Camera Synchronization for Panoramic Videos

Chapter · March 2018

DOI: 10.1007/978-3-319-65840-7_20

Chapter 20
Camera Synchronization for Panoramic Videos

Vamsidhar R. Gaddam, Ragnar Langseth, Håkon K. Stensland, Carsten Griwodz, Michael Riegler, Tomas Kupka, Håvard Espeland, Dag Johansen, Håvard D. Johansen and Pål Halvorsen

Abstract Multi-camera systems are frequently used in applications such as panorama videos creation, free-viewpoint rendering, and 3D reconstruction. A critical aspect for visual quality in these systems is that the cameras are closely synchronized. In our research, we require high-definition panorama videos generated in real time using several cameras in parallel. This is an essential part of our sports analytics system called *Bagadus*, which has several synchronization requirements. The system is currently in use for soccer games at the Alfheim stadium for Tromsø IL and at the Ullevaal stadium for the Norwegian national soccer team. Each Bagadus installation is capable of combining the video from five 2 K cameras into a single 50 fps cylindrical panorama video. Due to proper camera synchronization, the produced panoramas exhibit neither ghosting effects nor other visual inconsistencies at the seams. Our panorama videos are designed to support several members of the

V. R. Gaddam · H. K. Stensland · C. Griwodz · M. Riegler · P. Halvorsen (✉)
Simula Research Laboratory, Fornebu, Norway
e-mail: paalh@simula.no

V. R. Gaddam
e-mail: vamsidhar@simula.no

H. K. Stensland
e-mail: haakonks@simula.no

C. Griwodz
e-mail: griff@simula.no

R. Langseth · T. Kupka · H. Espeland
ForzaSys AS, Fornebu, Norway
e-mail: ragnar@forzasys.com

T. Kupka
e-mail: tomas@forzasys.com

H. Espeland
e-mail: haavares@forzasys.com

D. Johansen · H. D. Johansen
UiT – The Artic University of Norway, Tromsø, Norway
e-mail: dag@cs.uit.no

H. D. Johansen
e-mail: haavardj@cs.uit.no

M. Montagud et al. (eds.), *MediaSync*,
https://doi.org/10.1007/978-3-319-65840-7_20

trainer team at the same time. Using our system, they are able to pan, tilt, and zoom interactively, independently over the entire field, from an overview shot to close-ups of individual players in arbitrary locations. To create such panoramas, each of our cameras covers one part of the field with small overlapping regions, where the individual frames are transformed and stitched together into a single view. We faced two main synchronization challenges in the panorama generation process. First, to stitch frames together without visual artifacts and inconsistencies due to motion, the shutters in the cameras had to be synchronized with sub-millisecond accuracy. Second, to circumvent the need for software readjustment of color and brightness around the seams between cameras, the exposure settings were synchronized. This chapter describes these synchronization mechanisms that were designed, implemented, evaluated, and integrated in the Bagadus system.

Keywords Camera array · Panorama video · Frame stitching · Shutter synchronization · Exposure synchronization

20.1 Introduction

Media synchronization is certainly not a new research area. Over the last decades, various mechanisms have been proposed, covering a wide span of usage scenarios. In our research on the Bagadus system [1], we developed a camera array system for real-time video panorama recording. To generate high-quality wide field-of-view panoramas, Bagadus ensures that multiple components between the back-end processing machines and cameras are accurately synchronized in various ways. For instance, frame-accurate synchronization is required to avoid errors around moving objects, different colors, and luminances. As Bagadus aims at running live in real-world scenarios, real-time performance and a perceptually good video quality are key requirements. This chapter addresses mainly the challenges we faced related to synchronization of cameras to achieve our video quality goals.

Our first challenge was related to performance and image consistency issues when handing camera synchronization in software. As Bagadus was developed to capture soccer games, the objects it records often move at speeds that demand accurate camera trigger synchronization. To avoid parallax errors and ghosting effects due to moving objects captured in the seam region by several cameras, the cameras must be shutter-synchronized with sub-millisecond accuracy. In this respect, the fairly widespread software approach to handling camera synchronization, which consists of measuring the cameras' temporal drift to sort the drift-compensated frames before panorama rendering, fails to solve the consistency problem. Experience with software-triggered exposure suffered from operating system scheduler limitations. Our solution was to add an external trigger signal that is broadcast synchronously to all cameras.

Our second challenge was related to the inconsistent visual quality we initially observed across different cameras. It is important for the visual quality of panorama videos that there is a consistent brightness and color balance across the pixels that are captured by the individual cameras. We can install identical color lookup tables on all cameras, but we cannot fix exposure. Since Bagadus is deployed outdoors in a geographic region with unstable weather conditions, a constant exposure setting would yield intervals of over- or underexposed frames. Likewise, we cannot use auto-exposure on all cameras. Since auto-exposure relies on a brightness histogram, and all cameras see different parts of the field without a common overlap, each camera will make independent and different exposure decisions, both in case of a full-frame histogram and a histogram for a region of interest (ROI). For Bagadus, we therefore developed a solution where the cameras use one pilot camera that performs auto-exposure. Its exposure settings are then read by host software and copied to all other cameras. This achieves very good exposure synchronization while adapting to most external conditions.

20.2 The Bagadus Sport Analysis System

Bagadus [1–3] is a sports analysis systems for real-time panorama video presentation of sport events. The system is currently installed in two stadiums in Norway: Alfheim stadium in Tromsø (Tromsø IL) and Ullevaal stadium in Oslo (the Norwegian national soccer team). An overview of the architecture and interaction of the different components is given in Fig. 20.1. As shown in the figure, Bagadus consists of three main sub-systems: a player tracking system, a coach annotation system, and a video system.

Using these sub-systems, Bagadus implements and integrates many well-known components to support the selected sport application scenario where the combination of different technologies raises requirements on both spatial synchronization and temporal synchronization of multiple signals from independent sources. The novelty lies in the combination and integration of components enabling automatic presentation of video events based on the positional sensor and analytics data that are synchronized with the video system. For example, Bagadus supports automatic presentation of video clips of all the situations where a given player runs faster than 10 miles per hour or when all the defenders were located in the 18-yard penalty box [4]. Furthermore, we can select and follow single players and groups of players in the video, and retrieve and repeat the events annotated by expert users. Thus, where people earlier used a huge amount of time for analyzing the game manually, generating video summaries, and making long reports, Bagadus serves as an integrated system where the required operations and the synchronization with video are managed automatically.

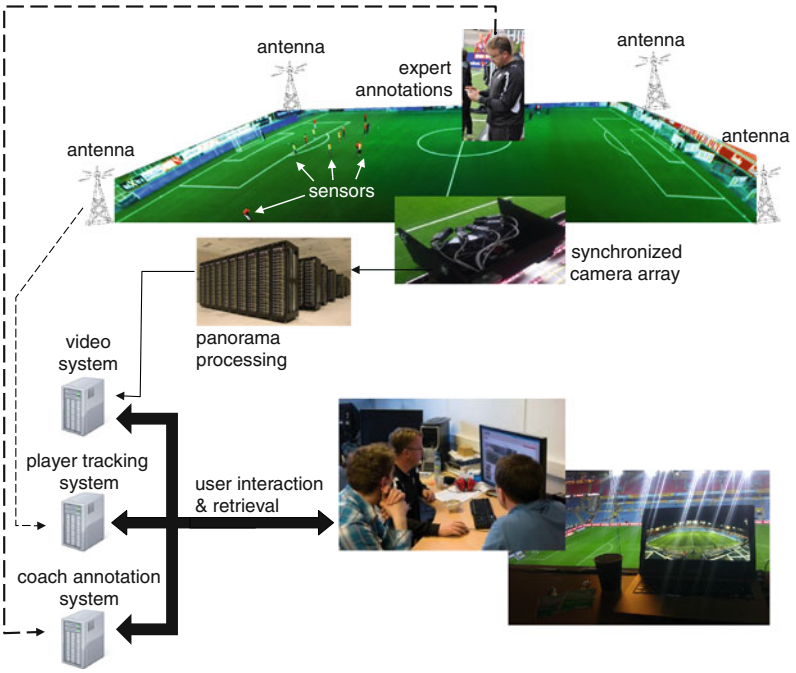


Fig. 20.1 Architectural overview of the Bagadus system

20.2.1 Player Tracking System

There exist several player tracking systems based on, for example, GPS, radio, or optical technologies. Bagadus potentially can use any type of tracking, but in the current installation, the *player tracking system* imports player positions from the ZXY Sport Tracking system [5]. ZXY is designed for fixed installations using advanced radio technology for both its positioning and data communication, and it collects several objective data elements including the players' positions typically at 20 or 40 Hz.

In Bagadus, the player positions are used not only to gather player movement statistics, but also to highlight players in the videos and to extract video clips based on player movement. For this, the time stamps of the video clips and the time stamps of the sensor data must be synchronized. Fortunately, this is not a very hard synchronization problem since the clocks on all machines are synchronized using the network time protocol (NTP) [6]. The clock synchronization accuracy provided by NTP is sufficient for aligning positional data with video clips. This synchronization issue will therefore not be covered further in this chapter.

20.2.2 Coach Annotation System

The *coach annotation system* [7] replaces the traditional pen-and-paper annotations that coaches previously had to (and still) rely on for annotating soccer games. Using the Bagadus mobile app, coaches can annotate the video quickly with a press of a button. The system is based on hindsight recording where the end of the event is marked, and the system then captures video prior to this mark. Furthermore, the registered events are stored in an analytics database that can later be shown along with the corresponding video of the event. Here, the time of the tagged event and the video must be synchronized, and the mobile device therefore synchronizes its local time with the NTP machines. However, it is only a fairly loose second-granularity time-sync requirements, and hence, it is not further discussed here.

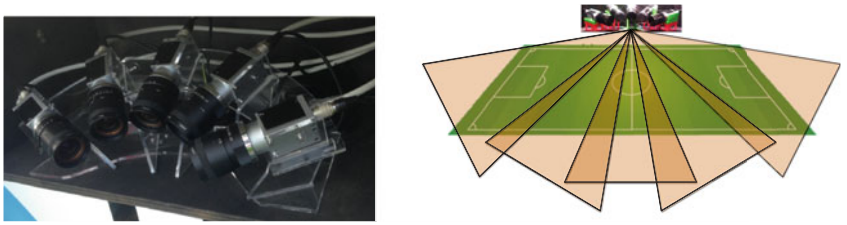
20.2.3 Video System

The *video system* consists of multiple small shutter and exposure synchronized cameras that record high-resolution video of the soccer field. The cameras are set up to cover the full field with sufficient overlap to identify common features necessary for camera calibration and panorama frame stitching. The frame stitching is based on a homographic mapping of each camera image into the panorama. The mapping matrices for each camera are statically generated through an offline calibration process. Nevertheless, the shutters and exposures of each camera must still be coordinated and dynamically adjusted.

20.2.3.1 Camera Setup

To capture the entire soccer field, Bagadus uses five 2K Basler industry vision cameras [8], each delivering a maximum resolution of 2046 × 1086 pixels at 50 frames per second (fps) over Gigabit Ethernet. We use an 8-mm lens [9] with virtually no image distortion and avoided having to apply lossy debarreling operations. To maximize the panorama resolution, the cameras are rotated by 90°, giving a per-camera field-of-view of 66°.

The cameras are statically mounted in a circular pattern, each covering a different part of the soccer field, as shown in Fig. 20.2. The cameras are pitched, yawed, and rolled to look directly through a common point about 5 cm in front of the lenses in an attempt to reduce parallax effects. As a result, only minor adjustments are required before the images can be stitched together into the panorama frame.



(a) Camera rig **(b)** Overlapping camera capture regions

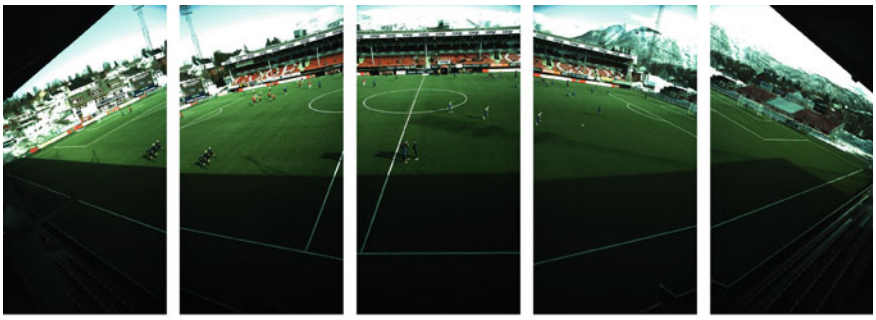
Fig. 20.2 Camera setup at the stadiums

20.2.3.2 Video Processing Pipeline

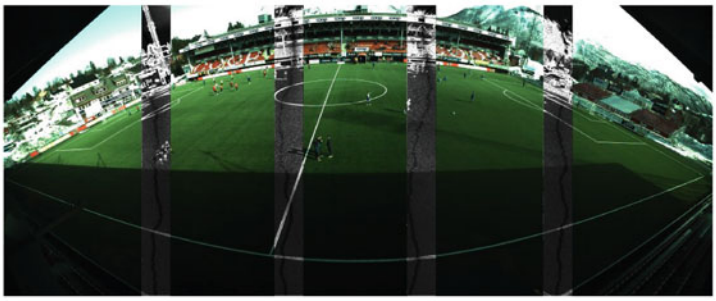
To support live video feeds, our camera output is processed through a custom real-time video pipeline [10, 11]. In its basic configuration, Bagadus' video pipeline consists of a background subtractor, a warper, a color corrector, a stitcher, and an encoder. The pipeline can also be configured to include YUV and RGB converters, Bayering modules [12], a debarreler, and a high dynamic range (HDR) module [13]. Most pipeline components can utilize both the central processing unit (CPU) and the graphics processing unit (GPU) [14, 15] for high-performance data processing. Figure 20.3 illustrates the data processing used in our two Bagadus deployments, with samples of individual recorded video frames as shown in Fig. 20.3a. Figure 20.3b depicts how these frames are stitched together in the overlapping regions by a dynamically calculated seam. The final panorama video (see Fig. 20.3c) has a frame resolution of 4096×1680 pixels and is encoded and compressed with the x264 encoder [16] into the H.264/MPEG-4 Advanced Video Coding (AVC) compression format.

20.2.3.3 Interactive Pan-Tilt-Zoom

When encoding video, we prioritize video quality over compression. As this leads to high demands on available network bandwidth to transfer the full-resolution panorama video, we explored the use of personalized views to reduce bandwidth requirements. Personalized interactive Pan-Tilt-Zoom (PTZ) operations that lead to server-sided encoding achieve lower bandwidth consumption by transcoding the panorama to a final view at a resolution appropriate for the receiver [17]. Moreover, personalized views for a large number of interactive receivers can be constructed from tiles that are stored in several representations of different quality, allowing each receiver to stream high-quality sub-streams (tiling) only for a region of interest [18, 19]. This means that a client only retrieves the video streams (tiles) at high resolutions that are relevant to the user experience. The other tiles can be either retrieved in low quality or omitted completely. Such tiling is also possible using the Spatial Relationship Description (SRD) extension [20] of the Dynamic Adaptive Streaming



(a) Alfheim stadium individual camera frames



(b) Alfheim stadium stitching areas

(c) Alfheim stadium generated panorama

(d) Ullevaal stadium generated panorama

Fig. 20.3 Panorama examples from Alfheim stadium and Ullevaal stadium

over HTTP (DASH) standard, and combined with the new tiling feature of the High Efficiency Video Coding (HEVC) standard [21], a tiled video can be stitched in the compressed domain and decoded on a single CPU while switching tile quality based on the active region of interest. Nevertheless, the quality (an invisible seam) of the original panorama, tiled or not, is of high importance as the users may dynamically move their regions of interest across the seams between the cameras.

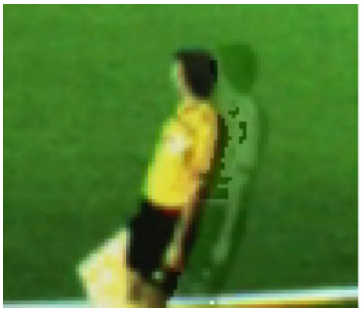
20.2.4 Synchronization Challenges

As outlined in this section, we encountered several synchronization issues while building and deploying Bagadus in order to ensure a high-quality panorama video. Synchronization of video frames with data from the positional and event annotation systems was trivially overcome by having hardware clocks synchronized using NTP. The sub-second accuracy provided by NTP [22] is sufficient for this purpose. We did, however, also encounter two additional and non-trivial synchronization challenges, both directly related to the visual quality of the final panorama video. These were: (1) *shutter synchronization*, which is required to avoid that moving objects appear at different positions in the corresponding frames from different cameras; and (2) *exposure synchronization*, as the cameras have different metering points for auto-exposure potentially resulting in mismatching lighting levels in the various cameras. In the remainder of this chapter, we focus on how Bagadus overcomes these two challenges.

20.3 Camera Shutter Synchronization

To ensure an errorless generation of panorama frames with respect to parallax errors stemming from moving objects, we need a synchronization signal that triggers the camera shutters. In this section, we show the potential problems of parallax errors stemming from lack of camera trigger synchronization. We then describe our solution and show how we have solved the problem using external trigger signals.

The initial processing stages of the Bagadus pipeline are processed independently, making use of three computers and multiple threads on each of them. This is necessary to deal with the bandwidth of five gigabit Ethernet-connected cameras. The three machines are running in the same local area network and synchronized by NTP to the same time server, which ensures a fairly accurate clock synchronization and suppresses drift [22]. To keep track of frames from different streams that contribute to the same panorama frame, we write a local Unix time stamp into the header of each frame as it is retrieved from the camera. This time stamp is maintained throughout the processing modules, but otherwise ignored until the frame is rendered into the panorama. At that point, we read the time stamps and use them as a barrier before writing the panorama. We consider two time stamps t_1 and t_2 equivalent if they are



(a) Ghosting from feathering (weighted average between the two overlapping images)

(b) A parallax error where a player appears on both sides of the static vertical seam just selecting pixels from the cameras on both sides

Fig. 20.4 Ghosting effects using different stitching techniques

closer in time relative to the video's frame rate. Given a video with an f Hz frame rate, we have that:

$$t_1 \equiv t_2 \implies |t_2 - t_1| < \frac{1}{2 \times f}$$

To ensure that all moving objects are located in the same place when different cameras record corresponding frames, the camera shutters need to be synchronized. If they are not, the frames can be captured any time within the time interval for a given frame set, and even small deviations can be seen when soccer players move through the stitching seam. As an example, consider an athlete running 100 m in 10 s. If we are running our video recorder at 25 fps, the maximum time shift between frames is 40 ms, and the athlete would move 40 cm per frame.[1] At that speed, there will be a highly visible ghosting effect in the generated panorama frames, as shown in Fig. 20.4. In particular, Fig. 20.4a depicts a stitch where we have used a simple weighted average between the two overlapping images which are out of sync. Figure 20.4b shows a similar example where the player is visible on both sides of the seam.

To create a smooth transition between the stitched frames in the panorama, each source frame must be captured at the exact same time. Even small deviations can, as we illustrated above, be seen when soccer players move through the stitching seam. An important part of the frame synchronization operation is therefore to make sure that the frames are captured simultaneously. After evaluating existing approaches, we ended up with a solution where the cameras are triggered externally. The approaches for this can be divided into two main classes: software-based triggers and hardware-based triggers.

[1] This is slightly slower than the 100-m sprint world record from 2009 by Usain Bolt of 9.58 s [23]. Bolt's movement between frames would be approximately 42 cm. The speed of the Ronny Heberson's free kick [24] was clocked at a whooping 221 km/h, which would result in a difference of 2.5 m between frames from different cameras.

20.3.1 Software-Based Triggers

Earlier work on shutter synchronization has to a large extent focused on extracting the temporal relation of frames in a software post-processing step and using those relations to recreate a temporally consistent scene. Hasler et al. [25] rely on the cameras' built-in microphone to capture sound and use the audio signal to align video frames. An important problem is that audio samples are usually not time-aligned to sample accuracy but only roughly to frame accuracy. Pourcelot et al. [26] considered post-processing based on the timing of a flickering light source recorded with a shutter speed of 1000 Hz. Also, Shrestha et al. [27, 28] use flashes to synchronize individual camera timelines. Ruttle et al.'s approach of a spinning disk [29] provides high accuracy with less visual disturbance. Bradley et al. [30] go far beyond this by even removing motion blur by lighting a scene up with stroboscopic light. We found this kind of method not applicable in practice in a soccer stadium because light sources are either obstructing the field-of-view, overexpose the entire frame, or become indiscernible from lighting change in the recorded scene. In the literature, the accuracy achieved by rough clock synchronization is frequently considered acceptable for both panorama stitching and 3D reconstruction. Duckworth et al. [31] and Moore et al. [32] have illustrated the inaccuracy that is introduced if a 3D reconstruction is undertaken from unsynchronized frames. Consequently, synchronization should be the first step in 3D reconstruction. Synchronization from silhouettes has received quite a bit of attention [33–36]. Haufmann et al. [37] integrate synchronization into 3D volume carving by minimizing variations in epipolar geometry. Nischt and Swaminathan [38] extract 3D points and minimize the variation in the fundamental matrix between cameras. Shankar et al. [39, 40] sort frames by minimizing the variance between trajectories of scale-invariant feature transform (SIFT) feature points, Tao et al. [41] use point triplets, and Velipasalar and Wolf [42] use the trajectories of bounding boxes around extracted objects, while Whitehead et al. [43] use textured objects. These approaches are not compatible with the Bagadus camera setup. All of them require camera positions that allow reasonably accurate 3D localization of points. Our cameras, however, are arranged to allow panorama creation by 2D homographic transformation only, which requires (roughly) identical camera centers (i.e., a baseline of zero). Obviously, that voids all attempts of 3D reconstruction.

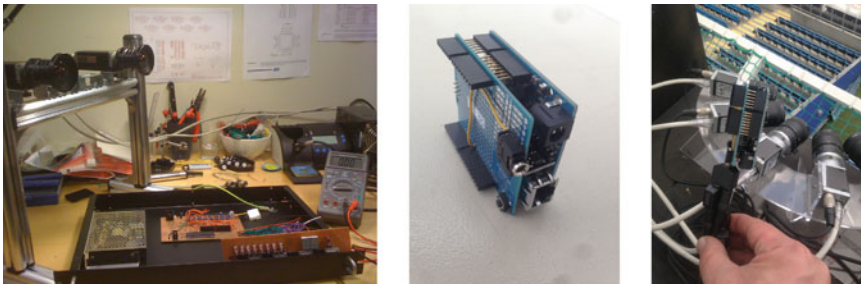
20.3.2 Hardware-Based Triggers

After evaluating existing software-based trigger solutions for Bagadus, we selected a hardware approach where the cameras are triggered externally. In this respect, several existing systems use hardware triggers. Genlock [44] is a classical method for synchronizing cameras in the world of broadcast entertainment; i.e., the video output of one source, or a specific reference signal from a signal generator, is used to

synchronize other television picture sources together. Smith [45] listed it as a basic requirement for video support in athlete training. Collins et al. [46] used it to acquire multi-view video. Alternatively, a shutter synchronization circuit for stereoscopic systems (RS232) was proposed [47] where the switching of the LCD shutter and the vertical synchronization scanning signal of the VGA graphic card can be synchronized. Blue-C used an unspecified hardware synchronization method [48]. Wilburn et al. [49] carry the synchronization signal over Firewire to an array of camera sensors and redistribute it via Ethernet (CAT5) cables in the array. A non-standardized method uses the transmission of the Society of Motion Picture and Television Engineers (SMPTE) time codes over wireless networks such as Wi-fi [50]. Litos et al. [51] constructed a PCB to refine the NTP-based rough synchronization of computers with Firewire cameras by recording and evaluating a high-resolution clock constructed from LEDs. Obviously, this yields very accurate knowledge of synchronicity, but it relies on the computer's operating system and camera implementation for synchronous recording. Sousa et al. [52] use the power supply itself to synchronize self-timed cameras attached to a single field-programmable gate array (FPGA) platform. Nguyen et al. [53] built a triggering circuit using an Arduino board to synchronize a structured light projector with a camera. Such hardware triggers seem to be the best solution for Bagadus, and we therefore built our own similar solution to meet the system requirements.

20.3.3 The Bagadus Shutter Synchronization

For Bagadus' intended deployment scenarios, the easiest approach was to directly connect the recording machines to the cameras and send trigger signals from the machines. Here, the challenge is the synchronization accuracy of the signals sent to the cameras. If the recording machines send signals to their own cameras, the trigger signal processes must compete for the CPU on highly loaded machines. Furthermore, for a perfectly stitched frame, we need machine synchronization at a more accurate level than NTP can deliver. An improvement would be to use one single process on only one of the machines, but the process is still competing for the resources. Our first approach was to develop a fairly generic hardware synchronization box for industry vision cameras, as shown in Fig. 20.5a. This box was powered and generated trigger signals for up to eight cameras over a dedicated cable. Furthermore, the user could set a frame rate for the box, with multipliers for every individual camera port. The box provided power to the cameras as well, because the cameras' 6-pin Hirose connector is used for both synchronization and power. This box worked fine, but due to own hardware design, the need for own power supply, a large box size, and a small clock drift varying with the large Norwegian temperature changes, we opted for a system using off-the-shelf equipment (Fig. 20.5b). It has a simpler operation in software, supports an arbitrary number of cameras, can draw power from an Ethernet cable, and provides synchronization with the recording machine to avoid drift. Figure 20.5c shows the box installed at Ullevaal stadium.



(a) Building trigger box, V1 **(b)** Trigger box, V2 **(c)** V2 in use

Fig. 20.5 Bagadus trigger boxes

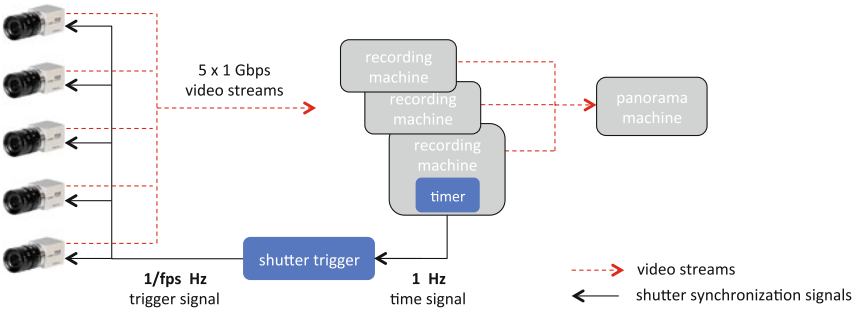


Fig. 20.6 Camera shutter synchronization in Bagadus. We have one timer running on one of the NTP synchronized recording machines syncing the trigger box once a second, and a shutter trigger box sending a signal every 1/fps second

As shown in Fig. 20.6, our trigger box[2] is divided into two parts, where we both ensure that the trigger signal is generated without interrupts (the shutter trigger) and avoid clock drift between the trigger box and the recording machines (the timer). One of the recording machines is synchronized with an NTP server and runs a process that maintains a timer. This process ensures that the clock of the synchronization box does not drift from the recording machines that is synchronized with the trigger box. The timer process sends a signal once a second to restart the timer on the synchronization box; i.e., it is synchronized once every second. In order to ensure resource availability, the process runs with the highest scheduling priority on Linux using the `SCHED_FIFO` class. Furthermore, we use an external trigger box that sends a signal to all the cameras to capture an image. Here, we use an Arduino device which uses its local clock to send a broadcast shutter trigger signal every $1/\text{fps}$ seconds. It divides the number of CPU cycles per second to the frame rate and counts the cycles between each signal.

[2]The Bagadus trigger box design and code are available here: https://bitbucket.org/mpg_code/micro-trigger-box.

Our approach ensures that all the cameras capture a new frame at exactly the same time, with an accurate frame rate. It also means that the frames will typically be either completely synchronized, or one trigger interval off, making it easier to identify when a camera falls behind and loses a frame.

20.4 Camera Exposure Synchronization

Another challenge when capturing synchronized frames from cameras having a different field-of-view is contrasting lighting conditions between these cameras. In this section, we evaluate approaches for creating a completely automatic camera array capture system with a particular focus on dynamic camera settings. There are other settings that also might be synchronized. For example, in the current scenario using several cameras to generate a panorama video, the aperture must be kept constant to avoid changing the depth of the field. This means that the only parameters that one can (or should) control are the exposure time and the gain. However, we do not have full freedom in controlling both these parameters. The electric gain introduces noise in the system, so the highest image quality is achieved when the gain is as low as possible. The exposure time has an upper limit both because it can cause motion blur during the game and also because there is a hard limit set by the frame rate. So, a priority-based estimation must be used which changes the exposure time until it reaches the threshold and then modify the gain if the exposure needs more compensation.

The cameras in our setup allow for both continuous automatic and manual exposure configuration. In an outdoor setting, lighting conditions can change quite rapidly, and we found that some form of adaptive exposure control was required. One of the main challenges in building such a system is that there is no common area of the soccer pitch visible to all the cameras (see Fig. 20.2b) that can be used for metering the light in order to set the appropriate camera parameters. The video camera is able to determine the optimal automatic exposure. However, that will be independently set for each camera. Due to different fields of view, it causes potentially large exposure differences, which becomes a visual annoyance when the images are stitched together. That can, for example, be a major problem if some cameras contain areas that are particularly bright or dark, e.g., strong sun, bright snow, or strong shadows. Examples of some of the experienced visual effects due to different exposure settings in the camera array are shown in Fig. 20.7a and b.

When doing any form of image mosaicking, color correction generally needs to be applied on all individual source images. In an earlier version of Bagadus, this operation was part of the panorama processing pipeline [11]. However, we later adapted an approach where Bagadus controls the image exposure by broadcasting one identical configuration to all cameras [54]. For this to work, however, the cameras must have identical physical configurations to ensure that the cameras produce combinable results. Our setup with identical cameras, lenses, and capture software proved to work well with an identical exposure setting on every camera. This can be seen

(a) Panorama with independent auto exposure. There is snow around the field, and the metering system has to compensate for this and make a good choice of exposure values. Here, we have used a flat panorama with static (vertical) seams. One can clearly see the differences between the cameras

(b) Panorama with independent auto exposure. Here, we have used a cylindrical panorama with dynamic seams, and the differences between the cameras are again very visible

(c) Panorama with identical camera exposure values, i.e., all cameras use the same exposure setting. The visual improvement is huge, and the seams are already very hard to identify

Fig. 20.7 Panorama with independent auto-exposure versus identical exposure

in Fig. 20.7c, which shows a panorama generated from cameras with identical exposure settings. The individual auto-exposure sometimes produces a good result, e.g., the two rightmost images in Fig. 20.7b, but often, the automatic (non-synchronized) exposure results in very different exposure times. The effects of this is even greater on sunny days, as the differences increase.

Setting the exposure manually is, however, not always sufficient. The lighting conditions often change during a game due to the weather and the movement of the sun. We therefore present an approach where the time between two temporal frames

is exploited to communicate the exposures among the cameras where we achieve a perfectly synchronized array. An analysis of the system and some experimental results are presented. In summary, a pilot camera approach running in auto-exposure mode and then distributing the used exposure values to the other cameras seems to give best visual results.

20.4.1 Existing Exposure Approaches

When an array of multiple cameras produces images that are not captured using similar exposure parameters, there will be visual differences between adjacent camera images. Often, it can be solved by color correction approaches that handle the images post-recording [55]. Tian et al. [56] highlight challenges with color distortion in panoramic imaging and introduce a new approach for color correction. Another way for color matching is also proposed by Doutre and Nasiopoulos [57]. Xu et al. [58] provide a good performance evaluation of several color correction approaches. Xiong et al. [59] proposed an elegant color correction approach which applies a color transformation that is optimized over all the images to minimize drastic changes per image. Ibrahim et al. [60] provide an interesting approach for selecting the reference for color correction.

Nevertheless, even though color correction approaches can provide good results in panorama images, they can introduce artifacts like flicker and unnatural colors when it comes to the stitched videos. Furthermore, a better solution would be to attack the problem at the source rather than correcting for the error afterward; i.e., this problem can be handled even before the recording of the videos in a constrained space like a sports stadium.

20.4.2 The Bagadus Automatic Exposure Approaches

From our initial experiments and existing work in the field, we conclude that we need some kind of dynamic automatic exposure control synchronizing the camera settings across the camera array. In this respect, we use the cameras' internal metering mechanism to estimate the exposure parameters. The region of interest that is considered for metering can be modified for each camera. We make use of this functionality in the three exposure setting approaches described here (for a quick overview, see Table 20.1). Furthermore, we also present some experimental results showing the visual differences between the approaches and the importance of a synchronized exposure when generating panorama images or video frames. Finally, we present two other related approaches to improve the image quality, i.e., high dynamic range (HDR) and seam blending (see Table 20.1).

Table 20.1 Overview of the implemented automatic exposure approaches (independent metering, pairs metering, and pilot camera), high dynamic range, and seam blending in Bagadus

Approach	Description	Region selection	Visual output
Independent metering	Use a same colored, but not the exact same, area (the green grass) as a suited surface for metering to perform individual auto-exposure. The internal mechanism decides on a specific exposure for each individual camera	Manual	Fig. 20.7a, b
Pairs metering	As shown in Fig. 20.2b, adjacent cameras have an overlapping region of pixels which is used for metering to perform pairwise synchronized auto-exposure	Manual	Fig. 20.8
Pilot camera	One pilot camera functions in fully auto-exposure mode where the pilot camera's exposure parameters are transferred to all the other cameras	Automatic	Figs. 20.3d, 20.7c, 20.10
High dynamic range	Capturing the initial video frames with alternating exposure, switching between high (bright) and low (dark) exposure times which is combined using radiance and tone mappers	Manual	Fig. 20.14
Seam blending	A simple feathering approach where a weighted average between the two overlapping areas of the images is performed	–	Fig. 20.16

20.4.2.1 Independent Metering

The most trivial approach for an automatic exposure system is independent metering. Since our target space is a soccer stadium, we can use the green surface of the soccer field, which is a well-suited surface for metering, to evaluate the exposure parameters. Initially, a manual selection of metering region is selected per camera, and the cameras are configured to make an automatic exposure. The internal mechanism decides on a specific exposure value and gain to achieve a pre-defined gray value for the average of all the pixels from the metering region. An upper limit can be imposed on the exposure time to force the camera to use a higher gain in case of low light, instead of increasing the exposure time.

We already demonstrated the effects of such an approach in the beginning of this section where Fig. 20.7a depicts a scenario where snow is on the soccer field. The metering system has to compensate for this and make a good choice of exposure values. The influence of snow can also be observed in the independent metering

approach. Figure 20.7b shows another example using a different panorama generation approach also in a contrasting lighting setting with less snow. Again, we can observe differences between the cameras. The problem is that the exposures are different in each of the cameras, and even though each image is well exposed, they are not synchronized, i.e., introducing large visual differences in the generated panorama image.

20.4.2.2 Pairs Metering

This approach can be considered as an improvement of, but still a special case of, the independent metering presented above. In this approach, we exploit the fact that the adjacent cameras have an overlapping region. Therefore, camera pairs are formed which have defined regions of interest that point to the same physical space on the field. The selection of the regions of interest is performed manually to minimize the effect from the players or other elements on the field. Then, the cameras are run independently to perform automatic exposure, but metering based on the selected patches that are overlapped. Since the camera pairs are physically close to each other, the directional reflections will have minimum effect on the exposure. However, the first camera pair and the second pair are at a distance of 4 m from each other in this experiment (tested only on an early version of the pipeline [11]).

Figure 20.8 shows the pairs metering approach in one of the possible light conditions, i.e., when the sky is partially cloudy. In this approach, a clear difference can be seen at the center of the field due to the pairwise exposure settings. However, the left and the right camera pairs (using the same region of interest for metering) of the panorama are perfectly seamless. Nevertheless, our goal is to have a perfect setting for the entire panorama, and this approach was therefore early abandoned.

20.4.2.3 Pilot Camera

The pairs metering approach shows the potential of synchronizing the camera setting, but we need an automatic solution that is camera array wide. The idea here is therefore to use a pilot camera that functions in auto-exposure mode where the pilot camera's exposure parameters are transferred to the other cameras. Here, let the m

Fig. 20.8 Panorama generated using the pairs metering approach under a partially cloudy sky

cameras be named C_j, where $j \in [1, m]$, and C_p be the pilot camera. Let e_j and g_j be the exposure time and gain of camera C_j. Then, given e_p and g_p from the pilot camera which operates in auto-exposure mode, we need to compute e_j and g_j for the rest of the cameras. Furthermore, let T_j be the transformation function from the pilot camera to camera C_j. Then,

$$(e_j, g_j) = T_j(e_p, g_p). \tag{20.1}$$

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

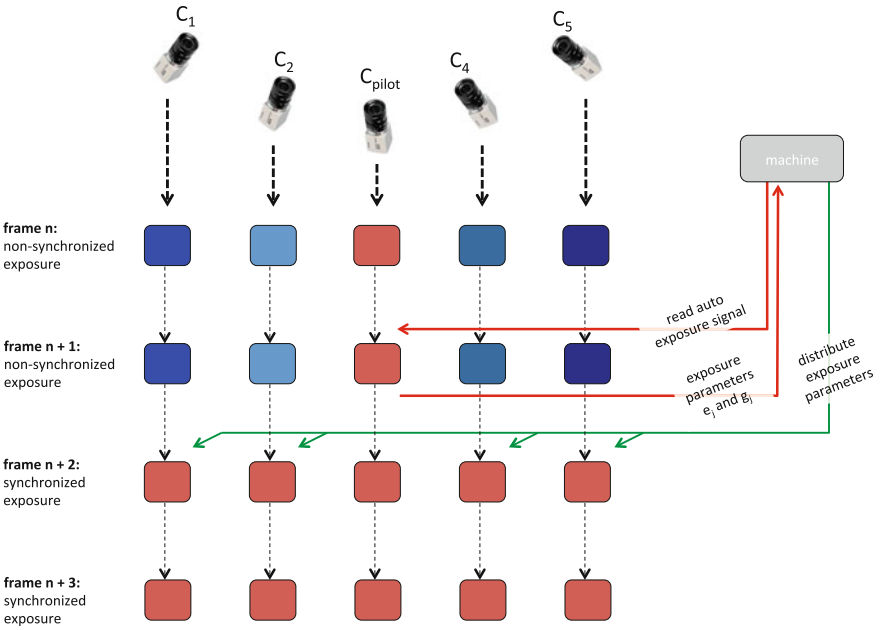


Fig. 20.9 The pilot camera approach. For frames n and $n + 1$, the colors of the frames indicate different exposure values (e_j and g_j). Camera C_{pilot} gets a signal from the controlling machine to read the auto-exposure values for frame $n + 1$. The values are sent back to the controlling machine, which again broadcasts the exposure values to the other cameras. Once received by all cameras, they all use the same exposure

The general processing flow is shown in Fig. 20.9. There are two types of threads that are running concurrently: one for controlling and communicating with the pilot camera and the others for the other cameras. All threads have a synchronization barrier at the end of every frame. Periodically, the pilot camera thread sends a trigger to the pilot camera to make an auto-exposure signal and lock the exposure until the next trigger. In Fig. 20.9, this can be seen before acquisition of frame n. After the exposure, the exposure parameters e_p and g_p are transferred back to the controlling machine. These parameters are communicated to other threads which in turn transfer these individually to the other cameras applying the appropriate transformation.

It can be observed that the frames n of the other cameras are not synchronized in exposure with the pilot camera, but we have observed empirically that the light conditions change slowly over the period of the exposure updating trigger. One more important detail is that the frame rate sets a hard upper bound on the execution time and thus on exposure time too. The formulation of transformation function cannot guarantee this because one of the transformations can demand a higher exposure time than the upper limit, especially when the cameras have lower response to light than the pilot camera. This problem can be handled in two ways. One way is to embed this property into the transformation function by placing an upper bound. The other way is to handle it in the driver before setting the camera parameters. We experienced that the driver solution is safer and more robust to further changes in the algorithm.

Figure 20.10 shows the pilot camera approach when there is an overcast sky, and using the flat panorama pipeline. Here, it can be observed from the figure that the exposure in the whole of the panorama is perfectly synchronized as there are no visual differences between the different parts in the stitched image. There is no specific color correction applied when stitching the panorama. Similar results can be observed for the later cylindrical panorama in Fig. 20.11c where the colors are the same and the stitches are hardly visible.

We therefore allow a single pilot camera to use automatic exposure. Then, at a given interval, for example, every ten seconds, we read this camera's automatic exposure values and broadcast these values to all the other cameras. This is done asynchronously by sending an exposure event message to the system server, i.e., the processing machine, which further broadcasts the message to all connected camera modules over the Transmission Control Protocol (TCP). This is not a time critical operation and can be done through a best effort approach on each individual camera stream. Figure 20.11 shows an extreme case, where each camera asynchronously receives the first exposure update. Once the recording is up and running, there are typically only minor changes on each update.

Fig. 20.10 Panorama generated using the pilot camera approach under an overcast sky

(a) Initial metering (frame k)

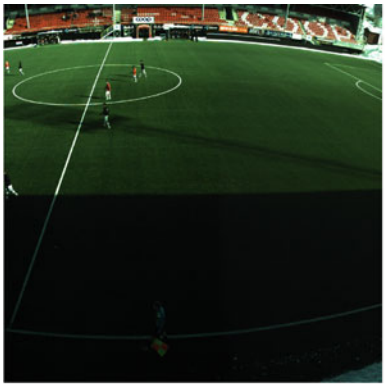
(b) The exposure update did not reach all cameras (frame $k+1$)

(c) All cameras' exposures are updated (frame $k+2$)

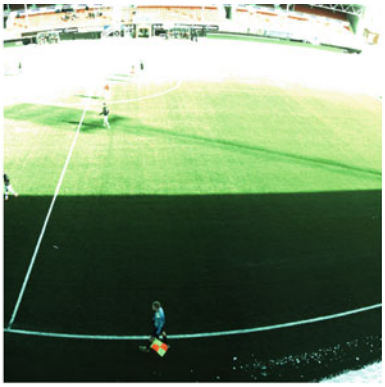
Fig. 20.11 Panoramas during an exposure synchronization. Initially, the frames are dark, and we here display how the exposure for different cameras changes during the metering process. Three frames are captured 20 ms apart at 50 fps. Usually, updated exposure settings are received by all machines between two frames

20.4.2.4 High Dynamic Range

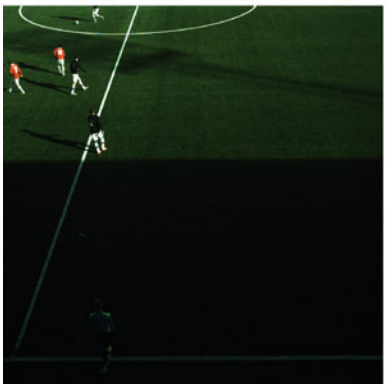
In Bagadus, as mentioned above, we also have support for HDR in order to deal with the challenging lighting conditions of a large outside environment in the presence of strong sun. This mechanism is achieved by capturing the initial video frames with alternating exposure, switching between high (bright) and low (dark) exposure times. However, in the case of exposure synchronization, the inclusion of the HDR module complicates the operation as the cameras must alternately switch between high and low synchronized exposure values. In order to generate a usable panorama, the two exposure times to use must be selected with care. Figure 20.12 shows an example output, with the low and high exposure images stitched together (displaying only parts of the entire panorama). Note how the two images complement each other by



(a) Low exposure set



(b) High exposure set



(c) Low exposure set (cropped)

(d) High exposure set (cropped)

Fig. 20.12 Sample exposure sets, used by the HDR module, showing the low and high exposure frames. Note that these have been stitched for visual purposes, though in the actual module, they are computed on the raw input image set, before stitching

showing details in different regions. This is vital to the success of the HDR process. By selecting exposure values that are too close together, we see less gain from the HDR module, and we may see a lot of noise in dark areas, as these are boosted.

This is especially challenging as we still wish to utilize dynamic exposures, setting automatic exposure on a single camera and allowing the other streams to duplicate the configuration. Setting automatic exposure in our system takes two or three frames when operating at 50 fps before the values are successfully updated. In this period, we decide to drop the frames causing a small hiccup of 50 ms. However, we experienced that the automatic exposure updates can be fairly infrequent; i.e., every 10 or 20 s is sufficient.

The actual low and high exposure times can be determined in several ways. One approach is to use two regions of interest: one positioned in the sunlight and one positioned in the shadow. This works well, although the resulting exposure values tend to be too far apart for the HDR module to handle it well. However, we learned that this was difficult to do in real-time updates, because changing this region of interest in the camera is extremely slow. Another approach is to perform auto-exposure with a single region of interest, preferably solely in the sun as this is more sensitive to changing light conditions, and then set the other exposure time as a static offset or a static percentage increase/decrease. This can work well in many situations, but we do not necessarily know the difference between the two regions, as this scales with brightness of the scene. We could also set a few static values and use whichever is closest based on the result of the auto-exposure.

We chose to use a mixture of these two approaches, by first defining the two regions as shown in Fig. 20.13. Then, we spend a few seconds at the beginning of the recording to estimate the optimal exposure times in each of these two regions. This

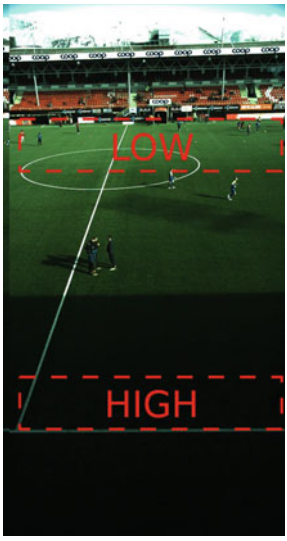


Fig. 20.13 Regions of interest used for determining auto-exposure for HDR

gives us the relative difference between the two regions. This difference is typically way too large, and we adjust the high exposure time E_{high} by:

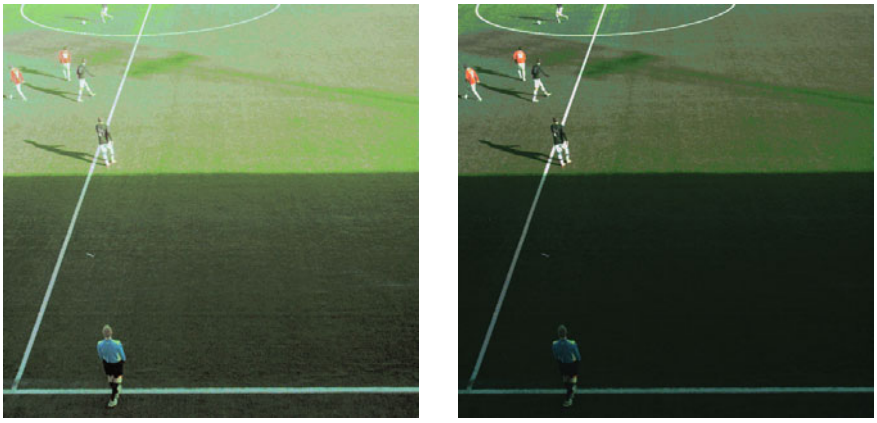
$$E_{high} = E_{low} + \frac{E_{high} - E_{low}}{2} \tag{20.2}$$

Then, we use the low exposure region of interest, i.e., sunny region, to determine the automatic exposure of E_{low} throughout the recording, updated as aforementioned. The high exposure time is determined by:

$$E_{high} = E_{high} + \frac{(E_{updated} - E_{low}) \times E_{low}}{E_{high}} \tag{20.3}$$

where $E_{updated}$ is the new low exposure time. This way, we maintain a correlation between the two exposure times. This assumes that the initial correlation was accurate, however, which may not be the case if the weather conditions move from cloudy to sunny during the recording.

Figure 20.14 shows example outputs of the HDR module using the images in Fig. 20.12c and d as input. There are multiple ways to combine the low and high exposure frames (see more examples in [13]), but here we have used the Debevec radiance mapper [61] combined with the Larson [62] and Reinhard [63] tone mappers. We believe that using the Debevec radiance mapper paired with the Larson tone mapper is a good solution taking the visual quality and the execution overhead into account. Although Reinhard's tone mapper also shows promising results, we could not make it pass the real-time requirements, and it also consumed a lot of memory on the GPU, thus affecting other components of the pipeline.



(a) Debevec - Larson **(b)** Debevec - Reinhard

Fig. 20.14 Visual quality after HDR using different radiance and tone mapping algorithms using the input images in Fig. 20.12c and d

20.4.2.5 Seam Blending

Even though we believe the pilot camera exposure synchronization approach is sufficient, we can occasionally, primarily in very bright weather, see some vignetting effects from the camera, i.e., a reduced brightness near the edges of the image compared to the center. An example with an artificially enhanced effect for the illustration is shown in Fig. 20.15a.

A frequently used solution is seam blending. Traditional solutions include Gaussian filtering and pyramid blending [64]. These are, however, too computational expensive for our real-time pipeline. Therefore, we use the simple feathering approach that performs a weighted average between the two overlapping images:

$$OutPixel(x, y) = \frac{i}{i_{max}} \times img_a(x, y) + \left(1 - \frac{i}{i_{max}}\right) \times img_b(x, y)$$

where i_{max} represents the size of the blending region and i the index within the region. Furthermore, we also perform a selective blend, where we ignore the pixels that are widely different, we filter out pixels that have a high edge cost in the stitch, and if not enough samples are found, we use the non-blended seam (more details are found in [65]). Finally, we experienced that it sufficient to perform the blending only in the luminosity component, as we primarily want to focus on removing very slight differences in exposure in the two images. Thus, the blending is performed by finding the average difference in luminosity between all the homogeneous pixels within a blending region of 40×20 pixels, around the original seam. The final result is visible in Fig. 20.16 where we can observe a panorama frame from a recording process,

(a) Stitch without blending **(b)** Stitch with blending

Fig. 20.15 Example of two images with large color differences, showing the effect of performing a post-stitch blending. Note that in the system large differences are very rare, and the images presented have been modified to increase the lighting difference for the illustration

Fig. 20.16 Panorama with pilot camera exposure synchronization and seam blending

implemented and running on a GPU using the compute unified device architecture (CUDA), with pilot camera exposure synchronization and stitch blending.

20.5 Conclusions

In our Bagadus system, the users expect a nice visual experience using the generated videos for sport analysis. In order to generate the high-resolution panorama covering the entire field-of-view of a soccer stadium, we have presented a camera array recording system which requires frame-by-frame synchronization, both with respect to shutter synchrony and camera exposure. The shutter synchrony is required to avoid ghosting effects, and this is solved using a hardware trigger where our trigger box sends shutter trigger signals at 1/fps using its local clock, and the trigger box itself is synchronized with the controlling machine to avoid clock drift. The camera exposure synchronization is required to minimize the visibility of the seams when stitching the individual frames into a full field-of-view panorama video. Our solution to this problem is to use a pilot camera that operates in fully auto-exposure mode, and its captured exposure parameters are transferred to and used by all the other cameras. We presented different approaches, and as shown in Figs. 20.3c and 20.16 for Alfheim stadium and Fig. 20.3d for Ullevaal stadium, the visual output of the proposed solutions is good.

Acknowledgements This work has been performed in the context of the iAD Centre for Research-based Innovation (project number 174867), and it is also supported in part by the EONS project (project number 231687)—both funded by the Research Council of Norway. Furthermore, there are numerous students and researchers that have worked on Bagadus or post-Bagadus solutions. For the synchronization of cameras, the authors want to acknowledge in alphabetical order: Alexander Eichhorn, Martin Stensgård, and Simen Sægrov.

References

1. Halvorsen, P., Sægrov, S., Mortensen, A., Kristensen, D.K., Eichhorn, A., Stenhaug, M., Dahl, S., Stensland, H.K., Gaddam, V.R., Griwodz, C., Johansen, D.: Bagadus: An integrated system for arena sports analytics a soccer case study. In: Proceedings of ACM MMSys, pp. 48–59 (2013)
2. Sægrov, S., Eichhorn, A., Emerslund, J., Stensland, H.K., Griwodz, C., Johansen, D., Halvorsen, P.: Bagadus: an integrated system for soccer analysis (demo). In: Proceedings of ICDSC, pp. 1–2 (2012)
3. Stensland, H.K., Gaddam, V.R., Tennøe, M., Helgedagsrud, E., Næss, M., Alstad, H.K., Mortensen, A., Langseth, R., Ljødal, S., Landsverk, Ø., Griwodz, C., Halvorsen, P., Stenhaug, M., Johansen, D.: Bagadus: an integrated real-time system for soccer analytics. ACM TOMCCAP **10**(1s) (2014)
4. Mortensen, A., Gaddam, V.R., Stensland, H.K., Griwodz, C., Johansen, D., Halvorsen, P.: Automatic event extraction and video summaries from soccer games. In: Proceedings of ACM MMSys, pp. 176–179 (2014)
5. ChyronHego: ZXY Sport Tracking. http://www.zxy.no/
6. Mills, D., Martin, J., Burbank, J., Kasch, W.: Network time protocol version 4: protocol and algorithms specification. RFC 5905 (Proposed Standard) (2010)
7. Johansen, D., Stenhaug, M., Hansen, R.B.A., Christensen, A., Høgmo, P.M.: Muithu: smaller footprint, potentially larger imprint. In: Proceedings of IEEE ICDIM, pp. 205–214 (2012)
8. Basler aca2000-50gc. http://www.baslerweb.com/products/ace.html?model=173
9. Azure-0814m5m. http://www.azurephotonicsus.com/products/azure-0814M5M.html
10. Stensland, H.K., Gaddam, V.R., Tennøe, M., Helgedagsrud, E., Næss, M., Alstad, H.K., Griwodz, C., Halvorsen, P., Johansen, D.: Processing panorama video in real-time. Int. J. Semant. Comput. **8**(2) (2014)
11. Tennøe, M., Helgedagsrud, E., Næss, M., Alstad, H.K., Stensland, H.K., Gaddam, V.R., Johansen, D., Griwodz, C., Halvorsen, P.: Efficient implementation and processing of a real-time panorama video pipeline. In: Proceedings of IEEE ISM (2013)
12. Langseth, R., Gaddam, V.R., Stensland, H.K., Griwodz, C., Halvorsen, P., Johansen, D.: An experimental evaluation of debayering algorithms on gpus for recording panoramic video in real-time. Int. J. Multimedia Data Eng. Manag. **6**(6) (2015)
13. Kellerer, L., Gaddam, V.R., Langseth, R., Stensland, H.K., Griwodz, C., Johansen, D., Halvorsen, P.: Real-time HDR panorama video. In: Proceedings of ACM MM, pp. 1205–1208 (2014)
14. Stensland, H.K., Wilhelmsen, M.A., Gaddam, V.R., Mortensen, A., Langseth, R., Griwodz, C., Halvorsen, P.: Using a commodity hardware video encoder for interactive applications. Int. J. Multimedia Data Eng. Manag. **6**(3), 17–31 (2015)
15. Wilhelmsen, M.A., Stensland, H.K., Gaddam, V.R., Mortensen, A., Langseth, R., Griwodz, C., Johansen, D., Halvorsen, P.: Using a commodity hardware video encoder for interactive video streaming. In: Proceedings of IEEE ISM (2014)
16. VideoLAN: x264. http://www.videolan.org/developers/x264.html
17. Gaddam, V.R., Langseth, R., Ljødal, S., Gurdjos, P., Charvillat, V., Griwodz, C., Halvorsen, P.: Interactive zoom and panning from live panoramic video. In: Proceedings of ACM NOSSDAV, pp. 19–24 (2014)
18. Gaddam, V.R., Bao Ngo, H., Langseth, R., Griwodz, C., Johansen, D., Halvorsen, P.: Tiling of panorama video for interactive virtual cameras: overheads and potential bandwidth requirement. In: Proceedings of IEEE PV, pp. 204–209 (2015)
19. Gaddam, V.R., Riegler, M., Eg, R., Griwodz, C., Halvorsen, P.: Tiling in interactive panoramic video: approaches and evaluation. IEEE Trans. Multimedia **18**(9), 1819–1831 (2016)
20. Niamut, O.A., Thomas, E., D'Acunto, L., Concolato, C., Denoual, F., Lim, S.Y.: Mpeg dash srd: Spatial relationship description. In: Proceedings of MMSys (2016)
21. Sanchez, Y., Skupin, R., Schierl, T.: Compressed domain video processing for tile based panoramic streaming using hevc. In: Proceedings of IEEE ICIP, pp. 2244–2248 (2015)

22. NTP.org: NTP faq—How does it work? http://www.ntp.org/ntpfaq/NTP-s-algo.htm
23. Wikipedia: 100 metres. https://en.wikipedia.org/wiki/100_metres
24. Quora: How fast can a soccer ball be kicked? https://www.quora.com/How-fast-can-a-soccer-ball-be-kicked
25. Hasler, N., Rosenhahn, B., Thormahlen, T., Wand, M., Gall, J., Seidel, H.P.: Markerless motion capture with unsynchronized moving cameras. In: Proceedings of IEEE CVPR, pp. 224–231 (2009)
26. Pourcelot, P., Audigié, F., Degueurce, C., Geiger, D., Denoix, J.M.: A method to synchronise cameras using the direct linear transformation technique **33**(12), 1751–1754 (2000)
27. Shrestha, P., Barbieri, M., Weda, H., Sekulovski, D.: Synchronization of multiple camera videos using audio-visual features. IEEE Trans. Multimedia **12**(1), 79–92 (2010)
28. Shrestha, P., Weda, H., Barbieri, M., Sekulovski, D.: Synchronization of multiple video recordings based on still camera flashes. In: Proceedings ACM MM. New York, USA (2006)
29. Ruttle, J., Manzke, M., Prazak, M., Dahyot, R.: Synchronized real-time multi-sensor motion capture system. In: Proceedings of ACM SIGGRAPH ASIA (2009)
30. Bradley, D., Atcheson, B., Ihrke, I., Heidrich, W.: Synchronization and rolling shutter compensation for consumer video camera arrays. In: Proceedings of IEEE CVPR, pp. 1–8 (2009)
31. Duckworth, T., Roberts, D.J.: Camera image synchronisation in multiple camera real-time 3D reconstruction of moving humans. In: Proceedings of DS-RT, pp. 138–144 (2011)
32. Moore, C., Duckworth, T., Aspin, R., Roberts, D.: Synchronization of images from multiple cameras to reconstruct a moving human. In: Proceedings of IEEE/ACM DR-RT, pp. 53–60 (2010)
33. Chang, R., Ieng, S., Benosman, R.: Shapes to synchronize camera networks. In: Proceedings of IEEE ICPR, pp. 1–4 (2008)
34. Sinha, S., Pollefeys, M.: Synchronization and calibration of camera networks from silhouettes. In: Proceedings of ICPR, vol. 1, pp. 116–119 (2004)
35. Sinha, S.N., Pollefeys, M.: Camera network calibration and synchronization from silhouettes in archived video. Int. J. Comput. Vis. **87**(3), 266–283 (2010)
36. Topçu, O., Ercan, A.Ö., Alatan, A.A.: Recovery of temporal synchronization error through online 3D tracking with two cameras. In: Proceedings of ICDSC, pp. 1–6 (2014)
37. Haufmann, T.A., Brodtkorb, A.R., Berge, A., Kim, A.: Real-time online camera synchronization for volume carving on GPU. In: Proceedings of AVSS, pp. 288–293 (2013)
38. Nischt, M., Swaminathan, R.: Self-calibration of asynchronized camera networks. In: Proceedings of ICCV Workshops, pp. 2164–2171 (2009)
39. Shankar, S., Lasenby, J., Kokaram, A.: Synchronization of user-generated videos through trajectory correspondence and a refinement procedure. In: Proceedings of CVMP, pp. 1–10 (2013)
40. Shankar, S., Lasenby, J., Kokaram, A.: Warping trajectories for video synchronization. In: Proceedings of ARTEMIS, pp. 41–48 (2013)
41. Tao, J., Risse, B., Jiang, X., Klette, R.: 3D trajectory estimation of simulated fruit flies. In: Proceedings of IVCNZ (2012)
42. Velipasalar, S., Wolf, W.H.: Frame-level temporal calibration of video sequences from unsynchronized cameras. Mach. Vis. Appl. **19**(5–6), 395–409 (2008)
43. Whitehead, A., Laganiere, R., Bose, P.: Temporal synchronization of video sequences in theory and in practice. In: Proceedings of IEEE WACV/MOTION, pp. 132–137 (2005)
44. Kovacs, J.: AN005 Application Note—An Overview of Genlock. http://www.mivs.com/old-site/documents/appnotes/an005.html
45. Smith, S.L.: Application of high-speed videography in sports analysis. In: Proceedings of SPIE 1757 (1993)
46. Collins, R.T., Amidi, O., Kanade, T.: An active camera system for acquiring multi-view video. In: Proceedings of ICIP, pp. 520–527 (2002)
47. Lin, M.Y.: Shutter synchronization circuit for stereoscopic systems (1998). https://www.google.com/patents/US5808588

48. Gross, M., Lang, S., Strehlke, K., Moere, A.V., Staadt, O., Würmlin, S., Naef, M., Lamboray, E., Spagno, C., Kunz, A., Koller-Meier, E., Svoboda, T., Van Gool, L.: Blue-c: a spatially immersive display and 3D video portal for telepresence. In: Proceedings of ACM SIGGRAPH (2003)
49. Wilburn, B., Joshi, N., Vaish, V., Levoy, M., Horowitz, M.: High-speed videography using a dense camera array. Proc. IEEE CVPR **2**, 294–301 (2004)
50. Meyer, F., Bahr, A., Lochmatter, T., Borrani, F.: Wireless GPS-based phase-locked synchronization system for outdoor environment. J. Biomech. **45**(1), 188–190 (2012)
51. Litos, G., Zabulis, X., Triantafyllidis, G.: Synchronous image acquisition based on network synchronization. In: Proceedings of CVPR Workshops (3DCINE), pp. 167–167
52. Sousa, R.M., Wäny, M., Santos, P., Dias, M.: Multi-camera synchronization core implemented on USB3 based FPGA platform. In: Proceedings of SPIE 9403 (2015)
53. Nguyen, H., Nguyen, D., Wang, Z., Kieu, H., Le, M.: Real-time, high-accuracy 3D imaging and shape measurement. Appl. Opt. **54**(1), A9 (2015)
54. Gaddam, V.R., Griwodz, C., Halvorsen, P.: Automatic exposure for panoramic systems in uncontrolled lighting conditions: a football stadium case study. In: Proceedings of SPIE 9012—The Engineering Reality of Virtual Reality (2014)
55. Hasler, D., Ssstrunk, S.: Mapping colour in image stitching applications. J. Visual Commun. Im. Represent. **15**(1), 65–90 (2004)
56. Tian, G.Y., Gledhill, D., Taylor, D., Clarke, D.: Colour correction for panoramic imaging. In: Proceedings of IV (2002)
57. Doutre, C., Nasiopoulos, P.: Fast vignetting correction and color matching for panoramic image stitching. In: Proceedings of ICIP, pp. 709–712 (2009)
58. Xu, W., Mulligan, J.: Performance evaluation of color correction approaches for automatic multi-view image and video stitching. In: Proceedings of IEEE CVPR, pp. 263–270 (2010)
59. Xiong, Y., Pulli, K.: Color correction for mobile panorama imaging. In: Proceedings of ICIMCS, pp. 219–226 (2009)
60. Ibrahim, M., Hafiz, R., Khan, M., Cho, Y., Cha, J.: Automatic reference selection for parametric color correction schemes for panoramic video stitching. Adv. Visual Comput. Lect. Notes Comput. Sci. **7431**, 492–501 (2012)
61. Debevec, P.E., Malik, J.: Recovering high dynamic range radiance maps from photographs. In: Proceedings of SIGGRAPH, pp. 369–378 (1997)
62. Larson, G.W., Rushmeier, H., Piatko, C.: A visibility matching tone reproduction operator for high dynamic range scenes. IEEE Trans. Visual. Comput. Graph. **3**(4), 291–306 (1997)
63. Reinhard, E., Stark, M., Shirley, P., Ferwerda, J.: Photographic tone reproduction for digital images. ACM Trans. Graph. **21**(3), 267–276 (2002)
64. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Prentice-Hall, Inc. (2006)
65. Langseth, R.: Implementation of a distributed real-time video panorama pipeline for creating high quality virtual views. University of Oslo (2014)