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.**

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.

Fig. 1: The re-projected virtual view. The panorama video with the marked region of interest is shown together with the generated virtual camera.

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 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.

[1]This size depends on the lighting and weather conditions.

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 tilting correspond to strictly navigating the panorama along the horizontal and vertical directions respectively. Lack of this

[2]Due to possible poor quality of printers, it is recommended to analyze the images on screen.

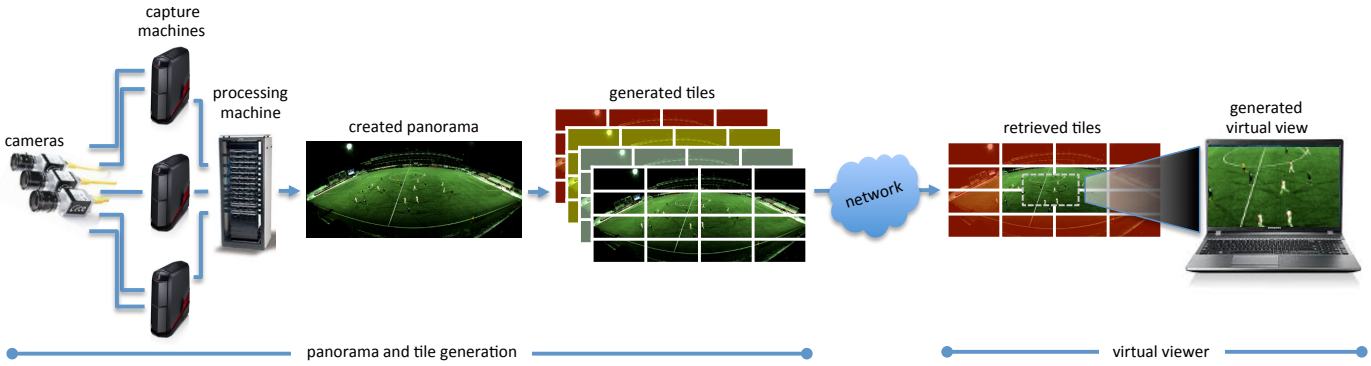


Fig. 3: The generated cylindrical panorama video is tiled and encoded in multiple qualities on the server side. The client side uses the current position of the virtual camera to retrieve full quality tiles for the virtual view and low quality (red) tiles outside the field of view of the virtual camera.

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.

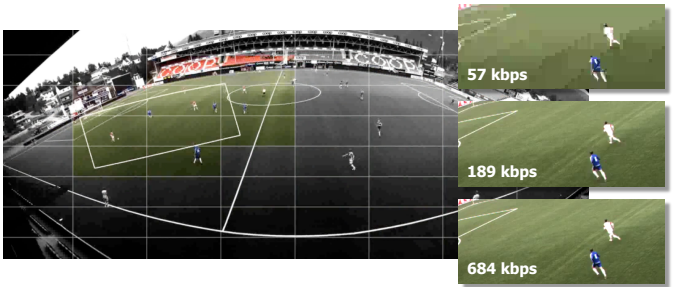


Fig. 4: Dividing the panorama video into 8x8 tiles, and encoding each tile in different quality

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

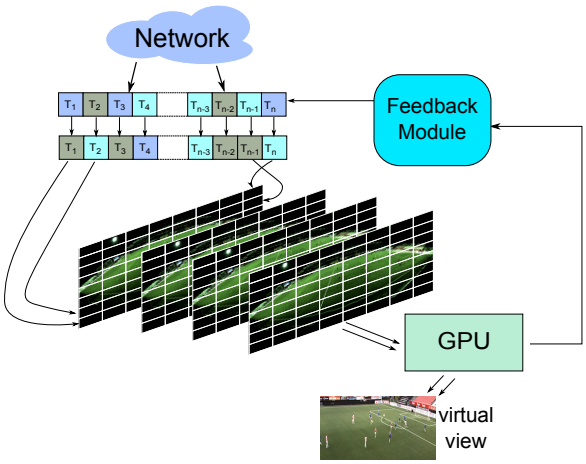


Fig. 5: The architecture on the client side to support tiling. The bottom array of tiles is the one that is being used to display the current view, the top array contains the tiles that are being fetched in a quality depending on the current viewing parameters.

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

[3]Videos: http://home.ifi.uio.no/vamsidhg/pv2015/

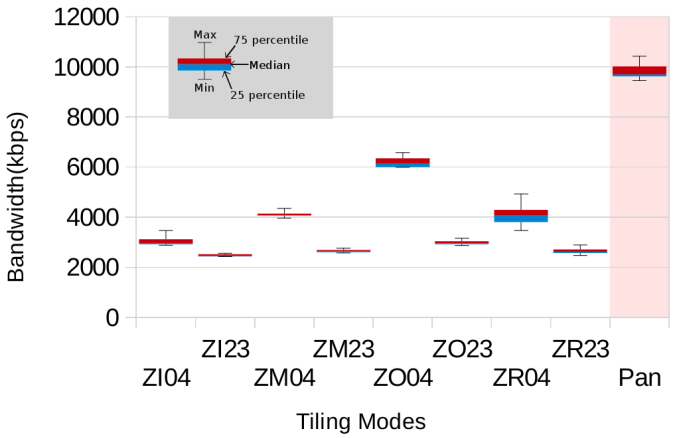


Fig. 7: The bandwidth during a 30 second virtual-view operation. The zoom-scenarios are zoomed-in (ZI), zoomed-out (ZO), medium-zoomed (ZM) and random zoom (ZR). 04 implies highest-lowest quality and 23 implies medium-low quality. As a reference, the plot also includes the results for the full panorama. The legend is equal for all figures.

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.

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

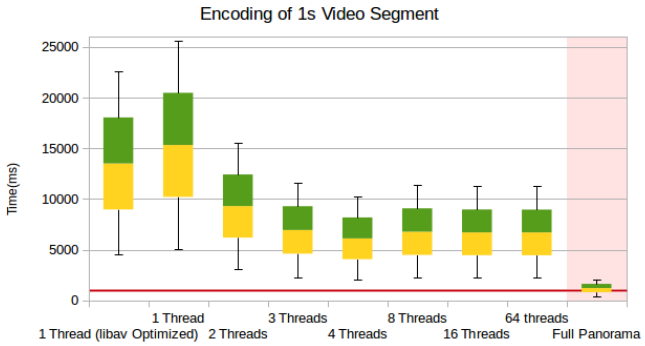


Fig. 8: The time to partition and encode the tiles of the full panorama into 5 different qualities using multiple threads. The encoding of a full panorama video in full quality is included as a reference.

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 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.

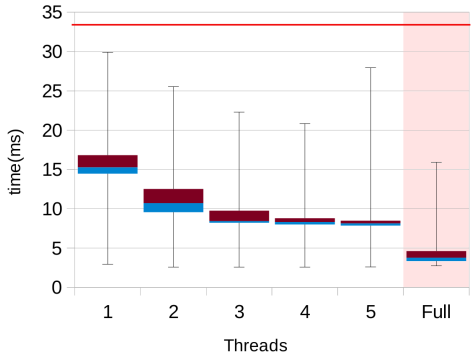
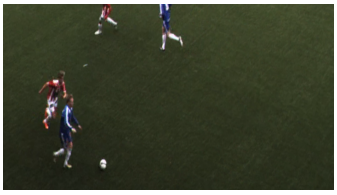


Fig. 9: Time to decode one frame using multiple threads in tiling case in comparison with the full panorama. The real-time 30fps deadline is marked as dashed line.

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