Bagadus: An Integrated System for Arena Sports Analytics – A Soccer Case Study –

Pål Halvorsen[1], Simen Sægrov[1], Asgeir Mortensen[1], David K. C. Kristensen[1], Alexander Eichhorn[1], Magnus Stenhaug[2], Stian Dahl[3], Håkon Kvale Stensland[1], Vamsidhar Reddy Gaddam[1], Carsten Griwodz[1], Dag Johansen[2]

[1]University of Oslo/Simula Research Laboratory [2]University of Tromsø [3]ZXY Sport Tracking

ABSTRACT

Sports analytics is a growing area of interest, both from a computer system view to manage the technical challenges and from a sport performance view to aid the development of athletes. In this paper, we present Bagadus, a 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 paper, we describe how the system can follow and zoom in on particular player(s). Next, the system will playout events from the games using stitched panorama video or camera switching mode and create video summaries based on queries to the sensor system. Furthermore, we evaluate the system from a systems point of view, benchmarking different approaches, algorithms and trade-offs.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Video

General Terms

Design, Experimentation

Keywords

System integration, camera array, sensor tracking, video annotation, sport analytics, soccer system

1. INTRODUCTION

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. In the area of soccer, several systems enable trainers and coaches to analyze the game play in order to improve the performance. For instance, Interplay-sports [3] has been used since 1994. Here, video-streams are manually analyzed and annotated using a soccer ontology classification scheme. Trained and soccer-skilled operators tag who has the ball, who made a pass, etc. ProZone [4] is another commonly used system that automates some of this manual annotation process by video-analysis software. In particular, it quantifies movement patterns and characteristics like speed, velocity and position of the athletes. In this respect, Di Salvo et al. [25] conducted an empirical evaluation of deployed ProZone systems at Old Trafford in Manchester and Reebook Stadium in Bolton, and concluded that the video camera deployment gives an accurate and valid motion analysis. Similarly, STATS SportVU Tracking Technology [5] 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 [1] 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 [28] and ZXY Sport Tracking [7] systems use global positioning and radio based systems for capturing performance measurements of athletes. For example, using a sensor system, ZXY captures a player's orientation on the field, position, step frequency and heart rate frequency with a resolution of samples up to 20 times per second. Then, these systems can present their statistics, including speed profiles, accumulated distances, fatigue, fitness graphs and coverage maps, in many different ways like 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, e.g., events tagged by coaches or other human expert annotators must be manually extracted from (or marked in) the videos. Furthermore, connecting the player statistics to the video requires manual work. One recent example is the Muihtu system [19], which integrates coach annotations with related video sequences, but the video must be manually transferred and mapped to the game timeline.

As the above examples show, there exist several tools for soccer analysis, i.e., both with respect to for example performance (real-time), algorithms and features. However, to the best of our knowledge, there does not exist a system that fully integrates all these features. In this paper, we therefore present Bagadus [24], which 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

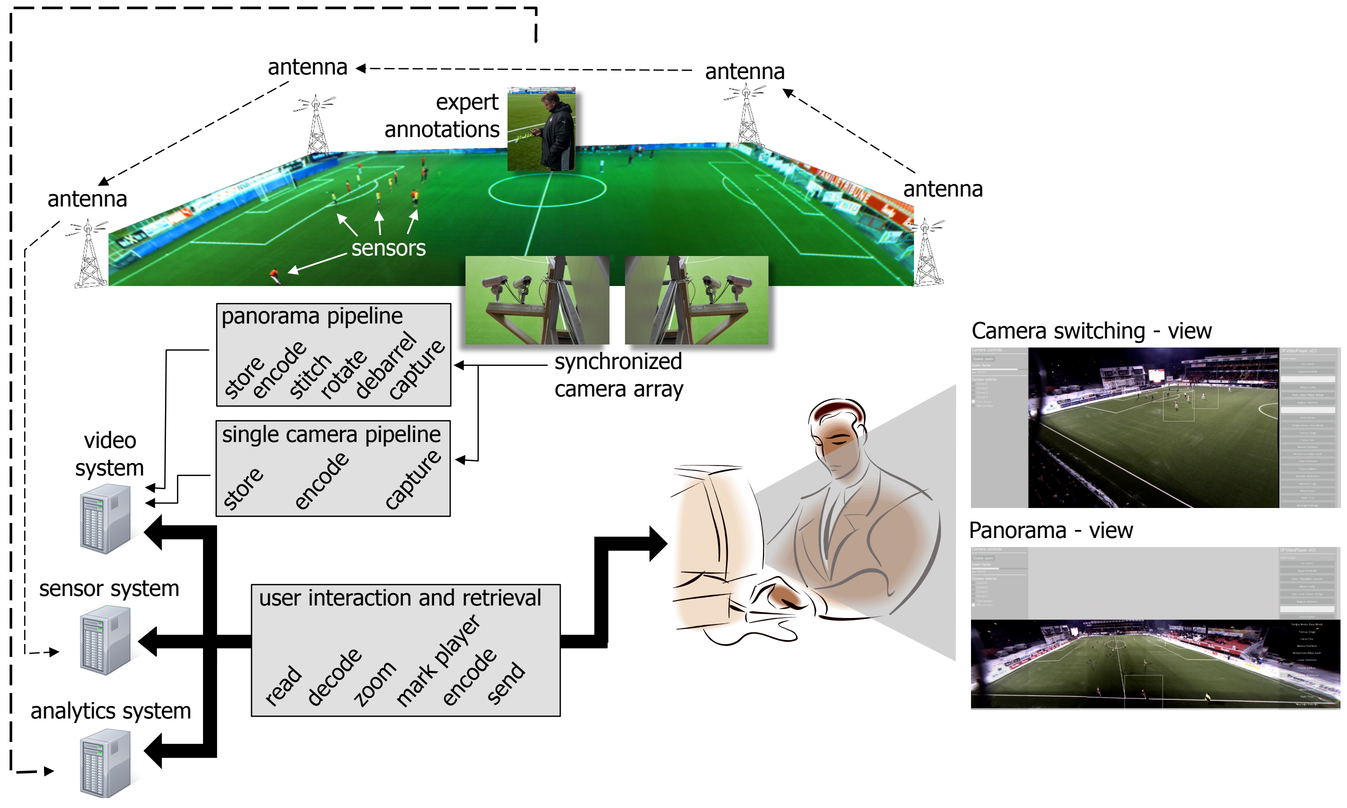


Figure 1: Overall Bagadus architecture.

playout 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 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. Furthermore, we also evaluate the system from a systems point of view; benchmarking different approaches, algorithms and tradeoffs. In summary, our so far unoptimized prototype supports all operations in real-time, except stitching of 4 HD cameras. However, as we discuss in the paper, a large potential for improved speed lies in more efficient algorithms, parallel processing and offloading to co-processing units like GPUs.

The remainder of the paper 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 the paper in section 8.

2. Bagadus – THE BASIC IDEA

The interest in sports analysis systems has recently increased a lot, and as described above, several systems exist, some for a long time already. However, they still require a large portion of manual work to integrate information from various computer systems and expert sport analytics. In this respect, **Bagadus**[1] 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 modes. The first allows playback of video that switches between streams delivered from the different the cameras, either manually selecting a camera or automatically following players based on sensor information. The second mode 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 both these modes, Bagadus allows a user to zoom in on and mark player(s) in the retrieved video (see figure 1).

To identify and follow players on the field, we use a *tracking* (sensor) subsystem. In this respect, tracking people

[1]The Sami word Bagadus translates to instruction, supervision or guidance.

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 [7] 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.

The Bagadus application implements and integrates many well-known components to support a particular application scenario, i.e., arena sports analytics. However, the combination of different technologies raises requirements on both spatial and temporal synchronization of multiple signals from independent sources. 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 to record and synchronize 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, 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 playout the events annotated by expert users. Thus, where people earlier used a huge amount of time for analyzing the game manually, Bagadus is an integrated system where the required operations and the synchronization with video is automatically managed.

3. VIDEO SUBSYSTEM

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 and delivers video events based on given time-intervals, player positions, etc. The video subsystem supports two different playback modes. The first allows playing video from individual cameras where the view switches automatically between the cameras, i.e., manually selecting a camera or automatically following players. For this mode, the video data from each camera is stored and encoded separately. The second mode plays back a panorama video stitched from the 4 camera feeds. The cameras are calibrated in their fixed position, and the captured videos are each processed in a capture-debarrel-rotate-stitch-encode-store pipeline.

3.1 Camera setup

To record high resolution video of the entire soccer field, we have installed a camera array consisting of 4 Basler industry cameras with a 1/3-inch image sensor supporting 30 fps 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, i.e., 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, i.e., all four cover the full field with sufficient overlap to identify common features necessary for camera calibration and stitching (see figure 2).

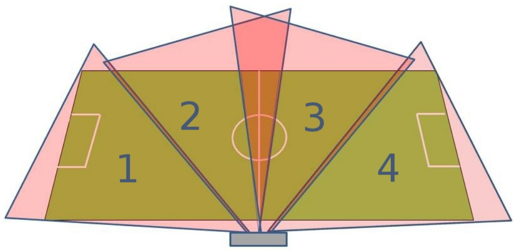


Figure 2: Camera setup at Alfheim stadium.

The cameras are managed using our own library, called Northlight, to manage frame synchronization, storage, encoding, etc. The system is currently running on computers with an Intel Core i7-2600 @ 3.4 GHz and 8 GB memory. Northlight integrates the SDK provided by Basler for the cameras, video encoding using x264, color-space conversion using FFmpeg, JPEG-compression using Open-JPEG and computer vision algorithms using OpenCV. Additionally, a time synchronization protocol is currently under development to easier give the stored frames the same time stamp between multiple machines.

3.2 Digital zoom

Bagadus supports digital zooming on tracked players, i.e., keeping the tracked player in the center of the image while zooming in. In this respect, an important operation is image interpolation which uses known data to estimate values at unknown points. In our context where we crop the image, it is for example required to re-size or remap (distort) the image from one pixel grid to another, i.e., to increase or decrease the total number of pixels.

As for the rest of our processing pipeline, we need a real-time operation, and we have compared 4 different interpolation algorithms, listed with increasing complexity: 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. In contrast to bilinear interpolation, which only takes 4 pixels (2x2) into account, bicubic interpolation considers 16 pixels (4x4). Images re-

Algorithm	Min	Max	Mean
Nearest neighbor	4.1	5.1	4.2
Bilinear	7.4	7.8	7.4
Bicubic	47.9	55.3	48.3
Lanczos	240.1	245.4	240.7

Table 1: A speed comparison (ms per frame) between different interpolation algorithms in OpenCV when remapping 300 frames.

sampled with bicubic interpolation are smoother and have fewer interpolation artifacts. Lanczos interpolation has the advantages of bicubic interpolation and is known to produce sharper results than bicubic interpolation. However, we need to process the frames in real-time, and in this respect, only the nearest neighbor and bilinear solutions can be used in order to achieve the target frame rate (see table 1). Our demo, however, is offline so here we have used the Lanczos interpolation method (see also discussion below comparing two different stitching pipelines).

3.3 Stitching

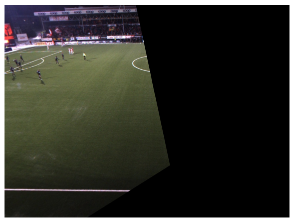
Tracking game events over multiple cameras is a nice feature, but in may situations, it may be desirable to have a complete view of the field. In addition to the camera selection functionality, we therefore generate a panorama picture by combining images from multiple, trigger-synchronized cameras.

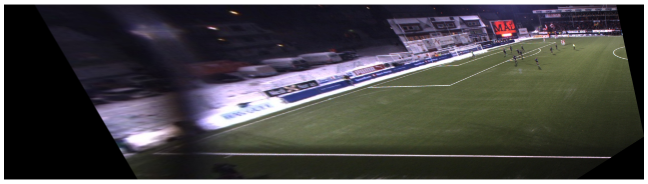
As seen in figure 1, the cameras are calibrated in their fixed position using a classical chessboard pattern [32], and the stitching operation requires a more complex processing pipeline. We have alternative implementations with respect to what is stored and processed offline, but in general, we must 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. A simple illustration is given in figure 3 where the processing of a single image is shown and how the overlapping areas are used to generate a panorama image. For the first steps, like debarreling, we again move pixels according to a mathematical model, i.e., resulting in an interpolation overhead given in table 1. The distortion parameters for debarreling are computed during the calibration phase.

After the initial steps, the overlapping areas between the frames are used to stitch the 4 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 et al. [10], i.e., we used the auto-stitcher functions using planar-, cylindrical- and spherical projections. Examples of the respective panorama images are given in figures 4(a), 4(b) and 4(c), respectively, and the resulting panorama image resolution[2] and per image execution time are given in table 2. As we can see from the figures and the table, neither of the OpenCV implementations are perfect.

[2]Note that many of the panorama resolutions are larger than what the 4 cameras deliver individually due to the warping, changing perspective, rotation and stretching of the original images.

(a) Debarreled left image.


(b) Warped left image.


(c) Debarreled, warped and padded left image.

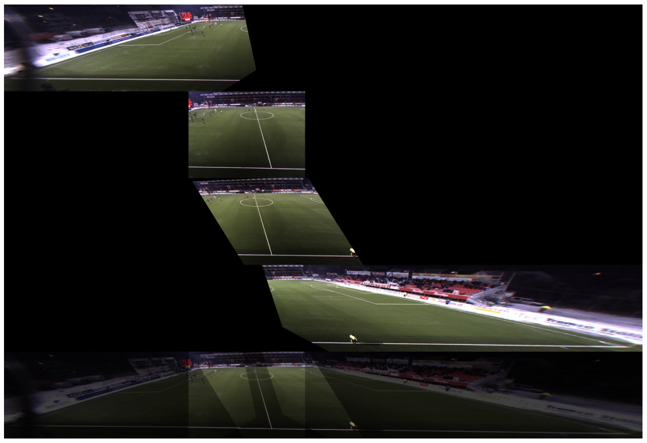

(d) Each of the 4 cameras, warped and padded to a horizontal line (the view of the second camera). It shows these 4 view superimposed on each other. The highlighted areas show where the views overlap.

Figure 3: Processing the images for stitching.

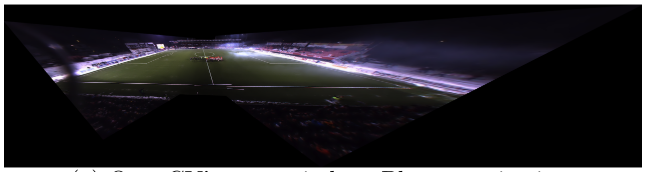
Algorithm	Image resolution	Execution time (ms per image)
OpenCV Planar projection	12040x3051	14103
OpenCV Cylindrical projection	9275x1803	4900
OpenCV Spherical projection	2371x1803	1746
Homography stitch	7000x960	974

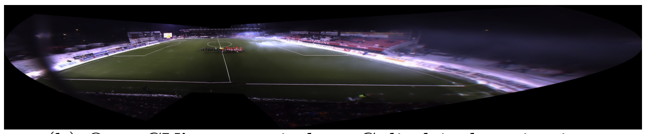
Table 2: Stitching characteristics.

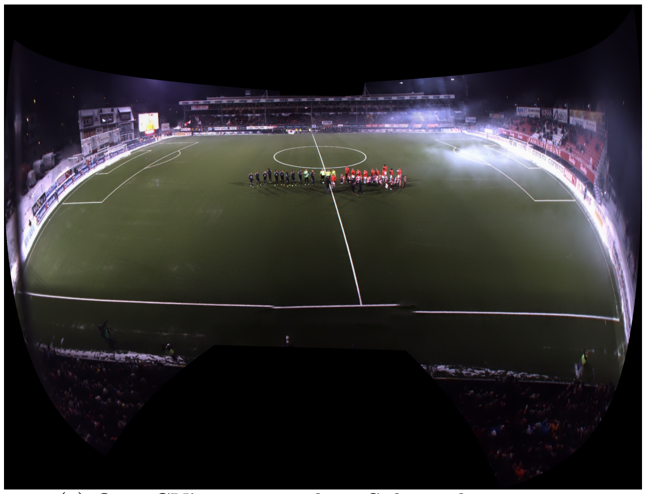
They all have an execution time longer than the deadline for real-time stitching, and the fastest (spherical) still has severe barreling effects.

To see if we could find a better and faster way to stitch the images, we investigated if we could use a homography given by the projective geometry translating ZXY's coordinate system to pixel coordinates. This approach includes a more manual (but the camera mounting is static) image registration step, using the properties of the pinhole camera model and the homography relationship between planes.

The first step is to find corresponding pixel points in order to compute the homography between the two planes [14].


(a) OpenCV's auto-stitcher: Planar projection.


(b) OpenCV's auto-stitcher: Cylindrical projection.


(c) OpenCV's auto-stitcher: Spherical projection.

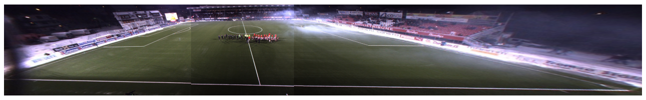

(d) Homography stitch.

Figure 4: Stitching comparison.

When the homography is calculated, the image can be warped 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. An example of the resulting image is found in figure 4(d). Compared to the other tested approaches, the image is at least as good, and the processing time is greatly reduced. Currently, the panorama pipeline is still non-real-time, but we are currently working on optimizing, parallelizing and offloading operations to multiple cores and GPUs.

3.4 Performance and resource consumption

Handling video is in general expensive in terms of resource consumption. The video is delivered from the cameras to the machines for storage in real-time with a negligible delay. However, there are many optimization trade-offs between the scarce resources. In this section, we try to illuminate some of these trade-offs and show the performance of the current prototype.

3.4.1 *Storage*

Four HD cameras produce a lot of data for each game, and additionally, the four camera feeds can be stitched together different ways giving potentially very large images (see table 2). In this respect, there is a tradeoff between processing and storage. Storing each video uncompressed (YUV directly from the cameras) consumes large amount of disk space (though it is cheap), and it consumes much disk- and network bandwidth, but on the other hand, the system can process the video in the stitching operations directly. To save storage space and bandwidth, we may store the videos compressed (lossless H.264) at the cost of a higher processing cost encoding the data to YUV format for pixel manipulation and frame extraction. In the current prototype, the video data is stored in segments of 90-frame group-of-pictures (GOPs), i.e., 3-second segments of video, as separate files to ease access and distribution over machines. Table 3 shows an experiment (using standard desktop SATA hard-disks) where we tested different options. The table includes statistics of the storage requirement per 90-frame GOP and the corresponding I/O requirement, the time to retrieve the first frame in the GOP from the disk so that it is ready for further processing (we do not need to wait for the entire GOP before starting to encode) and the total time to read and encode the entire GOP. The times are measured reading an entire game from an otherwise idle disk, but variations occur due to placements on the disk platters etc. Thus, the numbers are meant as indicators, not exact numbers, and better performance can also be achieved using better high-end disks.

In camera switching mode, we select frames from the four individual cameras directly without any other image processing. The first rows in table 3 show the tradeoff between YUV storage versus H.264 storage reading data from a local disk or from a remote disk over a gigabit local network. Obviously, there is a large difference between storing the data in YUV and H.264 encoding. The YUV format requires more than twice the storage and bandwidth and it has a faster startup time compared to H.264[3]. If data is stored locally, both options are able to deliver data in time for real-time playout, i.e., the 90-frame GOPs (3 seconds of video data) are read (and encoded) in less than 3 seconds. If we need to read from a remote disk, we need to store data compressed using H.264 to achieve real-time retrieval from the disk.

When the system is in panorama mode, it requires a lot more data. Using the stored stitch-images in table 3, we assume the stitching pipeline presented below is performed off-line. Taking the full stitch as an example, we save more than four times the storage space and bandwidth if we compress the data using H.264. Again, we observe that the data can be displayed earlier reading uncompressed YUV data, but in this case, we cannot read data fast enough to support real-time delivery, i.e., reading a 90-frame GOP below 3 seconds. In the case of using H.264, we can on average read data fast enough to support real-time delivery, even using the standard disks we used.

3.4.2 *Stitching Processing pipeline*

Depending on the type of operation, we have defined two different stitching processing pipelines. The first is meant for offline stitching where we store the stitched images back to disk, and the second is meant for live stitching reading directly from the camera feeds. The point is that performing

[3]H.264 can achieve a much better compression, but for the current prototype, we used lossless mode for later pixel manipulation with the goal of reading video data in real-time.

Camera mode	Data location	Storage format	Storage space (MB)		Bandwidth (Mbps)		Read and encode			
							first frame (ms)		GOP (ms)	
			mean	max	mean	max	mean	max	mean	max
Single camera	local disk	H.264	76.72	80.93	204.60	215.81	242.0	804.2	1236.4	3041.5
		YUV	165.89	—	442.37	—	24.9	348.5	1600.8	3386.0
	remote disk	H.264	76.72	80.93	204.60	215.81	376.8	993.6	1850.5	3451.7
		YUV	165.89	—	442.37	—	92.2	386.3	3235.9	5138.5
2-camera stitch	local disk	H.264	69.94	73.29	186.51	195.83	509.2	691.4	1508.7	2084.4
		YUV	279.94	—	776.50	—	35.0	396.1	2177.1	4017.8
	remote disk	H.264	69.94	73.29	186.51	195.83	560.4	970.5	1707.0	2215.7
		YUV	279.94	—	776.50	—	235.1	727.0	3701.3	3386.0
4-camera stitch	local disk	H.264	179.69	183.23	479.18	488.81	441.0	555.7	2150.2	3398.5
		YUV	910.98	—	2429.28	—	120.2	533.5	8150.9	12825.9
	remote disk	H.264	179.69	183.23	479.18	488.81	601.8	993.0	2872.4	3828.6
		YUV	910.98	—	2429.28	—	281.3	434.3	13396.4	20310.4

Table 3: Storage, I/O and processing tradeoffs per 90-frame, 3-second GOP (segment): compressed (H.264) vs. uncompressed (YUV) and local vs. remote disk performance.

these steps offline, we can focus on quality, and in this case, we have used Lanczos interpolation (see table 1) as part of the debarreling and stitching steps. For live stitching, we have used the nearest neighbor interpolation due to the speed.

Table 4 contains the processing times from the two different homography stitching pipelines over 100 stitches from the 4 cameras, i.e., using the nearest neighbor or Lanczos interpolation. The experiment uses the same stored data in both tests. This adds a disk read (not included in table 4, see table 3 for 90 frame measurements) and a disk write. The pipeline in the used system will only have a write in the offline mode (and a read later when the data is accessed), i.e., in a live scenario, there are no reads or writes to disk. Note also that our implementations of the stitching algorithms work on RGB data rather than the YUV data requiring a convertion. This is due to our initial work on OpenCV and will be removed in future improvements of the stitching algorithm.

Note that only the image manipulation steps (debarrel and stitch) differ significantly, and the others are merely variations over different runs. These steps are also where we are currently investigating alternatives as described later in section 7. The other steps are on average performed below the real-time threshold given by the 30 fps boundary. The total time is therefore just an indication of how long delay we have per image in a live stream (if we manage to get debarreling and stitching in real-time too).

3.4.3 Real-Time Capabilities

Bagadus successfully shows that an integration of subsystems is possible to make an automatic system with the required functionality. However, our benchmarking of the system shows that parts of the non-optimized pipeline are non-real-time in order to support a 30 fps video playout in panorama mode, i.e., table 4 reports stitching performance in the area of 1 fps whereas the rest can be processed in more than 30 fps. However, the stitching operation can be parallelized and offloaded, and if this operation is performed offline and stored, the results from table 3 indicate that 90-frame GOPs encoded in H.264 can be read in less than 3 seconds. Thus, real-time retrieval of the panorama video is supported by the current system.

Convert YUV to RGB	Nearest	Lanczos
Mean	4.9	5.1
Maxmimum	12.0	12.3
Minimum	4.7	4.7

⇓

Debarrel	Nearest	Lanczos
Mean	17.1	986.9
Maxmimum	21.1	1053.3
Minimum	16.8	965.6

⇓

Stitch	Nearest	Lanczos
Mean	974.4	2773.1
Maxmimum	1041.0	3027.7
Minimum	963.4	2712.7

⇓

Convert stitch RGB to YUV	Nearest	Lanczos
Mean	28.0	28.6
Maxmimum	34.8	36.3
Minimum	27.7	27.7

⇓

Write	Nearest	Lanczos
Mean	84.3	11.3
Maxmimum	3360.5	386.4
Minimum	3.8	3.8

Total	Nearest	Lanczos
Mean	1109.0	3806.3
Maxmimum	4374.1	4298.0
Minimum	1017.7	3716.0

Table 4: Profiling of the stitching pipelines: image quality vs. processing speed (in ms per frame).

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., [8, 9, 16, 30]). The accuracy of such tracking solutions vary according to scenarios and is continuously improving, but they are still giving errors, i.e., both missed detections and false positives [8].

For stadium sports, an interesting approach is to use sensors on players to capture the exact position. ZXY Sport Tracking [7] 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 20 Hz. 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 ran, number of sprints of a given speed, foot frequency, heart rate, etc. can be queried for, 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).

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 sending the a complex database query until the video starts to play the corresponding query result events.

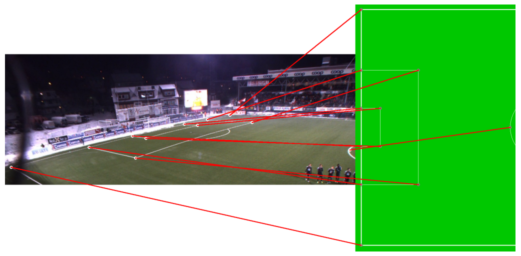
4.1 Mapping sensor positions to image pixels

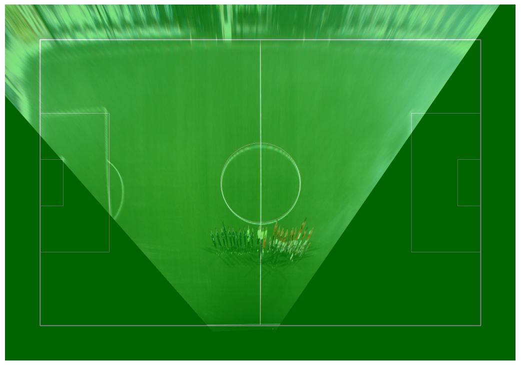
The ZXY system reports the players' positions on the field using the Cartesian coordinate system. In order to locate a player in the video, we need a translation from the sensor coordinates to the image pixels for all valid pixel coordinates in a video frame. In this respect, we calculate a 3x3 transformation matrix using fixed, known points on the field as shown in figure 5(a). Then, using the homography between two planes, each plane can be warped to fit the other as shown in figures 5(b) and 5(c) using camera 2 as an example. The accuracy of the mapping is fairly good, i.e., only in the outer areas of the image where debarreling have changed some pixels we see a very small deviation between the planes. However, if we look at the mapping to the stitched image in figure 5(d), 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 figure 10).

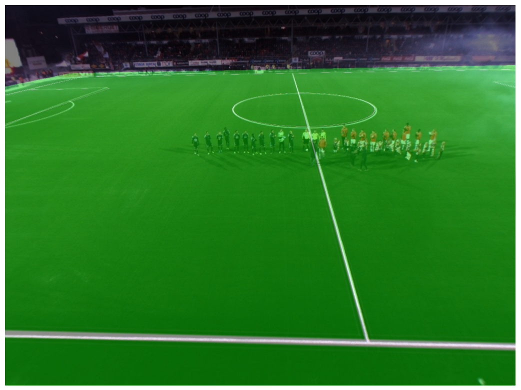
In order to have a system where the players are tracked in real-time, the $ZXY(x, y) \rightarrow pixel(u, v)$ mapping using the 3x3 matrix must be fast. A profile of the system when tracking all 22 soccer players indicate that about 7.2 – 7.7 *micro*seconds are consumed for this operation, i.e., coordinate translation is hardly noticeable compared to the other components in the system.

4.2 Automatic camera selection

As shown in figure 2, the 4 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


(a) Mapping between coordinates in the ZXY plane and the image plane.


(b) Warping and superimposing the image from camera 2 to the ZXY plane.


(c) Warping and superimposing the ZXY plane onto the image from camera 2.

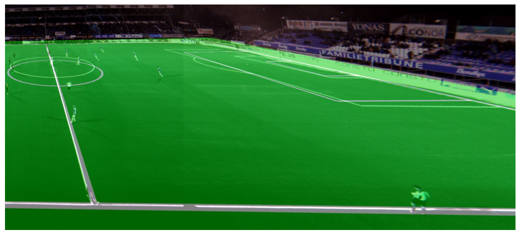
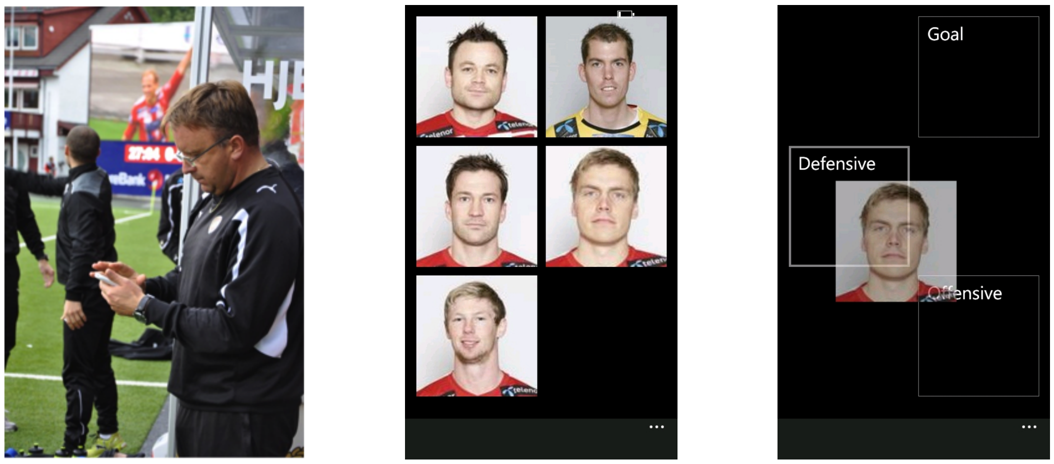

(d) Warping and superimposing the ZXY plane onto the stitched image (cropped out only parts of the field for readability).

Figure 5: Pixel mapping between the video images and the ZXY tracking system.



(a) Event tagging. (b) Select player. (c) Drag to event.

Figure 6: 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.

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 developed Muithu, a novel notational analysis system [19] that is non-invasive for the users, mobile and light-weight. 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 device, the cellular is used in the notational process as notational device. A Windows Phone 7.5 tiles based interface (see figure 6(b) and 6(c)) provides input, and figure 6(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. Figure 7 depicts the number of notational events captured during the first 9 games in the 2012 season at Alfheim. The average number of input notations for a regular 90 minute elite soccer game is in the order of 16 events.

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

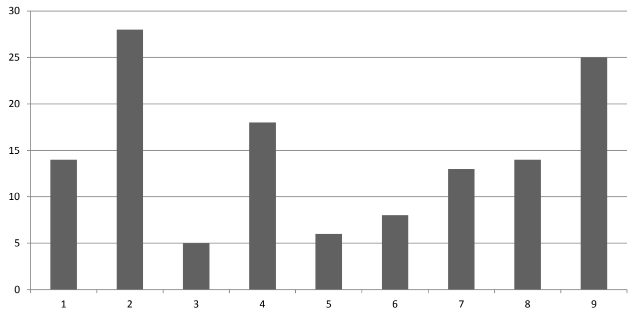


Figure 7: Number of notational events captured during the first 9 elite series games in 2012.

appear on adjacent 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 6(b) and 6(c)). Using this interface as an example, figure 8 depicts the distribution of such notations during a home game in September 2012. Observe that offensive notations are displayed above the time line, defensive notations below.

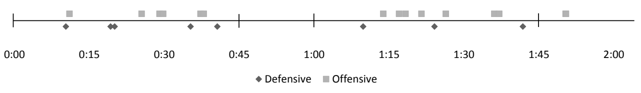


Figure 8: Notations captured in game number 9 from figure 7.

Recall of performance related events without any observation aids is traditionally problematic in soccer, but the recall abilities of the head coaches using Muithu has 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 pre-configured 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. 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 expedite 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 16 seconds is appropriate for capturing the event on video.

6. ALFHEIM STADIUM CASE STUDY

As mentioned above, we test our ideas using a case study where we have an installation at Alfheim stadium in Tromsø (Norway). The interface of the first prototype[4] is shown in figure 9, 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.

Additionally, the system has support for generating automatic summaries, i.e., 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 following pseudo-query:

```
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")
```

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 left 18-yard box. The returned timestamps and positions are then used to select video frames (selecting correct camera or the panorama picture) which are automatically presented to the user.

Extracting summaries like the example above used to be a time consuming 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 query above was measured to be around 671 ms (see table 5 for more detailed statistics). Note that this was measured on a local machine, i.e., if the display device is remote, network latency must be added.

[4]A video of the Linux-based system is available at http://www.youtube.com/watch?v=1zsgvjQkL1E. The system has been extended since the creation of this video, but due to time restrictions before the submission deadline, it has not been updated, i.e., this video does not include search. However, we already showed that we are able to search in the DAVVI system [18].

Operation	Mean	Minimum	Maximum
Query received	2.7	1.5	5.3
Query compiled	4.9	2.9	7.8
First DB row returned	500.4	482.4	532.1
First video frame displayed	671.2	648.0	794.6

Table 5: Latency profiling (in ms) of the event extraction operation using ZXY and the video system.

In a similar way, we can also extract video events based on expert annotations as shown in figure 9. All annotated events from the analytics subsystem appear in the list of events, and the corresponding video starts by clicking on the title. Here, we get even lower response times as we do not have to make a complex search to find the time intervals as they are embedded in the event annotation itself.

7. DISCUSSION

Bagadus is a prototype of a soccer analysis application which 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. However, our first prototype aims to prove the possible integration at the system level, rather than being optimized for performance.

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 hard. This makes finding a homography between planes difficult and error-prone, which affects the stitched result. Another problem we have identified, are parallax errors. OpenCV's autostitcher has functionality for selecting seams at places where parallax errors are less obvious. However, when stitching video, players are bound to run through a seam and parallax errors will become prominent. Such problems arise because the centers of the projection of different cameras are not well enough aligned. We are looking at solutions to eliminate this problem, one of the most interesting solutions is the arrangement of cameras similar to [12]. Furthermore, the stitching itself can be moved from a homography based stitching to more advanced warping techniques like mentioned in [22]. A rather intriguing challenge would be to incorporate such a process into Bagadus and perform it in real-time. Moreover, we have later found several promising alternative algorithms in the area of video processing (vision), e.g., [17,21–23], and there is also scope for improvement in color correction [29], since the exposure times and other parameters across the cameras may vary. Furthermore, there are large potentials for algorithm parallelization onto both multiple cores and to external processing units like graphical processing units (GPUs). This area is our primary subject of on-going work.

We have in this paper shown Bagadus in the context of analyzing only one game. However, we have earlier shown how one could search for events and on-the-fly generate video summaries in terms of a video playlist [18] over large libraries of video content. In the used test scenario, there are events identified from multiple sub-components, e.g., 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,

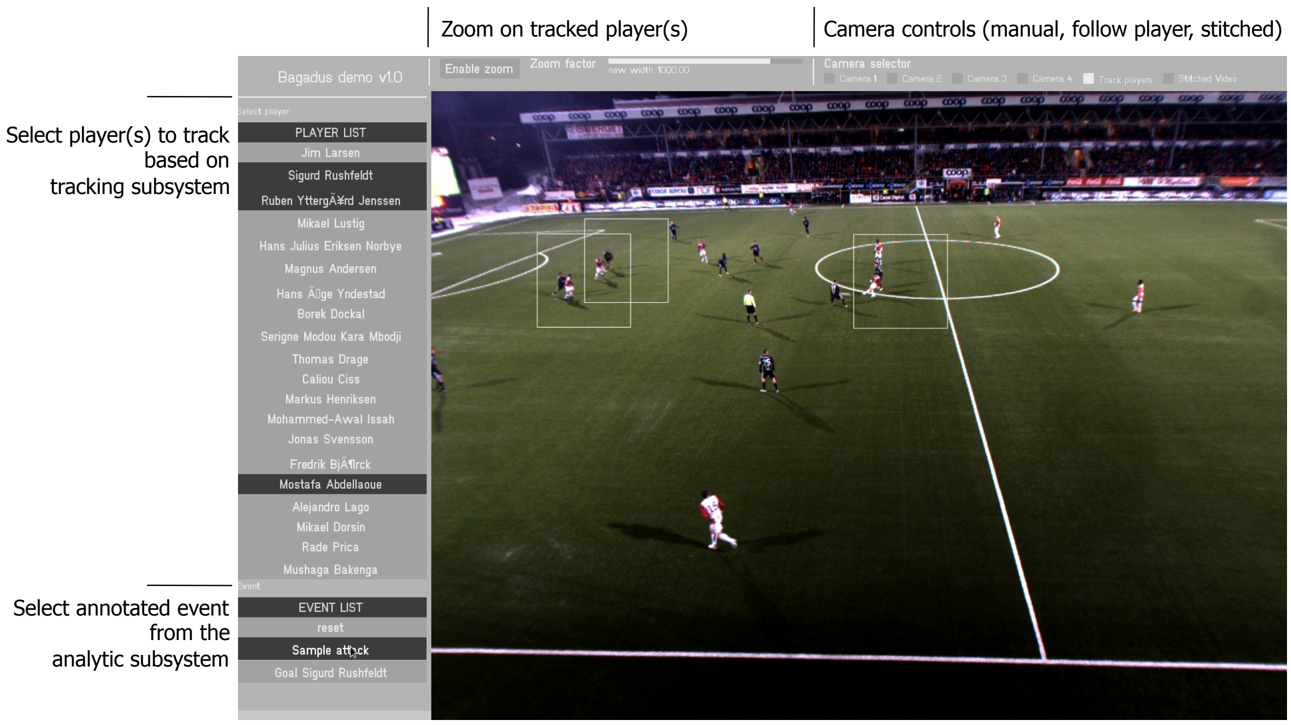


Figure 9: An example run using the Linux interface (tracking three players in camera switching mode).

i.e., the underlying video system fully supports 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, giving 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 [12, 15, 20, 31] and surveillance [11, 13, 26]. 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 codecs, 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 [18].

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 sub-millisecond level, i.e., as the players are moving fast across cameras, imperfect synchronization 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 (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 (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 timestamp, the effects have been small. Another more visible (still very infrequent) effect of time skew is that the box-marker marking the players(s) in the video gives small misplacement errors as shown in figure 10. However, the bounding box is slightly larger compared to the person-object 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. We have for example designed a time code protocol which reduce the effect of clock drift in the trigger box by assigning each frame a time stamp according to the clock in the video capture-machine. This protocol is currently under development.

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 ± one meter. This could give effects like shown in figure 10, but for the practical purposes of our case study, it has no influence on the results.

The players are as described tracked using the ZXY Sport Tracking system. Another issue which is not yet included in Bagadus is ball tracking, i.e., a feature that will potentially improve the analysis further. Even though ball tracking is

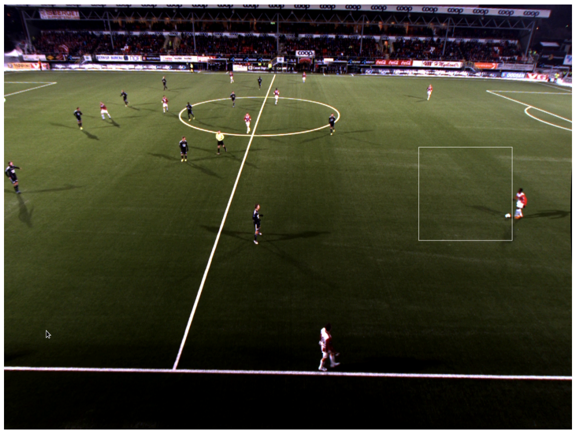


Figure 10: An example of when the tracking box fails to capture the tracked player. Even though our analysis of the system indicate very infrequent errors, it may be various reasons for failed tracking, e.g., both clock skew, sensor system accuracy and coordinate mapping.

not officially approved by the international soccer associations due to the limited reliability and failure to provide a 100% accuracy, there exist several approaches. For example, Adidas and Cairos Technologies have earlier tried to put sensors inside the ball, i.e., using a magnetic field to provide pinpoint accuracy of the ball's location inside the field [6, 27]. Other approaches include using multiple cameras to track the ball. Hawk-Eye [2] 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 to be included.

This paper 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

In this paper, we have presented a 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. The performance of most of the system enables real-time operations, except the non-optimized, sequential stitching operation. However, by performing the stitching offline, real-time retrieval and playout are supported at the cost of a higher storage requirement. Furthermore, we have presented functional results using a prototype installation at Alfheim Stadium in Norway. Bagadus enables a user to follow and zoom in on particular player(s), playout 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 several areas for future improvements, e.g., in the areas of image quality improvements, performance enhancements and subjective user evaluations. All these areas are subjects for ongoing work, e.g., we are testing algorithms discussed in section 7 for improving the image quality, and we are parallelizing algorithms onto multiple cores and offloading calculations to GPUs for speed improvements.

Acknowledgements

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 Marius Tennøe, Henrik Kjus Alstad, Espen Helgedagsrud, Mikkel Næss (students at the University of Oslo) and Roger B.A. Hansen (student at the University of Tromsø). Kai-Even Nilssen has been extremely helpful with the practical installation at Alfheim, and the coaches in TIL (Per-Mathias Høgmo and Agnar Christensen) have given feedback on the functionality of the system.

9. REFERENCES

[1] Camargus - Premium Stadium Video Technology Inrastructure. http://www.camargus.com/.

[2] Hawk-Eye. http://www.hawkeyeinnovations.co.uk/page/sports-officiating/football.

[3] Interplay sports. http://www.interplay-sports.com/.

[4] Prozone. http://www.prozonesports.com/.

[5] Stats Technology. http://www.sportvu.com/football.asp.

[6] The adidas intelligent football. http://www.gizmag.com/adidas-intelligent-football/8512/.

[7] ZXY Sport Tracking. http://www.zxy.no/.

[8] Horesh Ben Shitrit, Jerome Berclaz, Francois Fleuret, and Pascal Fua. Tracking multiple people under global appearance constraints. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 137–144, November 2011.

[9] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806 –1819, 2011.

[10] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, August 2007.

[11] Chao-Ho Chen, Tsong-Yi Chen, Je-Ching Lin, and Da-Jinn Wang. People tracking in the multi-camera surveillance system. In *Proceedings of International Conference on Innovations in Bio-inspired Computing and Applications (IBICA)*, pages 1 –4, December 2011.

[12] Christoph Fehn, Christian Weissig, Ingo Feldmann, Markus Muller, Peter Eisert, Peter Kauff, and Hans Bloss. Creation of high-resolution video panoramas of sport events. In *Proceedings of the IEEE International Symposium on Multimedia (ISM)*, pages 291 –298, December 2006.

[13] Luis M. Fuentes and Sergio A. Velastin. People tracking in surveillance applications. *Image and Vision Computing*, 24(11):1165 – 1171, 2006.

[14] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[15] Sachiko Iwase and Hideo Saito. Parallel tracking of all soccer players by integrating detected positions in multiple view images. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 751 – 754, August 2004.

[16] Hao Jiang, Sidney Fels, and James J. Little. A linear programming approach for multiple object tracking. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2007.

[17] Hailin Jin. A three-point minimal solution for panoramic stitching with lens distortion. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.

[18] 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. DAVVI: A prototype for the next generation multimedia entertainment platform. In *Proceedings of the ACM Multimedia conference (ACM MM)*, pages 989–990, October 2009.

[19] Dag Johansen, Magnus Stenhaug, Roger Bruun Asp Hansen, Agnar Christensen, and Per-Mathias Høgmo. Muithu: Smaller footprint, potentially larger imprint. In *Proceedings of the IEEE International Conference on Digital Information Management (ICDIM)*, pages 205–214, August 2012.

[20] Jinman Kang, Isaac Cohen, and Gerard Medioni. 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)*, pages 172–179, 2003.

[21] Jubiao Li and Junping Du. Study on panoramic image stitching algorithm. In *Proceedings of the Pacific-Asia Conference on Circuits, Communications and System (PACCS)*, pages 417–420, August 2010.

[22] Wen-Yan Lin, Siying Liu, Y. Matsushita, Tian-Tsong Ng, and Loong-Fah Cheong. Smoothly varying affine stitching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 345–352, 2011.

[23] Tomohiro Ozawa, Kris M. Kitani, and Hideki Koike. Human-centric panoramic imaging stitching. In *Proceedings of the Augmented Human International Conference (AH)*, pages 20:1–20:6, 2012.

[24] Simen Sægrov, Alexander Eichhorn, Jørgen Emerslund, Håkon Kvale Stensland, Carsten Griwodz, Dag Johansen, and Pål Halvorsen. Bagadus: An integrated system for soccer analysis (demo). In *Proceedings of the International Conference on Distributed Smart Cameras (ICDSC)*, October 2012.

[25] Valter Di Salvo, Adam Collins, Barry McNeill, and Marco Cardinale. Validation of Prozone: A new video-based performance analysis system. *International Journal of Performance Analysis in Sport (serial online)*, 6(1):108–119, June 2006.

[26] Nils T. Siebel and Stephen J. Maybank. Fusion of multiple tracking algorithms for robust people tracking. In *Proceedings of the European Conference on Computer Vision-Part IV (ECCV)*, pages 373–387, 2002.

[27] Cairos technologies. Goal Line Technology (GLT) system. http://www.cairos.com/unternehmen/gltsystem.php.

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

[29] Yingen Xiong and Kari Pulli. Fast panorama stitching for high-quality panoramic images on mobile phones. *IEEE Transactions on Consumer Electronics*, 56(2), May 2010.

[30] Ming Xu, James Orwell, and Graetne Jones. Tracking football players with multiple cameras. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 2909–2912, October 2004.

[31] Sunghoon Choi Yongduek, Sunghoon Choi, Yongduek Seo, Hyunwoo Kim, and Ki sang Hong. Where are the ball and players? Soccer game analysis with color-based tracking and image mosaick. In *Proceedings of the International Conference on Image Analysis and Processing (ICIAP)*, pages 196–203, 1997.

[32] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 666 –673, 1999.