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

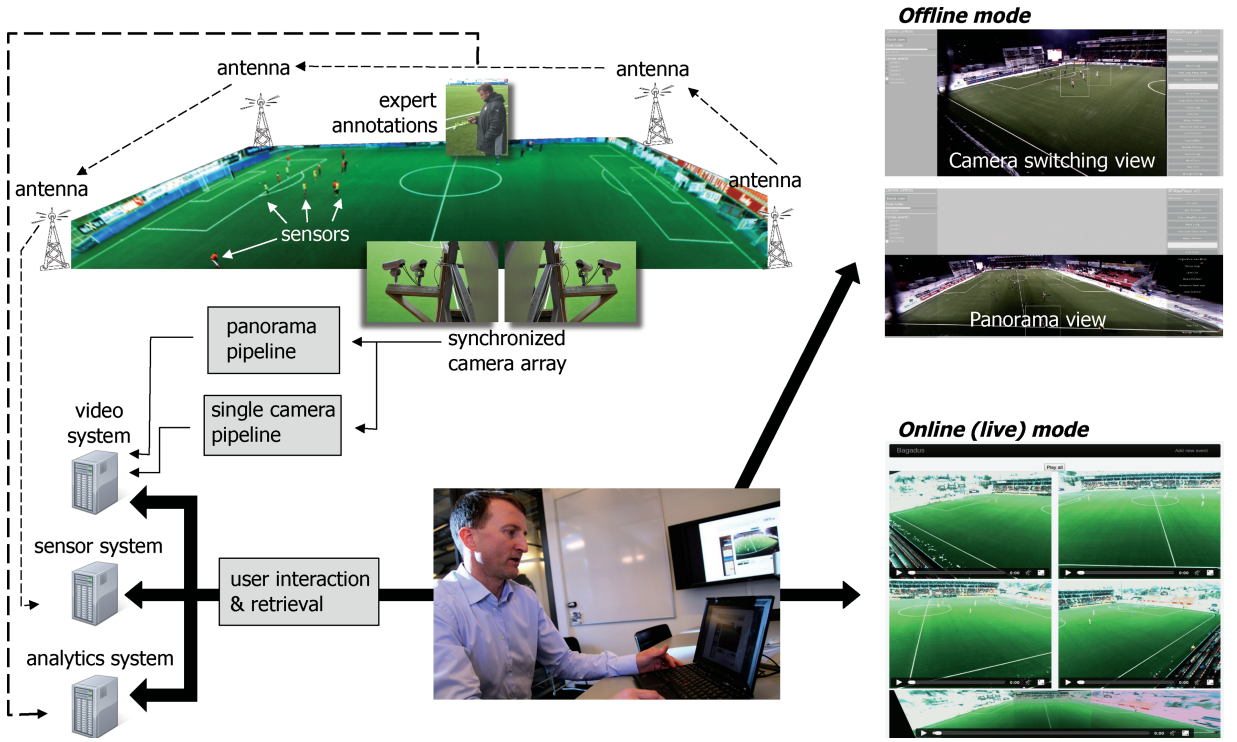


Fig. 1. Overall Bagadus architecture.

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

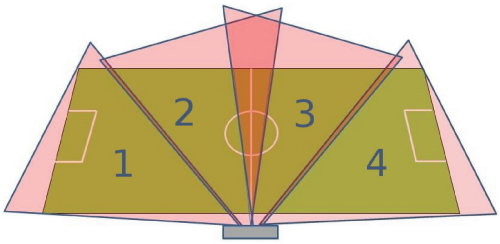


Fig. 2. Camera setup at Alfheim Stadium.

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

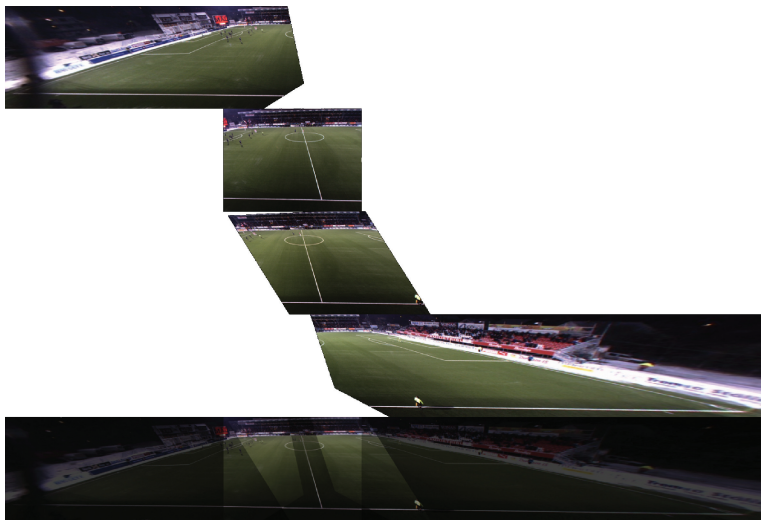
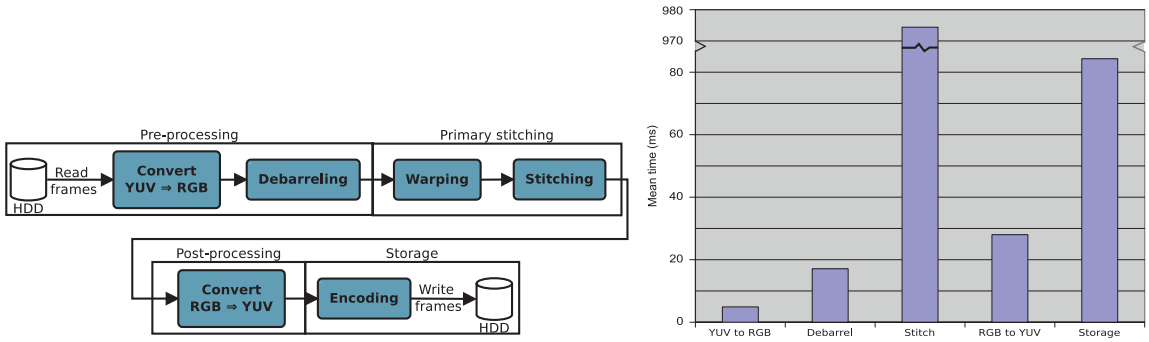


Fig. 3. The stitching process. Each image from the four different frames are warped and combined into a panorama.



(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

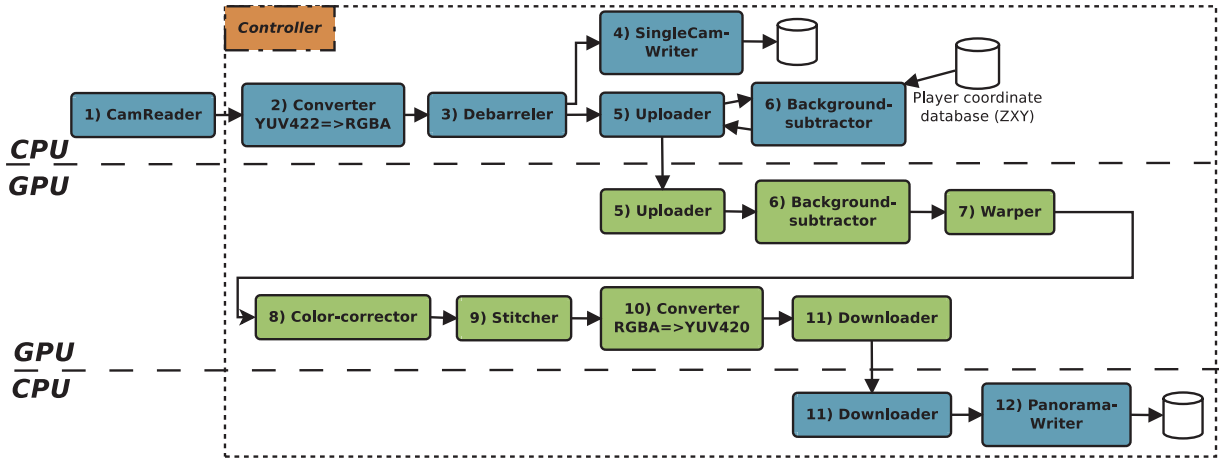


Fig. 5. The parallel and distributed processing implementation of the stitching pipeline.

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.

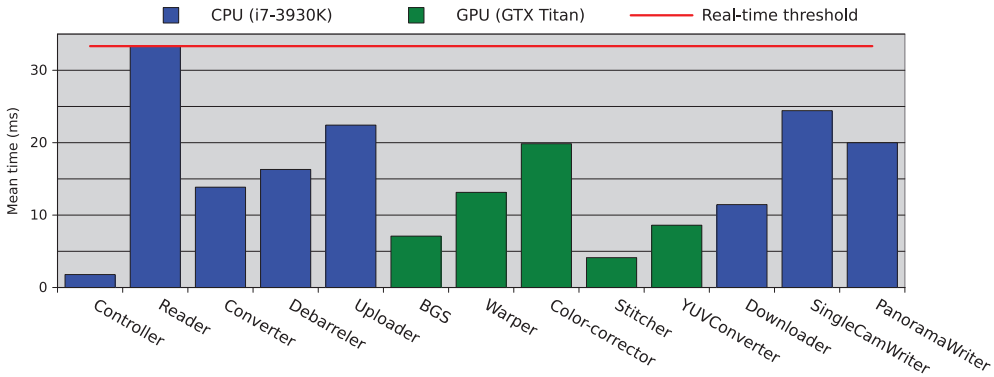
(c) Dynamic stitch with no color correction. In the left image, one can see the seam search area between the red lines, and the seam in yellow. In the right image, one clearly sees the seam going outside the player, but there are still color differences.

(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.

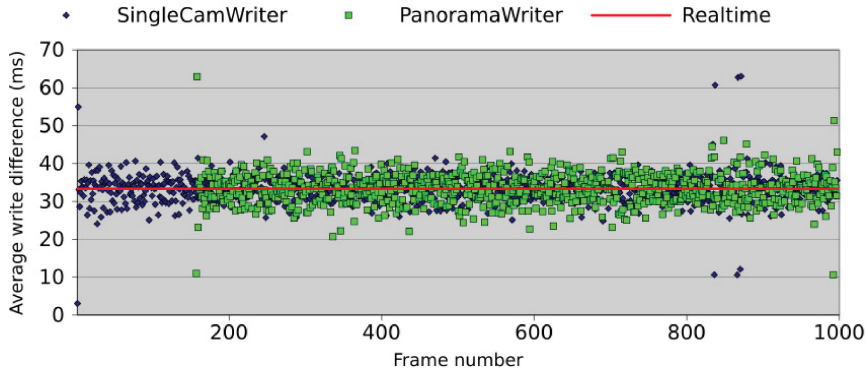
Fig. 6. Stitcher comparison: improving the visual quality with dynamic seams and color correction.

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).

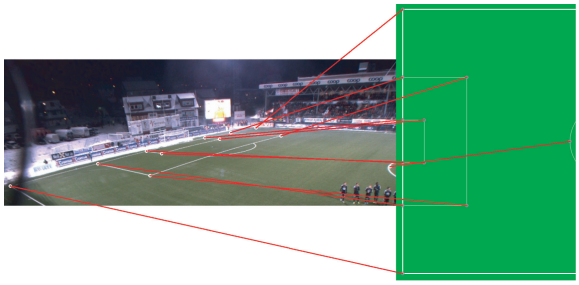
In order to have a system where the players are tracked in real time, the $ZXY(x, y) \rightarrow pixel(u, v)$ mapping using the 3×3 matrix must be fast. A profile of the system when tracking all 22 soccer players indicates that about 7.2–7.7 microseconds are consumed for this operation, that is, coordinate 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.

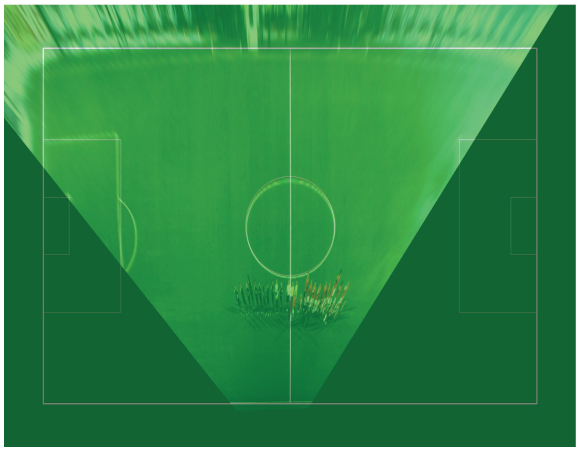
5. ANALYTICS SUBSYSTEM

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).



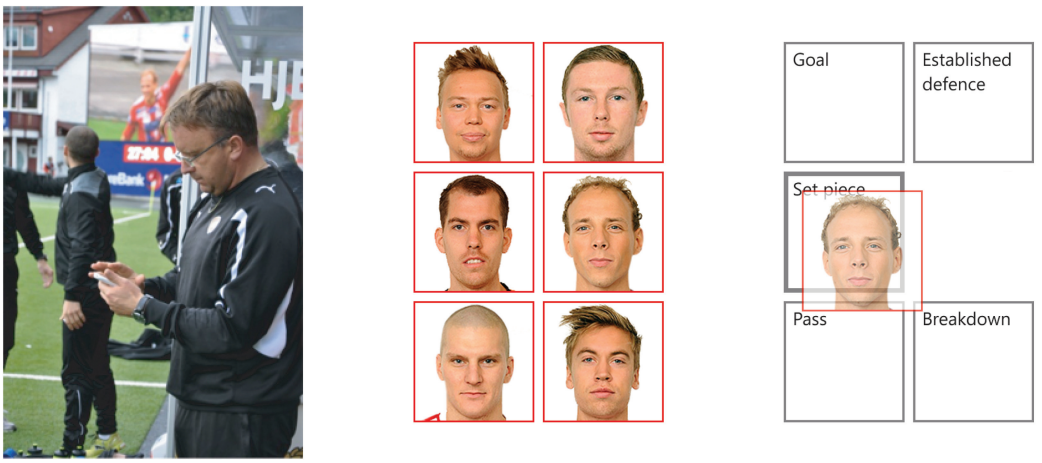
(c) Warping and superimposing the image from camera 2 to the ZXY plane.

(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.

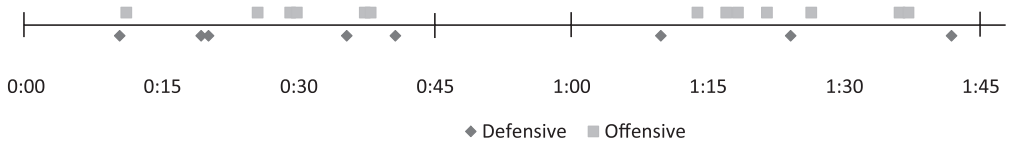


Fig. 10. An example of notations captured during a game (time axis in HH:MM after game start). Observe that offensive notations are displayed above the timeline, defensive notations below.

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.

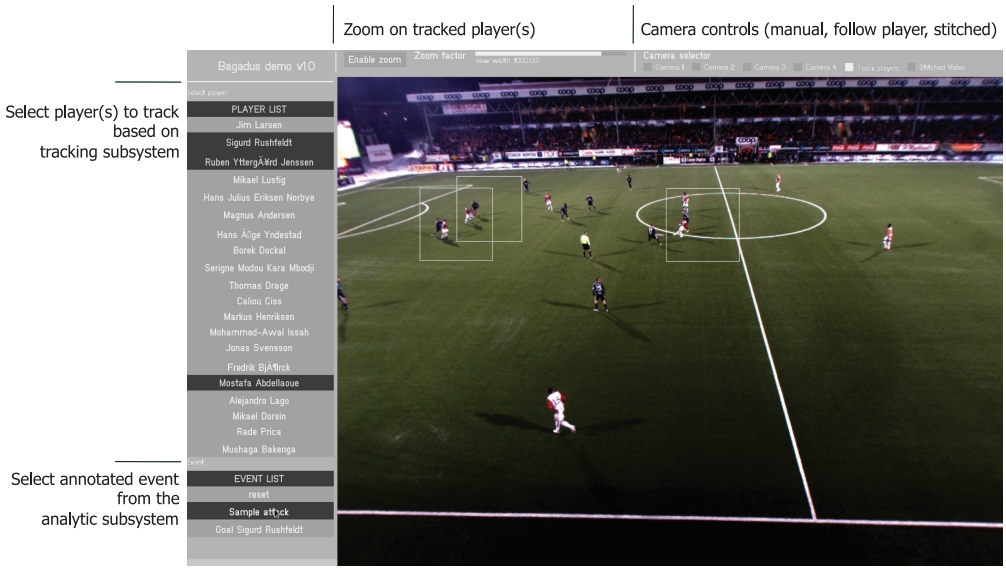


Fig. 11. The offline Linux interface (tracking three players in camera-switching mode).

```
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

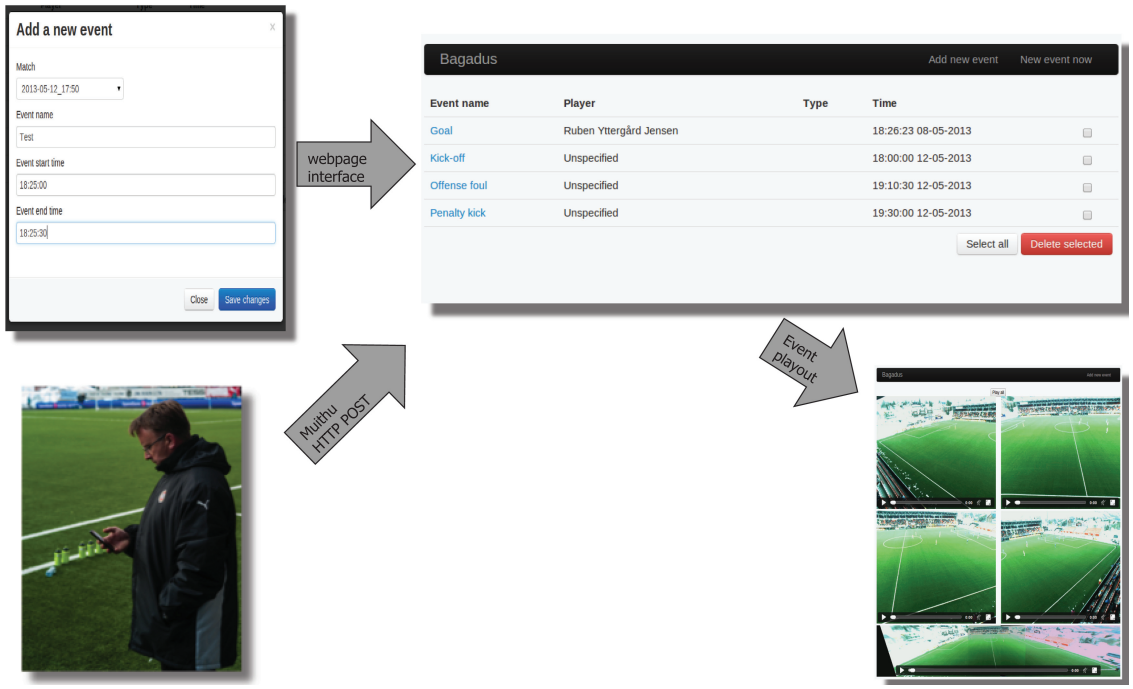


Fig. 13. The online HTML5 interface used for expert annotated events. Here the events are sorted by player and then time.

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