom to pan, tilt and zoom using the system, and the system supports several algorithms for pixel interpolation. Furthermore, in order to support a large number of concurrent users, we aim for a lightweight system to generate the views in real-time. Therefore, we have implemented several versions running on both CPUs and GPUs, and we discuss the system's ability to scale to a large number of users – both to spectators within a stadium as well as users outside.

Furthermore, we generate a virtual view frame in about 10 ms on a GPU, and our system supports real-time interactions. If processed on the server side, we support a limited number of users per machine. However, taking into account existing work to transfer panorama video in real-time [6] and the fact that GPUs are becoming a commodity also on mobile phones, we work on an adaptation of our system that creates the virtual camera on the client side. This system will trade bandwidth for server processing power by broadcasting the panorama video to all clients instead of computing and transmitting a personalized view for each client.

The rest of our paper is organized as follows: Section 2 briefly describes the example soccer scenario with our real-world installation. In section 3, we describe our system, and we present our experiments and evaluation results in

section 4. Section 5 provides some discussions before we conclude the paper in section 6.

2. EXAMPLE SCENARIO

Our prototype is currently installed at Alfheim, a soccer stadium in Tromsø, Norway. In this scenario, we use the panorama video in a sport analysis system [7, 9] where video events must be generated in real-time. The camera array is integrated with a player-position sensor system and an expert annotation system. The integrated system enables users to search for events, follow one or more players in the video, and automatically query for video summaries.

Figure 1: Generated panorama video.

The scenario poses an array of challenges. To stitch the individual cameras into a panorama video (see figure 1), the camera shutters must be synchronized. This was accomplished by building a custom trigger box[1]. The cameras must again be synchronized with the sensor system to correctly identify the corresponding video frames and sensor records [7]. Furthermore, the high-rate video streams must be transferred and processed in real-time [11].

Even though the system was originally meant for the coaches, a lot of this functionality is also interesting for outside users. Our goal is to enable spectators in the stadium and supporters at home to use a part of the system with individual videos. This means that every user must be able to interact with the system in real-time to generate their own personal view from a single viewpoint.

Until recently, our virtual views were limited to cropping and scaling. For a better user experience, we need perspective correction, and to scale the system up to thousands of spectators in a stadium, we need a scalable way of generating and delivering the individual views.

3. SYSTEM OVERVIEW

Our system is divided into a panorama generation part and a video delivery part, which supports user-controlled interactive virtual cameras. A sketch of our system is given in figure 2. In this section, we describe these two parts.

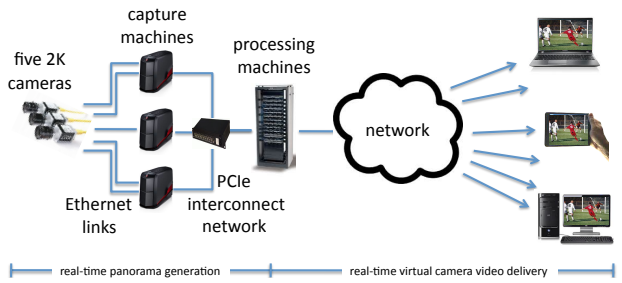


Figure 2: Setup.

[1]Hardware design and software open sourced at `https://bitbucket.org/mpg_code/micro-trigger-box`

3.1 Real-time Panorama Video Capture

As we have earlier described the real-time version of the panorama video pipeline [11], we only highlight the changes made to increase the resolution (better and more cameras) and to distribute the capture and processing.

The capture pipeline (left-most part of figure 2) consists of several sub-components. First, we have a *camera reader* that fetches raw video frames from the (frame-synchronized) cameras over a point-to-point Ethernet network. Depending on the chosen output format from the cameras, we have a *debayering* component interpolating the frames from a Bayer pattern to full YUV 4:2:2 used later in the pipeline. To move data to the panorama processing machine, we use a *Dolphin* high-speed interconnect component transferring the uncompressed video data from the recording machines to the single processing machine. On the processing machine, we first have a *frame synchronizer*, which retrieves a full set of synchronized frames from the separate camera streams. It delivers each set of frames to the *panorama stitcher*, which generates cylindrical panorama frames. Finally, we have a *video encoder*, which encodes the panorama video in H.264 for immediate delivery to clients and for storage for on-demand operations.

3.1.1 Video Capture

To capture the entire soccer field, we use five Basler industry vision cameras [2], each of which delivers a maximum resolution of 2046×1086 pixels at 50 frames per second over Gigabit Ethernet. We use an 8mm lens [1] with virtually no image distortion, allowing us to bypass the lossy debarreling step of our previous pipeline [11]. To maximize the panorama resolution, the cameras are rotated by 90° (see figure 3), giving a vertical field-of-view (fov) of 66°. Furthermore, the cameras are mounted in a circular pattern, i.e., pitched, yawed and rolled to look directly through a point 5cm in front of the lenses, in an attempt to reduce parallax effects. As a result of this, only minor adjustments are required before the images can be stitched together. The video capture system also determines the required exposure, which requires frequent changes due to quickly changing light conditions outdoors. Auto-exposure is performed on the center camera once every 4 seconds , and the camera reader module broadcasts the resulting exposure parameters to the other camera readers.

Figure 3: Mounted camera array

3.1.2 Distributed Processing

The new setup requires many more resources (particularly bandwidth) per camera, and we therefore had to distribute capturing as shown in figure 2. To transfer the video streams to the panorama processing machine, a single gigabit Ethernet network is insufficient. We therefore use Dolphin Interconnect cards, which use PCIe as the interconnect protocol and allow for ultra-low latency Direct Memory Access (DMA) across several machines. We transfer each frame

with a single DMA transfer at a rate of 19.41 Gbps and a total latency of 1.26 ms per frame.

3.1.3 Cylindrical Projections

Our new pipeline generates a cylindrical stitched panorama as shown in figure 1. The idea is to have the cameras in the center of a virtual cylinder where each source image can be considered a (cropped) plane that is tangential to the cylinder and orthogonal to its camera's viewing axis. Each pixel of the cylinder is then computed as the (interpolated) pixel value of the ray from the camera center through the pixel intersecting the image plane.

The radius (r) of the cylinder is determined by the width (W_s) of the source images and its field of view (fov):

$$r = \frac{W_s}{2 * tan(\frac{fov}{2})} \tag{1}$$

The viewing axes of the cameras form a plane L (orthogonal to the rotation axis of the cylinder). The angle between the viewing axes of two neighbouring cameras is $\approx 28.3°$, with the 0°angle assigned to the center camera. For brevity, we say also that the angle α of a camera is the angle α of its corresponding source image. The unrolled cylinder forms a Cartesian coordinate system, where $(0, 0)$ corresponds to the intersection of the center camera's viewing axis with the cylinder and the X axis corresponds to the intersection of L and the cylinder.

Every pixel coordinate (T_x, T_y) on the unrolled cylinder determines the corresponding horizontal (θ) and vertical (ϕ) angles of a ray from the camera center through this coordinate.

$$\theta = \frac{T_x}{r} \quad \text{and} \quad \phi = arctan\left(\frac{T_y}{r}\right) \tag{2}$$

To determine the pixel(s) in the source image for every pixel (T_x, T_y), the source image with the closest α value to θ is selected and α is subtracted, essentially centering the coordinate system on that camera's viewing axis. Then, the point x', y', z' in 3D space where the ray intersects the image plane is determined by:

$$z' = r \quad \text{and} \quad x' = tan(\theta) * z' \tag{3}$$

$$y' = tan(\phi) * \sqrt{z'^2 + x'^2} \tag{4}$$

This relationship is visualized in figure 4.

This algorithm requires each image to be perfectly aligned and rotated. The camera mount seen in figure 3 provides a good alignment, but there are small deviations that must be corrected. A small per-camera vertical correction is easily applied with no additional complexity. For each source image, a standard rotational matrix is created to compensate for rotation around the camera's x, y and z axis.

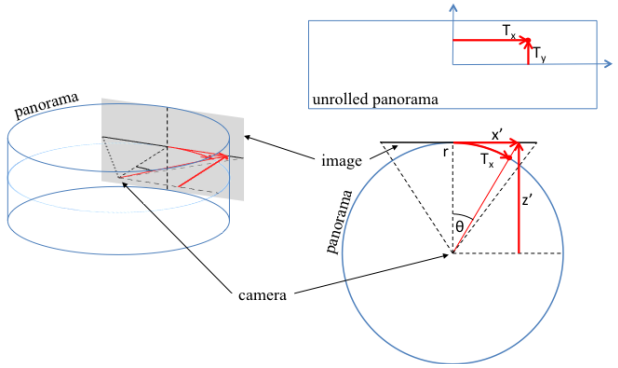


Figure 4: Creating panorama pixels from a captured image

The Cartesian coordinates of the pixels on the cylinder are multiplied with this rotational matrix, rotating the cylinder before we project onto it. Then we can find θ and ϕ based on the new coordinates:

$$\theta = arctan\left(\frac{x}{z}\right) \quad \text{and} \quad \phi = arctan\left(\frac{y * sin(\theta)}{x}\right) \tag{5}$$

The computational complexity of equations in 5 is significantly greater than that of eq. 2.

3.2 Live Panoramic Zoom and Panning

Using the cylindrical panorama as an intermediate representation, our system can generate an arbitrary virtual camera view from the position of the camera array. The user has the full freedom to pan, tilt and zoom. As shown in figure 5, the virtual camera view is corrected to a perspective view very similar to that of a physical camera.

Figure 5: Panorama video with labeled ROI (left) and the virtual camera generated (right). It can be observed that it is not a simple crop from the bigger video.

This enables the virtual camera video delivery part shown on the right in figure 2. It consists of fetching and decoding the panoramic video file, creating the virtual camera and handling user input. The implementation uses one thread for most of the process including decoding the current frame, moving the data from host to device, creation of the virtual camera and rendering the texture onto a display. Fetching of video segment data and user input are handled by different threads.

3.2.1 Video Handling

For streaming, we use HTTP segment streaming (with plans for adaptive HTTP streaming). The segments of the panoramic videos are served by an Apache server along with a manifest file. The manifest file is used to inform the clients when the next file is ready for download. The viewer checks the manifest file periodically and downloads the next segment when ready.

As soon as the panoramic video segment is transferred, it is kept ready for processing. This process runs in the background without blocking either the display thread or the user input thread. At the moment, the system demands a large network bandwidth due to the full resolution panorama video. Nevertheless the panoramic video is compressed using H.264, saving quite a lot of space.

3.2.2 Virtual Camera

Panning and zooming are performed by a virtual perspective camera. The cylindrical panoramic texture generated as

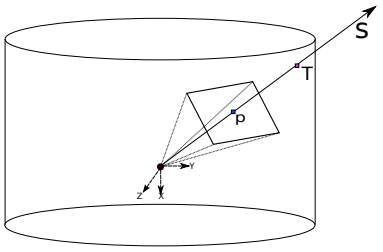


Figure 6: The intersection of the ray from the virtual view with the unit cylinder

described in section 3.1.3 is viewed through a virtual camera and the output is rendered on the screen.

When we use the virtual camera, the operation is to fetch the pixels of the image formed on the camera from the cylindrical texture. This can be better seen in figure 6.

A pin-hole camera for a point projection from a 3D point P to image point q can be written in the following manner:

$$\lambda q = [K|0_3] \begin{bmatrix} R & 0 \\ 0_3 & 1 \end{bmatrix} \begin{bmatrix} 0_3^T & -C \\ 0 & 1 \end{bmatrix} P \tag{6}$$

where R is the general (3×3) 3D rotation matrix as a function of θ_x, θ_y and θ_z, the rotation angles around the x, y and z axes respectively and K is the camera intrinsic matrix built with focal length(f).

Let p be the current pixel. So we need to find the ray that passes from the camera center C to the pixel p. The ray can be represented by:

$$s = \lambda R^{-1} K^{-1} p \tag{7}$$

Then the intersection of this ray with the unit cylinder gives us the exact position on the cylindrical texture. The intersection point can be found as follows:

$$T_x = \left(\frac{W_p}{FOV}\right) \left\{ arctan\left(\frac{-s(1)}{s(3)}\right) \right\} + \frac{W_p}{2} \tag{8}$$

$$T_y = \left(\frac{1}{2} - \frac{s(2)}{\sqrt{s(1)^2 + s(3)^2}}\right) H_p \tag{9}$$

where W_p, H_p and FOV are the width, height and the field of view of the panoramic texture respectively. (T_x, T_y) are the coordinates on the unrolled cylindrical texture as described in section 3.1.3. When these calculations are performed with sub-pixel accuracy, it is not necessary that the intersection always lands at one pixel. So there is a need for interpolation from the surrounding pixels. A few experimental results are presented in the next section in this respect.

3.2.3 Implementations

A straightforward implementation (CPU) of such a viewer is to loop through all the pixels in the virtual view and find the positions where the rays land on the panoramic texture. The heavy operations include an inverse tangent and a square root in every pixel calculation. Since the operations are well suited for parallelization, we have ported the program to a GPU.

A simple port (GPU1) using CUDA performs the calculation of the ray intersection and the fetching of the corresponding pixel on the GPU. So, in the initial implementation the videos are decoded on the CPU, the frames are transferred to the GPU, and calculations and fetching operations are performed on the GPU. Then the virtual image is created and transferred to the host for displaying/storing purposes.

Since it is possible to render OpenGL textures written by an NVidia CUDA kernel directly from the GPU to the screen, the current implementation (GPU2) uses that feature. A texture area is defined in advance and bound to the screen buffer. When the fetching operations are complete, the output is not transferred to the host, but written to the bound texture buffer on the GPU. Then this texture is displayed directly on the screen, saving the transfer overhead from device to the host. Other optimizations of GPU2 include the use of CUDA texture buffers instead of global memory on the GPU for the panorama frames to speed up the fetching operations owing to hardware-accelerated interpolation.

3.2.4 Operation

The user can pan, tilt or zoom using the virtual camera. When the panning operation is performed, θ_x is modified. θ_y is modified when a tilting operation is performed on the camera. The zoom is realized by modifying the focal length f of the virtual camera.

4. EXPERIMENTS

To test our prototype, we have performed a number of experiments during the development of the system. In this section, we present the results.

4.1 Real-time generation of panorama video

We have earlier shown that we are able to generate panorama video in real-time [11], and we therefore do not focus our evaluation on this part of the system. However, we have replaced the four 1K-cameras with five 2K-cameras, i.e., generating more than the double number of pixels. Nevertheless, using the setup in figure 2 distributing the capturing and processing, we are still able to process the panorama frames in real-time, i.e., generating a 25 fps video live. In average, we are able to generate the frames in 34.58 ms on an Intel i7 CPU, and we are currently working towards delivering a higher frame rate. In this respect, the only bottleneck is the processing. To deliver higher rates over the 1 Gbps Ethernet link, the cameras must use a Bayer representation which must be converted to YUV on the host, and at higher frame rates, both the stitcher and x264 encoder may become bottlenecks. We are therefore currently moving the new pipeline back to the GPU we had in [11], but for our current 25 fps prototype, the current setup is sufficient for real-time panorama generation.

4.2 Zooming and panning

After the panorama video has been generated, it is ready for consumption by local or remote clients. The video must be retrieved, processed for zooming and panning and processed for scaling (interpolation).

4.2.1 Video Download

First, the video must be fetched to the processing machine of the client. The time taken for downloading is measured and the results for a few video segments are presented in figure 7. Each of the segments has a playout duration of

3 seconds and varying file size. The experiments are performed using three different networks: 1) All on the same machine using localhost; 2) Client and server on the same local network; and 3) Client is 1.500 km avay from the server over a wide-area network. Figure 7 shows the average times taken for 23 files and their sizes in the background to illustrate the effect of the sizes. The plot also demonstrates a horizontal cut-off line which is the limit for smooth playout.

Usually, we run the system over the norwegian research backbone network (Uninett with 19 hops). We have, however, also tested it locally. It can be observed that even when the client and server are separated by a real network, which is subject to unknown traffic shaping and congestion on the Tromsø side, the client is still able to perform close to the smooth playout. Of course, this varies depending on the bandwidth available to the client. Since the client side processing is pipelined, only one segment buffer is required to make this run in real-time – if the segments can be downloaded in real-time. By applying adaptive HTTP streaming (we expect that this is feasible when the GPU port of the new server pipeline is complete), we will be able to overcome this bottleneck at the price of reduced quality.

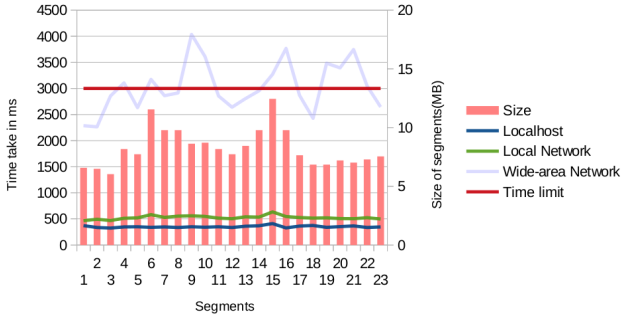


Figure 7: A plot showing various times taken

4.2.2 *Comparison of implementations*

In section 3.2.3, we described three different implementations of the computation and rendering of the virtual camera. The average execution time for each of these approaches is presented in table 1. The experiments were performed on a machine with an Intel i7-2600 CPU and an Nvidia GeForce GTX 460 GPU. The time is observed for the same video segment in all three cases and for a virtual camera with Full HD resolution (1920×1080). The time also includes the decoding time of the panorama frame and all transfers between host and GPU.

Approach	Average	variance
CPU	255.7	35.8
GPU1	13.3	2.6
GPU2	10.1	3.2

Table 1: Execution time per frame (ms).

The unoptimized (no multi-threading or SIMD instructions) CPU implementation is not fast enough to provide the full frame rate of 25 fps. A straight-forward GPU implementation reduces the computation time drastically.

4.2.3 *Interpolation comparison*

Since the panorama is rendered on a cylinder, pixel interpolation is required for every virtual camera independent of zoom or pan setting. Interpolation trades computing time for smoothness and sharpness, and we tested three approaches: nearest neighbour, bilinear and bicubic. Figure 8 presents a highly zoomed frame in the three different modes with the kernel execution time for each of these approaches.

As it can be seen in the figure, bicubic interpolation seems to provide the best visual quality at the cost of a higher execution time. However, we pay less than 3.5ms and choose the higher image quality. Furthermore, CUDA supports an optimized linear interpolation on some hardware, which could be used as an alternative.

(a) Nearest neighbour (2916 us). (b) Bilinear (2840 us). (c) Bicubic (3242 us).

Figure 8: Execution time for the interpolation algorithms

4.2.4 *Size of virtual view*

Most of the performance measures provided in this section are for a Full HD resolution, but the resolution of the virtual camera varies with the viewing device. Figure 9 therefore demonstrates the effect of the size on the kernel execution time of the final generation of the zoomed image (note the *micro*second scale). As can be seen in the plot, the resolution has negligible impact on performance (as the whole panorama video has to be moved to the GPU anyway).

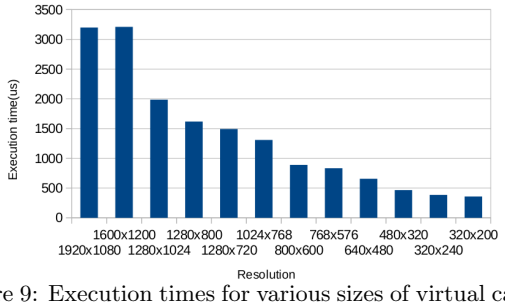


Figure 9: Execution times for various sizes of virtual camera

5. DISCUSSIONS

5.1 Panoramic stitching

In the current system, panoramic stitching is performed using a static lookup table. Each pixel in the output panorama is mapped to a pixel in one of the source images. At runtime, the pixels are merely interpolated, both from the cylindrical warp and the down-sampled chroma channels in the YUV 4:2:2 pixel format.

We intend to move this module to GPU, and re-introduce dynamic seam-finding [11]. The dynamic seam can be determined at runtime, and pixel values dynamically chosen from one of two lookup tables in the overlapping region of the source images.

5.2 Scalability

Even though the generation of each frame is fast, the scalability of the system depends on where the video for the virtual camera is generated. By running everything on the sender side, we have the advantage of supporting light client devices and reducing the bandwidth required of the client's access network. This option, however, requires large amounts of processing power, as well as network bandwidth on the server side, i.e., the scalability is limited on the server side. On the other hand, by transferring the panorama video to the client device, our servers can easily support a large number of clients at the cost of client processing and a distribution infrastructure that can deliver the panorama stream (broadcast, multicast, caches, proxies, P2P, ...). Since the current trend is to equip even mobile phones with powerful CPUs and GPUs, we consider the second option most promising. The benefit of this is particularly visible in our chosen scenario of delivering personalized panning and zooming to the spectators in the stadium, where broadcast support is feasible, while the limited wireless spectrum prevents the delivery of individual streams.

5.3 Ongoing work

Such a system, even though it provides an interactive way to control the virtual camera, is not practical if the user is really meant to perform pan and zoom operations manually during the entire game. So, we are currently working on providing a virtual camera based on a request from the user. For example, the user will have the ability to select the main feature of interest, such as the ball, a specific player or a group of players. The challenge then is to create an aesthetic and smooth virtual camera on the client based on meta-information such as player and ball positions. Naturally, this must be done on the client in real-time to maintain our scalability goals.

We are currently experimenting with different approaches for generating a virtual view that follows a feature point aesthetically and smoothly. Our preliminary prototype seems to follow the ball well[2] (note that finding the position of the ball is out of the scope of this paper).

6. CONCLUSION

In this paper, we have presented a system for real-time interactive zoom and panning of panorama video used in a soccer stadium scenario. Based on video streams from five stationary 2K cameras, processed and stitched into a high-resolution panorama video, we are able to support any free view angle from the position of the camera array, i.e., a virtual camera. The unoptimized running prototype is able to generate one frame of the virtual camera from the panorama image in less than 10 ms on a commodity computer with a standard GPU. Thus, it can easily be scaled to support many concurrent users. If the processing is performed on the client device, at the cost of transferring the panorama video, any number of virtual cameras can be supported, i.e., all users may have their own virtual view.

There are a number of open challenges that are currently under investigation. We aim for more efficient implementations to reduce the resource consumption further, and we investigate more efficient algorithms for following a player or the ball smoothly.

[2]See http://www.youtube.com/watch?v=554RjEEtw3o

Acknowledgments

This work has been performed in the context of the *iAD* center for Research-based Innovation (project number 174867) funded by the Norwegian Research Council. Furthermore, the authors also acknowledge the support and hardware given by Dolphin Interconnect Solutions.

7. REFERENCES

[1] Azure-0814m5m.

[2] Basler aca2000-50gc.

[3] Y. Ariki, S. Kubota, and M. Kumano. Automatic production system of soccer sports video by digital camera work based on situation recognition. In *Proc. of ISM*, pages 851–860, 2006.

[4] P. Carr and R. Hartley. Portable multi-megapixel camera with real-time recording and playback. In *Proc. of DICTA*, pages 74–80, 2009.

[5] P. Carr, M. Mistry, and I. Matthews. Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording. In *Proc. of ACM MM*, pages 193–202, 2013.

[6] R. Guntur and W. T. Ooi. On tile assignment for region-of-interest video streaming in a wireless lan. In *Proc. of NOSSDAV*, pages 59–64, 2012.

[7] P. Halvorsen, S. Sægrov, A. Mortensen, D. K. Kristensen, A. Eichhorn, M. Stenhaug, S. Dahl, H. K. Stensland, V. R. Gaddam, C. Griwodz, and D. Johansen. Bagadus: An integrated system for arena sports analytics – a soccer case study. In *Proc. of ACM MMSys*, pages 48–59, Mar. 2013.

[8] A. Mavlankar and B. Girod. Video streaming with interactive pan/tilt/zoom. In M. Mrak, M. Grgic, and M. Kunt, editors, *High-Quality Visual Experience*, Signals and Communication Technology, pages 431–455. Springer Berlin Heidelberg, 2010.

[9] S. Sægrov, A. Eichhorn, J. Emerslund, H. K. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen. Bagadus: An integrated system for soccer analysis (demo). In *Proc. of ICDSC*, Oct. 2012.

[10] X. Sun, J. Foote, D. Kimber, and B. Manjunath. Region of interest extraction and virtual camera control based on panoramic video capturing. *IEEE Transactions on Multimedia*, 7(5):981–990, 2005.

[11] M. Tennøe, E. Helgedagsrud, M. Næss, H. K. Alstad, H. K. Stensland, V. R. Gaddam, D. Johansen, C. Griwodz, and P. Halvorsen. Efficient implementation and processing of a real-time panorama video pipeline. In *Proc. of IEEE ISM*, Dec. 2013.

[12] J. Wang, C. Xu, E. Chng, K. Wah, and Q. Tian. Automatic replay generation for soccer video broadcasting. In *Proc. of ACM MM*, pages 32–39, 2004.

[13] M. Wieland, R. Steinmetz, and P. Sander. Remote camera control in a distributed multimedia system. In B. Wolfinger, editor, *Innovationen bei Rechen- und Kommunikationssystemen*, Informatik aktuell, pages 174–181. Springer Berlin Heidelberg, 1994.

Appendix H

[Conference] Tiling of Panorama Video for Interactive Virtual Cameras: Overheads and Potential Bandwidth Requirement Reduction

[Authors:] **V. R. Gaddam**, H. B. Ngo, R. Langseth, C. Griwodz, D. Johansen, and P.Halvorsen
[Published:] IEEE Picture Coding Symposium (PCS), 2015.

Tiling of Panorama Video for Interactive Virtual Cameras: Overheads and Potential Bandwidth Requirement Reduction

Vamsidhar Reddy Gaddam*§, Hoang Bao Ngo*, Ragnar Langseth†, Carsten Griwodz*,
Dag Johansen‡ and Pål Halvorsen*
*Simula Research Laboratory & University of Oslo, Norway
†ForzaSys AS, Norway
‡UiT The Arctic University of Norway, Norway
§vamsidhg@ifi.uio.no

***Abstract*—Delivering high resolution, high bitrate panorama video to a large number of users introduces huge scaling challenges. To reduce the resource requirement, researchers have earlier proposed tiling in order to deliver different qualities in different spatial parts of the video. In our work, providing an interactive moving virtual camera to each user, tiling may be used to reduce the quality depending on the position of the virtual view. This raises new challenges compared to existing tiling approaches as the need for high quality tiles dynamically change. In this paper, we describe a tiling approach of panorama video for interactive virtual cameras where we provide initial results showing the introduced overheads and the potential reduction in bandwidth requirement.**

***Keywords*—Panorama, Virtual Camera, Bandwidth, Real-time Systems, Video**

I. Introduction

High-resolution, wide field-of-view video has lately received an increased attention in various scenarios like surveillance, sports and health care. For instance, in sports, many game analysis systems provide camera arrays where individual camera images are stitched together to cover the entire field. From the panorama video, one can further generate virtual cameras supporting personalized views which are zoomed, panned and tilted. One example is given in figure 1 where a virtual camera allows an individual user to interactively control an own personalized view, i.e., extracting pixels from parts of the stitched panorama video such that the area extracted is not a simple crop from the high-resolution panorama.

In this context, we have earlier presented the Bagadus system [1] where we generate panorama videos of a soccer stadium in real-time from which individual users can be their own cameraman and interactively control their own virtual view [2]. Furthermore, in our earlier version [2], the system streamed the cylindrical panorama video to the remote clients which extracted a perspective-corrected camera view on the client side. However, popular soccer games may attract large numbers of concurrent users, e.g., during the 2014 FIFA World Cup, their web player had about 24 million unique users [3]. As the bandwidth requirement for streaming the entire, full-quality panorama is rather high, we therefore have a large challenge in providing real-time virtual camera services in such scales.

We are soon demonstrating [4] an early prototype as a proof-of-concept. In this paper, we have extended this work, and we analyze the overheads and potentials of such a solution. In this paper, we present the costs of tiling approaches and the potential bandwidth savings exploiting multi-quality encoding. The contributions in this paper are two-fold. Our key novelty theoretically lies in changing constant rate factor (CRF) instead of the size of the video tiles in order to change the quality. To realize this idea practically, we present the architecture of a real system that can support tiling. Furthermore we evaluate trade-offs with respect to added overheads, like increased processing and storage space, and reduced bandwidth requirement. Depending on the zoom and panning actions of the user, the results indicate that there is a large potential for reducing the transfer cost by trading quality in the areas of the panorama that are most likely unused.

The rest of this paper is organized as follows: Section II describes the costs of scaling virtual views to several users. We briefly present our novelty in contrast to the state-of-the-art in section III. We then provide a brief description and evaluation of our system in sections IV and V, respectively. Then, we conclude our paper and present the prospective direction in section VI.

II. The Costs of Virtual Views

Our system generates a cylindrical panorama video in real-time from which individual *virtual views* can be generated and

Fig. 1: The re-projected virtual view. The panorama video with the marked region of interest is shown together with the generated virtual camera.

interactively controlled with full freedom to pan, tilt and zoom into the panorama video [2]. When it comes to delivering video to the client, we have explored two possibilities with respect to creating virtual views.

Initially, our system transfers the entire panoramic video and generates the virtual views on the client [2]. This gives cheap processing requirements on the server-side at the cost of very high bandwidth requirements on the client-side. In our example system installed at Alfheim stadium, the average size of each *1-second* segment of the the 4096×1680 panorama video (figure 1) is approximately 1.25 *MB*[1]. This means that the bandwidth requirement for each client becomes about 10 Mbps merely for the transfer of the panorama video, and in future systems, a much higher resolution panorama may be desirable to support better digital zoom. Then, after the panorama is successfully transferred, the client needs to process it so that a virtual view can be extracted. Using commodity graphics hardware, the virtual view can be extracted in real-time regardless of the size of the view [2]. Thus, the client devices easily manage the processing load, but the bandwidth requirement is quite high as mentioned above.

An alternative approach is to generate the virtual views on the server and only stream the generated video to the client, i.e., as a traditional streaming case. Thus, in this approach, the re-projection is performed on the server side. This approach requires nothing more than a browser that is capable to play a video on the client device, i.e., it severely reduces the computational load and the bandwidth requirements on the clients. However, the processing cost on the server-side becomes huge as additional encoding must be performed per stream, and it quickly becomes a large bottleneck. We have made a few experiments using the second generation hardware from Nvidia. Our experiments show that the GeForce GTX 750 Ti GPU can encode 16 full HD video streams at 30 frames per seconds [5]. We also found that this was the limiting factor in the number of unique views we could create in real-time. This implies that if we want to provide a service to say 100,000 concurrent users, we would require a cluster totaling to about 6,250 GPU devices. Such an initial installation costs, at the time of writing, about 937,500 USD merely for the GPUs.

Owing to the challenges mentioned above, no straightforward solution is going to work well for scaling our system to large numbers of concurrent users. However, the HTTP streaming solutions have proved to scale well from a sending-side point of view using for example CDNs. We have therefore adopted this approach, i.e., using a client side generated virtual view where the panorama is tiled to save bandwidths in areas not used for the virtual view extraction.

III. Related Work

Based on the decision in the previous section, to generate virtual views on the client side, the challenge is to reduce the cost of streaming a complete panorama video to each user. In this respect, researchers in the multimedia community have for some time analyzed region-of-interest streaming solutions. For example, tiling is discussed in [6], [7], [8], [9], [10]. Furthermore, [11], [12], [13], [14], [15] extensively address the problem of automatically generating personalized content, and [16] discusses plain cropping.

The works related to tiling can be broadly classified into two scenarios, the *server side* generation [6], [8], [9], [10] and the *client side* generation [17], [7]. The former emphasize on reducing the through-put on the server side because their processing happens on server and the video is merely transferred to the client for display. The latter works with a client which decodes and assembles tiles. However, the idea in both the scenarios is to downsample the panorama into multiple lower resolutions and transfer only the *additional info* when it is required. This approach has traditionally been used to reduce the number of samples in a signal. We exploit the fact that the panoramic camera systems in scenarios like sports and surveillance are most likely to be static. In the case of a static scene with some local movement in small regions H.264 CRF provides a more efficient compression than downscaling the video. Figure 2 demonstrates[2] the key difference from the existing work in tiling and the strength in our assumption. By using tiles encoded using different CRF, we can allow for better quality yet compressing to the same extent as the downsampled version.

(a) CRF varied - 158 kbps (b) Down scaled - 157 kbps

Fig. 2: Two output frames are presented to show the key difference between our method and other common approaches. The loss in sharpness due to down-scaling can be observed.

Similar to many other approaches, our solution is based on dividing the panorama into tiles as shown in figure 3, each encoded as an adaptive HTTP stream using for example HLS. A client retrieves segments in high quality for segments being used for the virtual camera, and the rest of the tiles are retrieved in lower quality depending on the strategy used. In contrast to, for example, [18], [9] retrieving only tiles in the region of interest, we need to retrieve all tiles since the virtual camera moves and at least low quality data needs to be available if the user zooms out or moves quickly.

Another difference is that the tiles fetched do not follow a strict logic apart from being in the neighborhood of the current tiles. In [10], for instance, all the tiles are being fetched, but the reduction in quality is reflected by salience. Moreover, the non-linear nature of a panorama-virtual view transformation introduces further complexities in the approach. For example, in figure 4, it can be seen that the collection of required tiles do not form any simple shape like a rectangle or a square, e.g., as used in [16]. This poses different challenges than the ones that are being tackled in for example [7] where the panning and

[1]This size depends on the lighting and weather conditions.

[2]Due to possible poor quality of printers, it is recommended to analyze the images on screen.

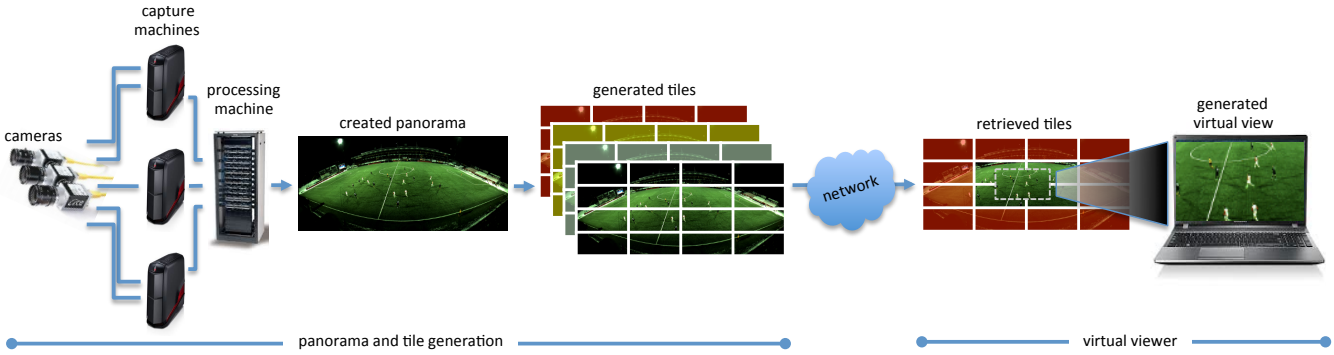


Fig. 3: The generated cylindrical panorama video is tiled and encoded in multiple qualities on the server side. The client side uses the current position of the virtual camera to retrieve full quality tiles for the virtual view and low quality (red) tiles outside the field of view of the virtual camera.

tilting correspond to strictly navigating the panorama along the horizontal and vertical directions respectively. Lack of this adds complexity on the tile retrieval strategy. In addition to taking into account available network bandwidth and client device resources (as usually done for *one* stream), the quality adaption strategy must also coordinate the tile qualities according to the dynamic position of the virtual camera.

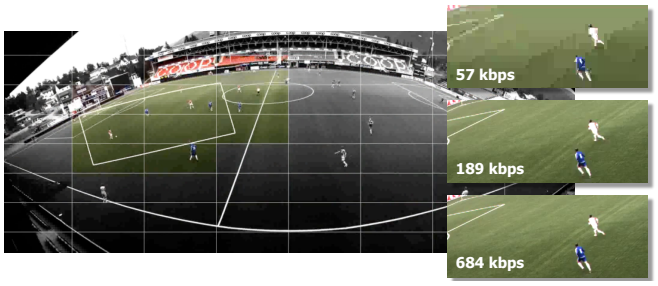


Fig. 4: Dividing the panorama video into 8x8 tiles, and encoding each tile in different quality

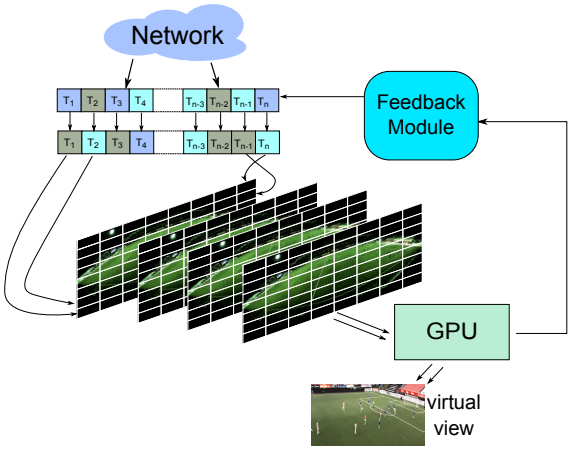


Fig. 5: The architecture on the client side to support tiling. The bottom array of tiles is the one that is being used to display the current view, the top array contains the tiles that are being fetched in a quality depending on the current viewing parameters.

IV. System Overview

Figure 3 shows the high level architecture of our system. The fundamental requirement for a smooth interactive playout is that the processing on the server and the client side must be performed within a strict real-time deadline.

A. Server-side

As one can see in figure 3, the creation of tiles and encoding them in multiple qualities are done on the server side. For the encoding, we use *libav* and *x264*. The qualities are determined by modifying the CRF from the highest to the lowest quality. Unlike constant quantization parameter (CQP), where it compresses every frame in a video by the same amount, the compression rate of each frame in CRF is decided by motion. It takes into account how the human eye perceives information of still and moving objects. Thus, we get better quality even with less bitrate in the encoded videos.

B. Client-side

One of the key contributions lie in the system implementation, where we present a working prototype for a virtual viewer using tiling. Figure 5 shows the architecture of our system. A tile manager fetches the appropriate quality tile at a position on the panorama, then it decodes the tile-segment and creates the corresponding part of the panorama frame. This manager runs concurrently in multiple threads, because there is no overlap between tiles. Once an entire panorama frame is decoded, it is transmitted to the GPU where the virtual view extraction takes place as mentioned in [2]. The view information is passed into the *Feedback Module* where quality selection is performed based on the currently viewed region on the panorama.

We implemented a simple feedback scenario where the next tile is fetched in either high or low quality depending on the current view. A greedy *binary scheme* where the high quality (b_h) is fetched even if one pixel from the tile is in the current virtual view and low quality (b_l) otherwise. Due to the

pipelined nature of the client and the finite size of each video segment, there is a latency of one segment size in updating the tile. However, this can be taken care of by employing some sort of prediction strategies [17].

V. Evaluation

For the sake of evaluation, we created several paths in different classes. The paths belong to different classes depending on their zoom levels. The reason for doing this is to evaluate the acceptance of different qualities and potential bandwidth at different zooms. We selected four zoom scenarios and recorded multiple paths in each of these scenarios mimicking a user that is trying to follow the game. The scenarios are zoomed-in (ZI), zoomed-out (ZO), medium-zoomed (ZM) and random zoom (ZR). The only difference is that in the first three scenarios the user has limited/no control over the zoom factor, and in the last scenario the user has full freedom to change the zoom factor. Apart from the zoom factor, the user is free to pan and tilt as she/he wishes in all the scenarios. We divide the panorama into 8×8 tiles and each tile is encoded into 5 different qualities ranging from highest to lowest [0 to 4]. We present results for selecting [$b_h = 0$, $b_l = 4$] and [$b_h = 2$, $b_l = 3$] in this paper[3]. An example of the output can be seen in figure 6. It can be observed that even though some parts of the panorama have extremely poor quality, the virtual view does not show the distortion.

Fig. 6: Reconstructed panorama from multi-quality tiles and the corresponding virtual view. The poor quality in some areas of panorama does not affect the quality of the virtual view.

For performance measurements, we used two machines, both equipped with an Intel Core i7-4770, 8GB RAM and SSD harddisk, one as client and the other as server. In general one can expect a machine of higher capabilities on the server side and of lower capabilities on the client side.

A. Reduced bandwidth

It can be observed in figure 7 that there is a significant reduction in bandwidth required to perform the virtual view operation by using tiling. The reduction in bandwidth is of course dependent on the number of tiles that are required in high quality, i.e., the plot shows that most bandwidth is saved when the user is zoomed in compared to the case of zoomed out. However, it must be noted that it is a simple binary scheme of selecting high(b_h)/low(b_l) quality depending only on the current viewing region. The bandwidth also varies depending on the values chosen for b_h and b_l.

[3]Videos: http://home.ifi.uio.no/vamsidhg/pv2015/

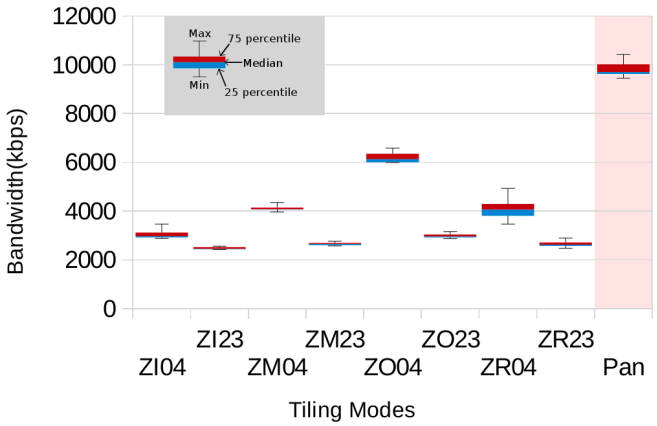


Fig. 7: The bandwidth during a 30 second virtual-view operation. The zoom-scenarios are zoomed-in (ZI), zoomed-out (ZO), medium-zoomed (ZM) and random zoom (ZR). 04 implies highest-lowest quality and 23 implies medium-low quality. As a reference, the plot also includes the results for the full panorama. The legend is equal for all figures.

B. Increased storage

The idea of delivering tiles at multiple qualities certainly comes with a cost of storage on server side. However, with the current costs of storage the increase is not that significant.The total size of full quality panorama video for 5 minutes is 362.5 MB, and the sum total for the tiles is 983.1 MB (398.2 MB, 281.9 MB, 145.2 MB, 92.0 MB and 58.4 MB for each individual quality), i.e., tiling gives approximately a $3\times$ storage requirement storing all the tiles in multiple qualities compared to the single full quality panorama.

C. Server-side processing

Our tiling component is still a prototype, and we have only tested a few different approaches on how to encode tiles efficiently. The first approach was to encode each tile sequentially, i.e., each tile is encoded into different qualities completely (figure 4) before we start with the next tile. The libav tools will optimize the encoding by using several threads on each tile. Thus, we can use the CPU efficiently. The second approach is parallelization where we encode several tiles concurrently.

In figure 8, we can observe a huge difference between sequential encoding of tiles and encoding a full panorama. However, it must be noted that the full panorama encoding happens only for one quality but the other measurements include the time taken for all 5 qualities. Hence, the comparision is between encoding 1 video at 4096×1680 to encoding 8×8

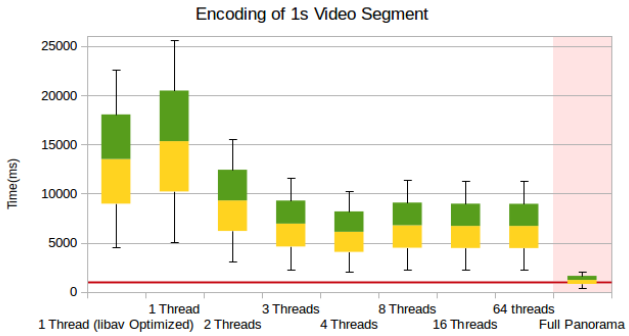


Fig. 8: The time to partition and encode the tiles of the full panorama into 5 different qualities using multiple threads. The encoding of a full panorama video in full quality is included as a reference.

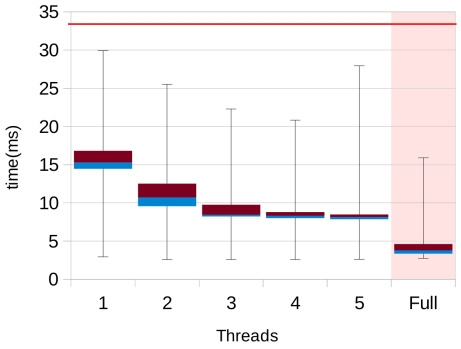


Fig. 9: Time to decode one frame using multiple threads in tiling case in comparison with the full panorama. The real-time 30fps deadline is marked as dashed line.

tiles, each in 5 different qualities - in total 320 video streams at 512×210 resolution. Even when libav is free to spawn multiple threads to encode for each tile, we would have to create several new threads to encode a tile and terminate them before continuing to the next, thus resulting in a huge overhead. We have also performed a second test using completely sequential encoding, i.e., one thread without libav optimization. As can be seen in figure 8, there is a noticeable difference when using optimization from libav.

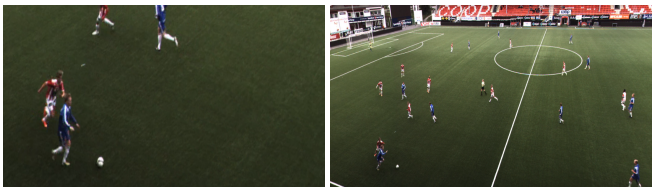
Instead of letting libav spawn threads for every tile, we created a thread pool of encoding workers and let each thread in that pool run concurrently on different tiles. we experimented with different number of threads in the thread pool. Our experimental results (figure 8) display a convex curve between the 1 thread and 64 threads performance with the 4 threads as the best case. In our case with a 4-core CPU, this result is expected since encoding is CPU bound. When using only 1-3 threads, we are not utilizing the 4-core CPU efficiently. Using more than 4 threads gives a lot of context-switching. Both scenarios increase the processing time compared to the 4-thread approach.

Obviously, the performance between multiple threads differs based on the CPU used for encoding. With the hardware we used for these experiments, we have not succeeded to fulfill the real-time requirement of encoding the tiles under 1 second. However the processing requirement has upper bounds and does not depend on the number of users. Hence, this will not pose further scaling issues.

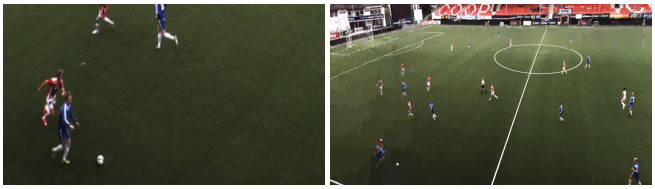
D. Client-side processing

As the complexity in figure 5 suggests and figure 9 reflects, there is a significant overhead in decoding the tiles. The tiling approach requires the tiles to be decoded across the entire panorama. This increases complexity of the viewer from the non-tiling case of decoding 1 video to decoding 64 videos simultaneously and still keeping the framerate. However the total resolution of the panorama frame that is being decoded does not change, it is just divided among 64 parts. We again used libav to decode the videos in both cases. One serious limitation of libav is that there is a global context that serializes several calls to the library at the process level. Another overhead in the tiling approach comes from the feedback module. On average, about 15 ms per frame are spent on the feedback module to decide the upcoming tile qualities.

E. Quality



(a) Full Pano. Data: 10425 kbps (b) Full Pano. Data: 10425 kbps



(c) binary - $b_h = 0$ and $b_l = 4$ SSIM: 0.9807 Data: 2912 kbps (d) binary - $b_h = 0$ and $b_l = 4$ SSIM: 0.9088 Data: 6575 kbps

(e) binary - $b_h = 2$ and $b_l = 3$ SSIM: 0.9574 Data: 2566 kbps (f) binary - $b_h = 2$ and $b_l = 3$ SSIM: 0.9433 Data: 3154 kbps

Fig. 10: Different qualities of output virtual view.

One basic requirement of tiling should be that quality of experience should not be altered heavily even with reduction in bandwidth. However, we do not perform any subjective studies for this paper. We intend to perform user studies to evaluate different tiling approaches in the future. Figure 10 presents objective quality results from using two approaches, $[b_h = 0,\ b_l = 4]$ and $[b_h = 2,\ b_l = 3]$ for tiling. The sizes of data downloaded during the operation of 30 seconds for each approach are also presented in the figure. It can be

seen that the quality is different depending on the zoom factor and the approach selected. The first column presents a case where most tiles in the view are from b_h quality input and the second column presents one of the worst case scenarios where about 30% of the pixels in the view are from b_l quality tiles. The SSIM values from figure 10 suggest that b_h requires to be highest quality when the view is a really zoomed-in view, but even an average quality b_h tile can work for a zoomed-out view. On the basis of these results we argue that a more inclusive strategy can be designed to maximize viewing experience and reduce the required bandwidth. In general, the choice of CRF variation across the tiles provides superior image quality at the same bit-rate as can be seen in figure 2.

F. Segment duration

The segment duration plays an important role in deciding the size of each file and hence the bandwidth. In the current tiling system, we chose the segment duration to be one second each, but in the non-tiled version we use 3 second segments. In general, the segmentation approach implies that each segment contains at least one I-frame each. Thus, increasing the segment duration to 3 seconds reduces the size of files significantly, but the tradeoff is that this increases the overall quality adaption latency in the system to at least 3 seconds.

VI. Conclusions

We have presented a system for real-time interactive zooming and panning of a tiled panorama video enabling every user to be her or his own cameraman [2]. To reduce the per-user bandwidth requirement, the idea is that the quality changes in different parts of the panorama video when moving the virtual camera, i.e., retrieving good quality for the used tiles and lower quality in the rest of the video. Furthermore, we have evaluated the costs of the tiling approach and the potential bandwidth savings exploiting multi-quality encoding. Depending on the zoom and panning actions of the user, the results indicate that there is a large potential for reducing the transfer cost by trading quality in areas of the panorama not used for the extraction of the virtual view.

There is still unfinished work as the algorithm deciding the tile quality is very basic, but we are looking at better approaches, e.g., having a degrading quality depending on the view focus and the distance to the view, predicting the movement of the virtual camera (e.g., following the ball). We are also investigating approaches to distribute the server-side processing as it is far from the real-time requirement. However, we have earlier demonstrated that the GeForce GTX 750 Ti GPU can encode 16 full HD video streams at 30 frames per seconds [5], but our tiling system must be ported for new experiments.

References

[1] P. Halvorsen, S. Sægrov, A. Mortensen, D. K. Kristensen, A. Eichhorn, M. Stenhaug, S. Dahl, H. K. Stensland, V. R. Gaddam, C. Griwodz, and D. Johansen, "Bagadus: An integrated system for arena sports analytics – a soccer case study," in *Proc. of ACM MMSys*, Mar. 2013, pp. 48–59.

[2] V. R. Gaddam, R. Langseth, S. Ljødal, P. Gurdjos, V. Charvillat, C. Griwodz, and P. Halvorsen, "Interactive zoom and panning from live panoramic video," in *Proc. of ACM NOSSDAV*, 2014, pp. 19:19–19:24. [Online]. Available: http://doi.acm.org/10.1145/2578260.2578264

[3] Fédération Internationale de Football Association, "2014 FIFA World Cup breaks online streaming records," http://www.fifa.com/aboutfifa/organisation/news/newsid=2401405/, last accessed: 2014-12-19.

[4] V. R. Gaddam, R. Langseth, H. K. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen, "Scaling virtual camera services to a large number of users[accepted]," in *Proc. of ACM MMSys*, 2015.

[5] M. A. Wilhelmsen, H. K. Stensland, V. R. Gaddam, P. Halvorsen, and C. Griwodz, "Performance and Application of the NVIDIA NVENC H.264 Encoder," http://on-demand.gputechconf.com/gtc/2014/poster/pdf/P4188_real-time_panorama_video_NVENC.pdf, last accessed: 2014-12-19.

[6] R. Guntur and W. T. Ooi, "On tile assignment for region-of-interest video streaming in a wireless LAN," in *Proc. of NOSSDAV*, 2012, p. 59. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2229087.2229105

[7] A. Mavlankar and B. Girod, "Video streaming with interactive pan/tilt/zoom," in *High-Quality Visual Experience*, ser. Signals and Communication Technology, M. Mrak, M. Grgic, and M. Kunt, Eds., 2010, pp. 431–455. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12802-8_19

[8] K. Q. M. Ngo, R. Guntur, and W. T. Ooi, "Adaptive encoding of zoomable video streams based on user access pattern," in *Proc. of MMSys*, 2011, p. 211. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1943552.1943581

[9] A. Shafiei, Q. M. K. Ngo, R. Guntur, M. K. Saini, C. Pang, and W. T. Ooi, "Jiku live," in *Proc. of ACM MM*, 2012, p. 1265. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2393347.2396434

[10] H. Wang, V.-T. Nguyen, W. T. Ooi, and M. C. Chan, "Mixing tile resolutions in tiled video: A perceptual quality assessment," in *Proc. of NOSSDAV*, 2013, pp. 25:25–25:30. [Online]. Available: http://doi.acm.org/10.1145/2578260.2578267

[11] F. Chen and C. De Vleeschouwer, "Personalized production of basketball videos from multi-sensored data under limited display resolution," *Comput. Vis. Image Underst.*, vol. 114, no. 6, pp. 667–680, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2010.01.005

[12] N. Babaguchi, Y. Kawai, and T. Kitahashi, "Generation of personalized abstract of sports video," in *Proc. of ICME*, Aug 2001, pp. 619–622.

[13] R. Kaiser, M. Thaler, A. Kriechbaum, H. Fassold, W. Bailer, and J. Rosner, "Real-time person tracking in high-resolution panoramic video for automated broadcast production," in *Proc. of CVMP*, 2011, pp. 21–29.

[14] X. Sun, J. Foote, D. Kimber, and B. Manjunath, "Region of interest extraction and virtual camera control based on panoramic video capturing," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 981–990, 2005.

[15] R. Xu, J. Jin, and J. Allen, "Framework for script based virtual directing and multimedia authoring in live video streaming," in *Proc of MMM*, Jan 2005, pp. 427–432.

[16] R. Heck, M. Wallick, and M. Gleicher, "Virtual videography," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 3, no. 1, pp. 4–es, Feb. 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1198302.1198306

[17] A. Mavlankar and B. Girod, "Pre-fetching based on video analysis for interactive region-of-interest streaming of soccer sequences," in *Proc. of ICIP*, Nov. 2009, pp. 3061–3064. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5414201

[18] M. Inoue, H. Kimata, K. Fukazawa, and N. Matsuura, "Interactive panoramic video streaming system over restricted bandwidth network," in *Proc. of ACM MM*, 2010, p. 1191. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1873951.1874184

Appendix I

[Demo] Be Your Own Cameraman: Real-Time Support for Zooming and Panning into Stored and Live Panoramic Video

[Authors:] **V. R. Gaddam**, R. Langseth, H. K. Stensland, P. Gurdjos, V. Charvillat, C. Griwodz, D. Johansen, and P. Halvorsen
[Published:] ACM International Conference on Multimedia Systems (MMSys), 2014.

Be Your Own Cameraman: Real-Time Support for Zooming and Panning into Stored and Live Panoramic Video

Vamsidhar Reddy Gaddam[1], Ragnar Langseth[1], Håkon Kvale Stensland, Pierre Gurdjos[2], Vincent Charvillat[2], Carsten Griwodz[1], Dag Johansen[3], Pål Halvorsen[1]

[1]Simula Research Laboratory & University of Oslo, Norway
{vamsidhg, ragnarla, haakonks, griff, paalh}@ifi.uio.no

[2]Universite de Toulouse, France
{pgurdjos, vincent.charvillat}@enseeiht.fr

[3]Univerity of Tromsø, Norway
dag@cs.uit.no

ABSTRACT

High-resolution panoramic video with a wide field-of-view is popular in many contexts. However, in many examples, like surveillance and sports, it is often desirable to zoom and pan into the generated video. A challenge in this respect is real-time support, but in this demo, we present an end-to-end real-time panorama system with interactive zoom and panning. Our system installed at Alfheim stadium, a Norwegian premier league soccer team, generates a cylindrical panorama from five 2K cameras live where the perspective is corrected in real-time when presented to the client. This gives a better and more natural zoom compared to existing systems using perspective panoramas and zoom operations using plain crop. Our experimental results indicate that virtual views can be generated far below the frame-rate threshold, i.e., on a GPU, the processing requirement per frame is about 10 milliseconds.

The proposed demo lets participants interactively zoom and pan into stored panorama videos generated at Alfheim stadium and from a live 2-camera array on-site.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Video; I.4.9 [**Applications**]: Video

General Terms

Experimentation, Performance

MMSys'14, March 19-21, 2014, Singapore, Singapore
ACM 978-1-4503-2705-3/14/03.
http://dx.doi.org/10.1145/2557642.2579370

Keywords

demonstration, panorama video, zoom, panning, real-time

1. INTRODUCTION

Panoramic video is becoming increasingly more popular in various scenarios. In the context of arena sports like soccer, it provides opportunities for innovation in broadcasting and in the area of game analysis. For example, based on a camera array covering the whole field, virtual views can be supported by zooming and panning into a stitched panorama video, e.g., allowing individual users to interactively control his or her own virtual camera. This type of challenges has attracted researchers' attentions for more than a decade [15, 17, 5, 7, 11, 6, 18], but often the zoom is made by simple cropping, or it does not scale. As can be seen in the literature, the challenges in providing such interactive experiences to a large audience span several fields, and one such challenge is to provide a real-time pannable and zoomable virtual camera to several thousands or even millions of users.

In this respect, we present an end-to-end real-time system to interactively zoom and pan into high-resolution panoramic videos. In contrast to many existing systems supporting zoom by cropping into a perspective panorama video, our system uses cylindrical panorama as an intermediate representation. Here, the perspective of the virtual view is corrected in real-time, and the result is a better and more natural zoom. The client has full freedom to pan, tilt and zoom into the panorama video using the system. The system supports (live switching between) several algorithms for pixel interpolation. Furthermore, our experimental results indicate that such zoomed virtual views can be generated far below the frame-rate threshold where an unoptimized GPU implementation generates a virtual view frame in about 10 ms. Thus, taking into account recent trends in device development, our approach should be able to scale to a large number of concurrent users in the near future to spectators and fans both within the stadium as well as outside. Inside the stadium, the Cisco connected stadium solution [4] already in use in more than 200 venues world-wide can be used to

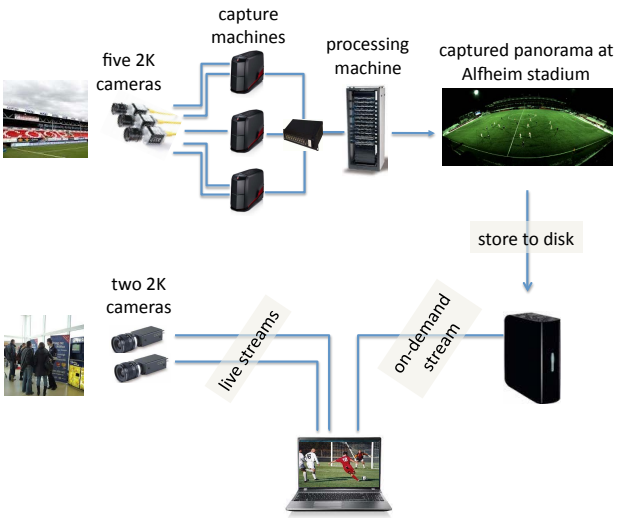


Figure 1: Demonstration setup

broadcast the panorama video to the spectators, and for users at home, there exist several scalable streaming solutions like segmented HTTP streaming [3, 13] or panorama video transfers in real-time [9].

In this demonstration, we present two versions of our system. The first is a demonstration which uses a recorded five 2K-camera panorama video from Alfheim stadium where the current prototype is installed. The second demonstration is a live, scaled down version[1] of the system with two 2K cameras zooming into live recorded video at the conference venue. The motivation is to provide a hands-on experience of such a system. The demo will allow users to interact with the virtual camera in both stored and live scenes.

2. SYSTEM OVERVIEW

The system presented in this demo is part of a larger system where the current prototype is running at Alfheim stadium in Tromsø. This system has been presented before [10, 14], but without the possibility to interactively zoom an pan into a high-resolution panorama.

2.1 System Setup

The relevant part of the system in this context is divided into a panorama generation part and a video delivery part, which supports our user-controlled interactive virtual cameras. A sketch of our system is given in figure 1. In this section, we describe these two parts.

2.1.1 *Real-time Panorama Video Capture*

To capture the entire soccer field, we use five Basler industry vision cameras [2], each of which delivers a maximum resolution of 2046×1086 pixels at 50 frames per second over Gigabit Ethernet. We use an 8 mm lens [1] with virtually no image distortion, allowing us to bypass the lossy debarreling step of our previous real-time pipeline [16]. To maximize the panorama resolution, the cameras are rotated by 90° (see figure 2), giving a vertical field of view of 66°. Furthermore, the cameras are mounted in a circular pattern. This means that they are pitched, yawed and rolled to look directly through a point 5 cm in front of the lenses, in an attempt to reduce parallax effects. As a result of this, only minor adjustments are required before the images can be stitched together. The video capture system also determines the required exposure, which requires frequent changes due to quickly changing light conditions outdoors. In this respect, auto-exposure is performed on the center camera once every 4 seconds, and the camera reader module broadcasts the resulting exposure parameters to the other camera readers.

Figure 2: Mounted camera array

After the generation of the panorama video, it is encoded and compressed using x264. At the moment, with a focus on video quality, the system still demands a large network bandwidth to the full resolution panorama video. However, there are large potentials for trading of quality for lower bandwidth requirements (but an investigation of this is out of scope for this demonstration).

2.1.2 *Live Panoramic Zoom and Panning*

To deliver a virtual camera to clients, the panorama video is delivered to the client device. Currently, we use HTTP segment streaming (no quality adaption at the time of writing, but it can easily be added in the future). The segments of the panoramic videos are served by an Apache server along with a manifest file. The manifest file is used to inform the clients when the next file is ready for download. In case of live streaming, the viewer checks the manifest file periodically and downloads the next segment when ready. As soon as the panoramic video segment is transferred, it is decoded for processing by the multi-threaded client process. The downloading thread runs in the background without blocking either the display thread or the user input thread.

Using the cylindrical panorama as an intermediate representation, our system then generate an arbitrary virtual camera view from the position of the camera array. All the details of the pixel calculations are out of scope of this demonstration description (see [8]), but the main operation is to find the pixels of the virtual camera image from the cylindrical texture as shown in figure 3. A pin-hole camera model is used, and the a point projection from a 3D point to an image point is calculated using a 3D rotation matrix as a function of the rotation angles around the x, y and z axes. A particular pixel is then found by calculating the ray that passes from the camera center to the pixel. The intersection of this ray with the cylinder gives the exact position on the cylindrical texture. Finally, since there is sub-pixel accu-

[1]The live 2-camera setup is due to hardware constraints bringing equipment to the conference venue. The five camera version requires several recording machines and a processing machine. A live view from the camera setup at Alfheim stadium could be possible, but as we sometimes have bandwidth problems over the 19-hop Tromsø-Oslo link, we expect an even lower available bandwidth from Tromsø to Singapore.

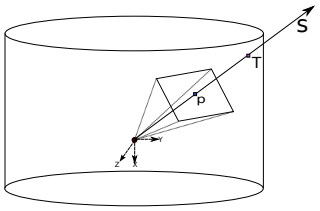


Figure 3: The intersection of the ray from the virtual view with the unit cylinder of the panorama image.

racy of the intersection calculation, the used pixel value is calculated using interpolation of all the surrounding pixels.

The user has the full freedom to pan, tilt and zoom, being his or hers own camera-man[2]. The virtual camera view is corrected to a perspective view very similar to that of a physical camera as shown in figure 4.

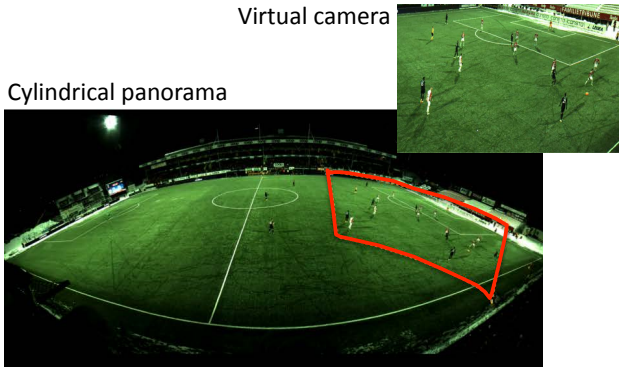


Figure 4: The virtual camera is generated from the region of interest marked in the panorama video. Note that it is not a simple crop from the bigger video.

2.2 System Performance

We have earlier shown that our system can generate panorama video in real-time using four 1K cameras [16]. Going for higher resolution, we have now distributed the capture and processing of five 2K cameras, and we are able to generate the panorama frames in 34.58 ms on an Intel i7 CPU. This is below our 25 fps threshold. The generated video can either be stored to disk or forwarded directly to connected client devices.

When presented to the user, the zooming and panning operations can be performed on both the sender and receiver side. In the current setup, we as explained above transmit the panorama video to the display device like a laptop. Here, the zoom and pan operations are executed where the perspective is corrected from the cylindrical panorama in real-time. We use the GPU on the client side, and each frame for virtual camera view with Full HD resolution (1920×1080) can be generated in about 10 milliseconds on a machine with an Intel i7-2600 CPU and an Nvidia GeForce GTX 460 GPU – again, far below the target 25 fps requirement. This time is dependent of the output resolution, and going for a mobile phone resolution it can be further reduced by a millisecond or two, e.g., about 1.5 milliseconds for iPhone 4 and 5.

[2]An example video using the known ball position, where the virtual camera-man is following the ball is shown here: `http://www.youtube.com/watch?v=554RjEEtw3o`

3. DEMONSTRATION

In this demo, we present a system for real-time interactive zooming and panning of panorama video. The general setup is shown in figure 1. We use two types of panorama input. We use stored panorama video captured in the Alfheim soccer stadium (the European League game between Tromsø IL and Tottenham Hotspurs) [12]. The video was recorded using five 2K industrial cameras, and processed and stitched in a distributed system. We also use a live system using 2 cameras on site to generate the panorama video. Both the stored-video and the live-video demos work in real-time, and the user interactions have a very low delay from command to visual response – thus, a user can be his or her own cameraman!

Acknowledgments

This work has been performed in the context of the *iAD* center for Research-based Innovation (project number 174867) funded by the Norwegian Research Council. Furthermore, the authors also acknowledge the support and hardware given by Hugo Kohmann and Roy Nordstrøm in Dolphin Interconnect Solutions.

4. REFERENCES

[1] Azure-0814m5m. http://www.azurephotonicsus.com/products/azure-0814M5M.html.

[2] Basler aca2000-50gc. http://www.baslerweb.com/products/ace.html?model=173.

[3] SmoothHD. http://www.smoothhd.com, 2009.

[4] Sports & Entertainment – Cisco Connected Stadium, 2013. http://www.cisco.com/web/strategy/sports/index.html.

[5] Y. Ariki, S. Kubota, and M. Kumano. Automatic production system of soccer sports video by digital camera work based on situation recognition. In *Proc. of ISM*, pages 851–860, 2006.

[6] P. Carr and R. Hartley. Portable multi-megapixel camera with real-time recording and playback. In *Proc. of DICTA*, pages 74–80, 2009.

[7] P. Carr, M. Mistry, and I. Matthews. Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording. In *Proc. of ACM MM*, pages 193–202, 2013.

[8] V. R. Gaddam, R. Langseth, S. Ljødal, P. Gurdjos, V. Charvillat, C. Griwodz, and P. Halvorsen. Interactive zoom and panning from live panoramic video. In *Proc. of ACM NOSSDAV*, Mar. 2014.

[9] R. Guntur and W. T. Ooi. On tile assignment for region-of-interest video streaming in a wireless lan. In *Proc. of NOSSDAV*, pages 59–64, 2012.

[10] P. Halvorsen, S. Sægrov, A. Mortensen, D. K. Kristensen, A. Eichhorn, M. Stenhaug, S. Dahl, H. K. Stensland, V. R. Gaddam, C. Griwodz, and D. Johansen. Bagadus: An integrated system for

arena sports analytics – a soccer case study. In *Proc. of ACM MMSys*, pages 48–59, Mar. 2013.

[11] A. Mavlankar and B. Girod. Video streaming with interactive pan/tilt/zoom. In M. Mrak, M. Grgic, and M. Kunt, editors, *High-Quality Visual Experience*, Signals and Communication Technology, pages 431–455. Springer Berlin Heidelberg, 2010.

[12] S. A. Pettersen, D. Johansen, H. Johansen, V. Berg-Johansen, V. R. Gaddam, A. Mortensen, R. Langseth, C. Griwodz, H. K. Stensland, and P. Halvorsen. Soccer video and player position dataset. In *Proc. of ACM MMSYS*, Mar. 2014.

[13] R. Pantos (ed). HTTP Live Streaming. http://www.ietf.org/internet-drafts/draft-pantos-http-live-streaming-10.txt, 2013.

[14] H. K. Stensland, V. R. Gaddam, M. Tennøe, E. Helgedagsrud, M. Næss, H. K. Alstad, A. Mortensen, R. Langseth, S. Ljødal, O. Landsverk, C. Griwodz, P. Halvorsen, M. Stenhaug, and D. Johansen. Bagadus: An integrated real-time system for soccer analytics. *ACM Trans. Multimedia Comput. Commun. Appl. (TOMCCAP)*, 10(1s):14:1–14:21, Jan. 2014.

[15] X. Sun, J. Foote, D. Kimber, and B. Manjunath. Region of interest extraction and virtual camera control based on panoramic video capturing. *IEEE Transactions on Multimedia*, 7(5):981–990, 2005.

[16] M. Tennøe, E. Helgedagsrud, M. Næss, H. K. Alstad, H. K. Stensland, V. R. Gaddam, D. Johansen, C. Griwodz, and P. Halvorsen. Efficient implementation and processing of a real-time panorama video pipeline. In *Proc. of IEEE ISM*, Dec. 2013.

[17] J. Wang, C. Xu, E. Chng, K. Wah, and Q. Tian. Automatic replay generation for soccer video broadcasting. In *Proc. of ACM MM*, pages 32–39, 2004.

[18] M. Wieland, R. Steinmetz, and P. Sander. Remote camera control in a distributed multimedia system. In B. Wolfinger, editor, *Innovationen bei Rechen- und Kommunikationssystemen*, Informatik aktuell, pages 174–181. Springer Berlin Heidelberg, 1994.

Appendix J

[Demo] Automatic Real-Time Zooming and Panning on Salient Objects from a Panoramic Video

[Authors:] **V. R. Gaddam**, R. Langseth, H. K. Stensland, C. Griwodz, and P. Halvorsen
[Published:] ACM International Conference on Multimedia (MM), 2014.

Automatic Real-Time Zooming and Panning on Salient Objects from a Panoramic Video

Vamsidhar Reddy Gaddam, Ragnar Langseth, Håkon Kvale Stensland,
Carsten Griwodz, Pål Halvorsen, Øystein Landsverk

Simula Research Laboratory & University of Oslo, Norway
{vamsidhg, ragnarla, haakonks, griff, paalh, oystesla}@ifi.uio.no

ABSTRACT

The proposed demo shows how our system automatically zooms and pans into tracked objects in panorama videos. At the conference site, we will set up a two-camera version of the system, generating live panorama videos, where the system zooms and pans tracking people using colored hats. Additionally, using a stored soccer game video from a five 2K camera setup at Alfheim stadium in Tromsø from the European league game between Tromsø IL and Tottenham Hotspurs, the system automatically follows the ball.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Video; I.4.9 [**Applications**]: Video

General Terms

Experimentation; performance

Keywords

Demonstration; panorama video; zoom; panning; real-time; tracking

1. INTRODUCTION

In several fields, especially in surveillance and sports, Pan-Tilt-Zoom (PTZ) cameras have gained popularity with their ability to use the camera sensor efficiently. Yet, one problem that a physically moving system inherently has is that it does not capture and store the data that is not currently in it's field of view. Here, we present a system that generates panorama video in real-time, and we introduce a *virtual* PTZ camera system extracting data from the panorama that is capable of following targets in real-time.

Such a system can be useful when multiple PTZ camera views are demanded for multiple targets. In order to keep the costs low, it is useful to use the same camera setup to create multiple camera views. Several panoramic capture systems have been demonstrated before with different shortcomings like ease of mobility [2], cost [5] and manual input [1].

MM'14, November 3–7, 2014, Orlando, Florida, USA.
ACM 978-1-4503-3063-3/14/11.
http://dx.doi.org/10.1145/2647868.2654882

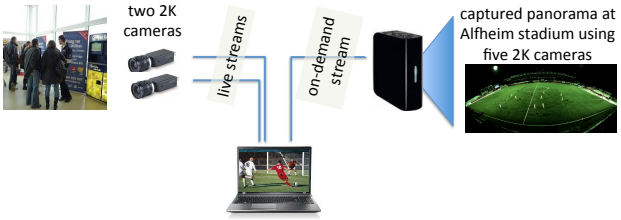


Figure 1: Demonstration setup

In this demonstration, we present two versions of our system. The first demonstration is a live, scaled down version of our sport stadium system using two 2K cameras zooming into live recorded video at the conference venue. A few funny colored hats are provided to users that act like targets. Then, the system will be able to track and zoom into the desired target automatically in real-time. The second is a demonstration that uses a recorded five 2K-camera panorama video from Alfheim stadium where our current prototype is installed. The demo will allow users to interact with the virtual camera in both stored and live scenes.

2. SYSTEM OVERVIEW

The system presented in this demo is part of a larger system where the current prototype is running at Alfheim stadium in Tromsø. This system has been presented before [4, 3], but without the possibility to automatically zoom and pan into a high-resolution panorama tracking selected objects.

The relevant part of the system in this context is divided into a panorama generation part, the object-tracking part and a video delivery part. The delivery side of the system supports both user-controlled interactive and automatically tracking virtual cameras. A sketch of our system is given in figure 1, and in the subsections below, we give more details of the different components of our system.

2.1 Real-time Panorama Video Capture

To generate a live panorama video, we use two 2K Basler industry vision cameras each of resolution 2046 × 1086 pix-

els along with 8 mm lenses. The cameras are shutter synchronized using our custom trigger solution. A recording machine grabs the frames, aligns them and stitches into a cylindrical panorama. The system automatically avoids effects due to parallax by finding dynamic seams that do not pass through moving objects.

Furthermore, to capture the entire soccer field, we use the same Basler cameras. To maximize the panorama resolution, the cameras are rotated by 90°, giving a panorama video of 4450x2000. Moreover, the cameras are mounted in a circular pattern. This means that they are pitched, yawed and rolled to look directly through a point 5 cm in front of the lenses reducing the parallax effects, and the capture system also dynamically determines the required exposure depending on changing light conditions. After the generation of the panorama video, it is encoded and compressed using x264. At the moment, with a focus on video quality, the system still demands a large network bandwidth to the full resolution panorama video. However, there are large potentials for trading of quality for lower bandwidth requirements.

2.2 Live object tracking

In the current setup, the object tracking module is a stand-alone program to support later improvements like distributed processing and multi-sensor fusion. The current tracker is a relatively simple tracking by detection. The major requirement is to be able to track objects introduced or disappeared during the process.

First, an adaptive background subtraction is performed, where the background is updated at regular intervals. Then, color-thresholding followed by object detection leads to a position of objects of different color. Once the positions for a video segment are succesfuly found, the position data is made available to the viewer which is described next. Even though this process currently shares the same resources as the stitching process, tracking is acheived with an average execution time of 7 ms per frame.

2.3 Live Automatic/Manual virtual viewer

Once the panoramic videos are encoded, they are made available to the client program running the virtual camera via HTTP segment streaming along with the position data if available. The client has two modes, an automatic mode and a manual mode. The manual mode allows the user to interact with the virtual camera, where one can pan/tilt/zoom manually into the live stream.

More importantly in this demo, the automatic mode follows a few heuristics to keep the demanded target in the virtual view yet provide a smooth video. The zoom for the on-site demo is set to be fixed because of the unknown 3D structure of the area, whereas in the soccer field the zoom variable also changes depending on the ball position in the field.

In addition to the virtual view, a preview window is presented where the portion of panorama that is being fetched, corrected for a perspective view and displayed is highlighted. This preview window proves to be a rather useful feature when developing the servoing algorithms. In addition, it also demonstrates that the virtual view is not a simple crop from the panorama video.

All the processing except for downloading and decoding the video frame happens on a GPU to utilize the parallel processing power. The client has been tested on both Mac and PC with different capabilities. Nevertheless the average processing time is approximately 12 ms per frame on commodity graphics hardware.

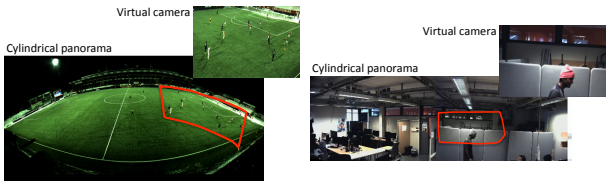


Figure 2: The virtual camera is generated from the region of interest marked in the panorama video. Note that it is not a simple crop from the bigger video.

3. DEMONSTRATION

In this demo, we present a system for real-time interactive zooming and panning of panorama video. The general setup is shown in figure 1. We use two types of panorama input. We use a stored panorama video captured in the Alfheim soccer stadium, i.e., the European League game between Tromsø IL and Tottenham Hotspurs. The video was recorded using five 2K industrial cameras, and processed and stitched in a distributed system. We also use a live system using two 2K cameras on site to generate the panorama video. Both the stored-video and the live-video demos work in real-time where the system zooms and pans into tracked objects, i.e., the ball in the soccer game and people wearing colored hats in the live panorame video[1].

Acknowledgments

This work has been performed in the context of the *iAD* center for Research-based Innovation (project number 174867) funded by the Norwegian Research Council.

4. REFERENCES

[1] P. Carr and R. Hartley. Portable multi-megapixel camera with real-time recording and playback. In *Proc. of DICTA*, pages 74–80, 2009.

[2] P. Carr, M. Mistry, and I. Matthews. Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording. In *Proc. of ACM MM*, pages 193–202, 2013.

[3] V. R. Gaddam, R. Langseth, S. Ljødal, P. Gurdjos, V. Charvillat, C. Griwodz, and P. Halvorsen. Interactive zoom and panning from live panoramic video. In *Proc. of ACM NOSSDAV*, Mar. 2014.

[4] P. Halvorsen, S. Sægrov, A. Mortensen, D. K. Kristensen, A. Eichhorn, M. Stenhaug, S. Dahl, H. K. Stensland, V. R. Gaddam, C. Griwodz, and D. Johansen. Bagadus: An integrated system for arena sports analytics – a soccer case study. In *Proc. of ACM MMSys*, pages 48–59, Mar. 2013.

[5] O. Schreer, I. Feldmann, C. Weissig, P. Kauff, and R. Schafer. Ultrahigh-resolution panoramic imaging for format-agnostic video production. *Proceedings of the IEEE*, 101(1):99–114, Jan 2013.

[1] http://home.ifi.uio.no/vamsidhg/acmdemo.mp4

Appendix K

[Demo] Scaling Virtual Camera Services to a Large Number of Users

[Authors:] **V. R. Gaddam**, R. Langseth, H. K. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen
[Published:] ACM International Conference on Multimedia Systems (MMSys), 2015.

Scaling Virtual Camera Services to a Large Number of Users

Vamsidhar Reddy Gaddam[1,*], Ragnar Langseth[1], Håkon Kvale Stensland[1], Carsten Griwodz[1], Dag Johansen[2], Pål Halvorsen[1]

[1]Simula Research Laboratory & University of Oslo, Norway
[2]University of Tromsø, Norway

[*]vamsidhg@ifi.uio.no

ABSTRACT

By processing video footage from a camera array, one can easily make wide-field-of-view panorama videos. From the single panorama video, one can further generate multiple virtual cameras supporting personalized views to a large number of users based on only the few physical cameras in the array. However, giving personalized services to large numbers of users potentially introduces both bandwidth and processing bottlenecks, depending on where the virtual camera is processed.

In this demonstration, we present a system that address the large cost of transmitting entire panorama video to the end-user where the user creates the virtual views on the client device. Our approach is to divide the panorama into tiles, each encoded in multiple qualities. Then, the panorama video tiles are retrieved by the client in a quality (and thus bit rate) depending on where the virtual camera is pointing, i.e., the video quality of the tile changes dynamically according to the user interaction. Our initial experiments indicate that there is a large potential of saving bandwidth on the cost of trading quality of in areas of the panorama frame not used for the extraction of the virtual view.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Video; I.4.9 [**Applications**]: Video

General Terms

Experimentation; measurement; performance

Keywords

Interactive immersion; panorama video; zoom, panning; real-time; virtual camera, video streaming

ACM MMSys'15, Mar 18-20, 2015, Portland, OR, USA
ACM 978-1-4503-3351-1/15/03
http://dx.doi.org/10.1145/2713168.2713189.

1. INTRODUCTION

There exist many types of panorama solutions where high-resolution, wide field-of-view video is captured and streamed in real-time. For example, in arena sports like soccer, American football and ice-hockey, many game analysis systems provide camera arrays where individual camera images are stitched together to cover the entire field. Then, to focus on parts of the area, it is often desirable to zoom and pan into the generated video. Figure 1 demonstrates an example output of such a system. In this case, a virtual camera is generated by extracting pixels from parts of the stitched panorama video allowing individual users to interactively control an own personalized view. However, these types of systems also give interesting opportunities for innovation in broadcasting scenarios where large number of fans and supporters would like to generate their own camera view of the event.

Figure 1: The panorama video with the marked region of interest is shown together with the generated virtual camera, emphasizing that the extracted area is not a simple crop from the high-resolution panorama video. It is generated by a perspective reprojection and hence we cannot achieve it by a simple rectangular cropping.

We have earlier described the Bagadus system [8, 15] generating panorama videos of a soccer stadium in real-time. Additionally, we have presented how individual users can be his own cameraman [5, 6] by extracting a zoomed and panned video from the panorama video, for example following a particular player. Here, the system streamed the cylindrical panorama video to the remote clients which extracted a perspective-corrected camera view on the client side. However, popular soccer games often attract millions of concurrent users. For example, during the 2014 FIFA

World Cup in Brazil, the web player and app had during the 56 games streamed video to about 24 million unique users [4]. If we additionally take into account the potential number of TV viewers, e.g., 909.6 million television viewers tuned in to at least one minute of the 2010 FIFA World Cup final at home [11], we definitely have a large challenge in providing real-time virtual camera services in such a scale using our previous approach of sending the entire, full-quality panorama to every user.

There are generally two approaches to manage the virtual camera. The first as we have presented earlier where the entire panorama is sent over the network and the virtual view is extracted at the client side. The second alternative performs the generation of the view from the panorama on the server side sending only the virtual view video over the network. Thus, the trade-off on the server-side is between processing and outgoing bandwidth, and vice-verse on the client side.

In general, the de facto streaming approach using segmented, adaptive HTTP streaming has proven to scale well. We have therefore adopted this solution in our system, and in this demonstration, we present a system where the panorama video is divided into tiles, i.e., each encoded in multiple qualities (and thus bit rate). Then, the panorama video tiles are retrieved by the client in a quality depending on the current position and the behaviour of the virtual camera, i.e., the video quality of the tile changes dynamically according to the user interaction, and the panorama is restored on the client side with different qualities in different areas of a frame.

Our initial experiments indicate that there is a large potential for saving bandwidth on the cost of trading quality in areas of the panorama frame not used for the extraction of the virtual view. The proposed demonstration therefore shows how the client performs and how the quality of the extracted view and the panorama video changes when the virtual camera moves to another region of interest.

2. THE COSTS OF VIRTUAL VIEWS

We have earlier presented our approach for generating the high resolution cylindrical panorama videos in real-time [8]. We have also demonstrated that these video can be used to generate individual personalized *virtual views* of the game [5]. When it comes to delivering video to the client, we have explored two possibilities with respect to creating virtual views.

Our initial approach is to transfer the entire panoramic video and generate the virtual views on the client. This gives cheap processing requirements on the server-side at the cost of very high bandwidth requirements. In our example system installed at Alfheim stadium, the average size of each 25-fps *3-second* segment of the the 4096×1680 panorama video is approximately 2.1 MB[1], i.e., the bandwidth requirement for each client becomes about 5.7 Mbps merely for the transfer of the panorama video, and in future systems, a much higher resolution panorama is desirable. Then, after the panorama is successfully transferred, the client needs to process it so that a virtual view can be extracted. Earlier we demonstrated that this can be accomplished in real-time on commodity graphics hardware [5], and figure 2 demonstrates the performance as a function of output resolution. These values are computed for extraction of virtual view on a GPU. Thus, the bandwidth requirement is quite high, but processing wise, the client devices manage the load.

[1]This number depends on the lighting and weather conditions, but the given number works well as an example.

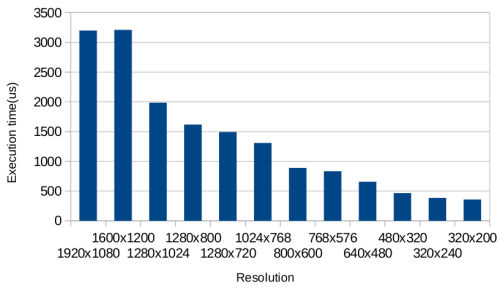


Figure 2: Execution times for various sizes of virtual camera on GTX 460

An alternative approach is to generate the virtual views on the server and only stream the generated virtual view video to the client. Thus, in this approach, the reprojection is performed on the server side. This approach requires nothing more than a browser that is capable to play a video on the client device, i.e., it severely reduces the computational load and the bandwidth requirements on the clients. However, the processing costs on the server-side are huge, and it quickly becomes a large bottleneck as not only must we generate the virtual view, but we must also encode the video for compression. We have made a few experiments using the second generation hardware from Nvidia [14]. Our experiments show that the GeForce GTX 750 Ti GPU can encode 16 full HD video streams at 30 frames per seconds [14]. Experiments showed that this was the limiting factor in how many unique views we could create in real-time. This implies that if we want to provide a service to say 100,000 concurrent users, we would require a cluster totaling to about 6,250 GPU devices. Such an initial installation costs at the time of writing about 937,500 USD merely for the GPUs.

Owing to the challenges mentioned above, no straight forward solution is going to work well for scaling our system to large numbers of concurrent users. However, as the HTTP streaming solutions have proved to scale well from a sending-side point of view using for example CDNs, we have looked at solutions for the first approach – client side generated virtual views.

3. SYSTEM OVERVIEW

Based on the decision in the previous section, the challenge is to reduce the cost of streaming a complete panorama video to every users. In this respect, researchers in the multimedia community have for some time analyzed region-of-interest streaming solutions. For example, tiling is discussed in [3, 2, 7, 12, 13, 16, 18]. Furthermore, [1, 10, 17, 19] extensively address the problem of automatically generating personalized content, and [9] discusses plain cropping. However, our target is a large scale solution scaling the delivery using modern HTTP streaming where each user independently interacts with the system to have a personalized view using zoom, pan and tilt, i.e., the entire panorama must be retrieved and the quality of the tiles are based on the per user interaction.

Similar to many other approaches, our solution is based on dividing the panorama into tiles as shown in figure 3, each encoded as an adaptive HTTP stream using for example HLS. A client retrieves segments in as high quality as possible for segments being used for the virtual camera, and the rest of the tiles are retrieved in decreasing lower quality depending on the distance to the edge of the virtual camera image. In contrast to for example [16] retrieving only tiles in the region of interest, we need to retrieve all tiles since the virtual camera moves and at least low quality data needs to be available if the user zooms out or moves quickly. Another difference is that the tiles fetched do not follow a strict logic apart from being in the neighborhood of the current tile. In [18], for instance, all the tiles are being fetched, but the reduction in quality is reflected by salience. Moreover, the non-linear nature of a panorama-virtual view transformation introduces further complexities in the approach. For example, in figure 3 it can be seen that the collection of required tiles do not form any simple shape like a rectangle or a square, e.g., as used in [9]. This poses different challenges than the ones that are being tackled in for example [12] where the panning and tilting corresponds to strictly navigating the panorama along the horizontal and vertical directions respectively. Such an approach adds complexity on the tile retrieval strategy as the quality adaption strategy not only must take into account available network bandwidth and client device resources (as usually done for *one* stream), but it must also coordinate the tile qualities according to the dynamic position of the virtual camera.

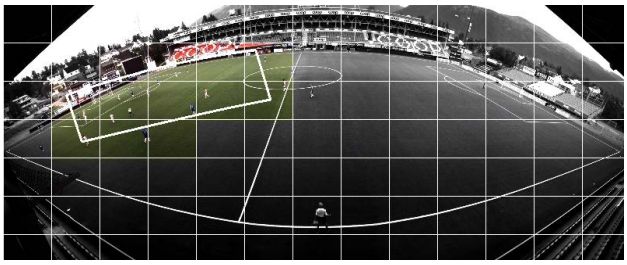


Figure 3: Example of panorama tiling (320x256px)

4. EVALUATION

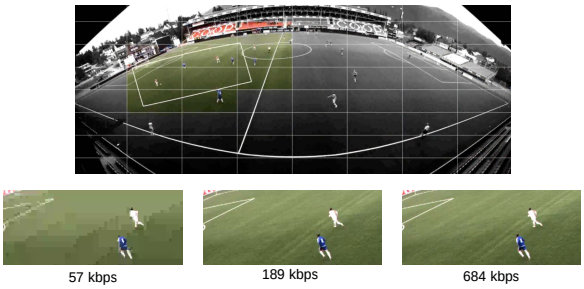


Figure 4: Tiles in different quality

Above, we said that the full quality panorama video required about 5.6 Mbps. If we divide the panorama in 8×8 tiles as shown in figure 4 with the given tile qualities (and bitrates), a complete full-quality panorama requires 8.6 Mbps due to the loss of compression across tile boundaries. However, if the user zooms as shown in the figure requiring only full quality for 10 of the tiles (the colored tiles used for the virtual view), the respective bandwidth requirements of the panorama decreases to 3.2 and 2.0 Mbps when using the middle and low quality for rest of the tiles (gray).

Figure 5 shows an example of how our virtual viewer works. It can be observed that the virtual view presents no loss in quality. However, the parts of panorama that are not being shown in the virtual view are fetched in the lowest possible quality. This phenomenon is quite evident in the preview image.

Figure 5: An example of a multi quality tiled-panorama and the virtual view that is extracted from it. The quality of the panorama is quite poor in the areas that are not being shown to the user. However if the user decides to pan quickly, she still gets a reliable low quality video instead of a black patch.

Such a system comes with a rather interesting trade-off with respect to the segment size. A segment of *3 seconds* compared to 3 segments of *1 second* each has a certain advantage in the encoded file-size due to the reduction in the number of I-frames. However, the segment size also determines how quickly a tile can change it's quality. Our initial experiments showed that this trade-off is an interesting one to study. Since the virtual camera moves, it is hard to see the differences of lower quality tiles if the levels are not too far apart. However, the user interaction with in a 3-second period can be assumed completely random and it cannot benefit from the predictive model as much as a 1-second segment could.

5. DEMONSTRATION

In this demo[2], we present a system for real-time interactive zooming and panning of panorama video using video from real-world installations in two Norwegian soccer stadiums. We show how the quality changes of different parts of the panorama video when moving the virtual camera.

[2]http://home.ifi.uio.no/vamsidhg/mmsysDemo

We also show that if there are large differences between the quality layers, reduced quality is noticeable when quickly moving the virtual camera, but if the layers are carefully selected, but still saving bandwidth, it might be hard to see the quality differences due to the view movement. Thus, the tiling approach has potential to greatly reduce the required bandwidth of a scenario where every user is his or her own cameraman [6].

Acknowledgments

This work has been performed in the context of the *iAD* center for Research-based Innovation (project number 174867) funded by the Norwegian Research Council.

6. REFERENCES

[1] N. Babaguchi, Y. Kawai, and T. Kitahashi. Generation of personalized abstract of sports video. In *Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on*, pages 619–622, Aug 2001.

[2] F. Chen and C. De Vleeschouwer. Personalized production of basketball videos from multi-sensored data under limited display resolution. *Comput. Vis. Image Underst.*, 114(6):667–680, June 2010.

[3] E. Foote, P. Carr, P. Lucey, Y. Sheikh, and I. Matthews. One-man-band: A touch screen interface for producing live multi-camera sports broadcasts. In *Proc. of ACM MM*, pages 163–172, 2013.

[4] Fédération Internationale de Football Association. 2014 FIFA World Cup breaks online streaming records. http://www.fifa.com/aboutfifa/organisation/-news/newsid=2401405/, 2014.

[5] V. R. Gaddam, R. Langseth, S. Ljødal, P. Gurdjos, V. Charvillat, C. Griwodz, and P. Halvorsen. Interactive zoom and panning from live panoramic video. In *Proc. of ACM NOSSDAV*, pages 19:19–19:24, 2014.

[6] V. R. Gaddam, R. Langseth, H. K. Stensland, P. Gurdjos, V. Charvillat, C. Griwodz, D. Johansen, and P. Halvorsen. Be your own cameraman: Real-time support for zooming and panning into stored and live panoramic video. In *Proc. of ACM MMSys*, pages 168–171, 2014.

[7] R. Guntur and W. T. Ooi. On tile assignment for region-of-interest video streaming in a wireless LAN. In *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video - NOSSDAV '12*, page 59, New York, New York, USA, 2012. ACM Press.

[8] P. Halvorsen, S. Sægrov, A. Mortensen, D. K. Kristensen, A. Eichhorn, M. Stenhaug, S. Dahl, H. K. Stensland, V. R. Gaddam, C. Griwodz, and D. Johansen. Bagadus: An integrated system for arena sports analytics – a soccer case study. In *Proc. of ACM MMSys*, pages 48–59, Mar. 2013.

[9] R. Heck, M. Wallick, and M. Gleicher. Virtual videography. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 3(1):4–es, Feb. 2007.

[10] R. Kaiser, M. Thaler, A. Kriechbaum, H. Fassold, W. Bailer, and J. Rosner. Real-time person tracking in high-resolution panoramic video for automated broadcast production. In *Proc. of CVMP*, pages 21–29, 2011.

[11] KantarSport. 2010 FIFA World Cup South Africa - Television Audience Report. http://www.fifa.com/mm/document/affederation/tv/-01/47/32/73/2010fifaworldcupsouthafrica-tvaudiencereport.pdf, 2010.

[12] A. Mavlankar and B. Girod. Video streaming with interactive pan/tilt/zoom. In M. Mrak, M. Grgic, and M. Kunt, editors, *High-Quality Visual Experience*, Signals and Communication Technology, pages 431–455. 2010.

[13] K. Q. M. Ngo, R. Guntur, and W. T. Ooi. Adaptive encoding of zoomable video streams based on user access pattern. In *Proceedings of the second annual ACM conference on Multimedia systems - MMSys '11*, page 211, New York, New York, USA, 2011. ACM Press.

[14] NVIDIA. NVIDIA - NVIDIA hardware video encoder. `http://developer.download.nvidia.com/compute/nvenc/v4.0/NVENC_AppNote.pdf`, 2014.

[15] S. Sægrov, A. Eichhorn, J. Emerslund, H. K. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen. Bagadus: An integrated system for soccer analysis (demo). In *Proc. of ICDSC*, Oct. 2012.

[16] A. Shafiei, Q. M. K. Ngo, R. Guntur, M. K. Saini, C. Pang, and W. T. Ooi. Jiku live. In *Proceedings of the 20th ACM international conference on Multimedia - MM '12*, page 1265, New York, New York, USA, 2012. ACM Press.

[17] X. Sun, J. Foote, D. Kimber, and B. Manjunath. Region of interest extraction and virtual camera control based on panoramic video capturing. *IEEE Transactions on Multimedia*, 7(5):981–990, 2005.

[18] H. Wang, V.-T. Nguyen, W. T. Ooi, and M. C. Chan. Mixing tile resolutions in tiled video: A perceptual quality assessment. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*, NOSSDAV '14, pages 25:25–25:30, New York, NY, USA, 2013. ACM.

[19] R. Xu, J. Jin, and J. Allen. Framework for script based virtual directing and multimedia authoring in live video streaming. In *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International*, pages 427–432, Jan 2005.