Portable Multi-Megapixel Camera with Real-Time Recording and Playback

Peter Carr Richard Hartley
Australian National University and NICTA
Canberra, Australia

Abstract—**We are interested in the problem of automatically tracking football players, subject to the constraint that only one vantage point is available. Tracking algorithms benefit from seeing the entire playing field, as one does not have to worry about objects entering and leaving the field of view. However, the image of the entire field must be of sufficient resolution to allow each of the players to be identified automatically.**

To achieve this desired video data, several high definition video cameras are used to record a football match from a single vantage point. The cameras are oriented to cover the entire playing field, and their images combined to create a single high-resolution video feed. The user is able to pan and zoom in real-time within the unified video stream while it is playing. The system is achieved by distributing tasks across a network of computers and only processing data that will be visible to the user.

Index Terms—**mosaic; video; distributed; resolution; real-time**

I. Introduction

In this work, we describe the development of a camera system to be used for post-game analysis of football matches. Although much work has already been done in automated sports tracking, our approach focuses on a suitable mechanism for data collection and review. Furthermore, the design requirements and constraints for our solution were established through consultation with the Australian Institute of Sport, making our system practical for real-world use.

Existing commercial systems usually require an extensive installation phase, making them difficult to transfer from site to site. As a result, they are of limited use for teams which spend a significant amount of time away from their home stadium. Our solution, on the other hand, is designed to be highly portable. In addition to being small, light-weight and quick to set-up, we also minimize the amount of access required to the grounds, as visiting coaching staff are usually limited to a single vantage point from which to record the game at opposition stadiums.

Several sports analysis systems simplify the tracking problem (at least in terms of data collection) by focusing on the immediate action—i.e, either following the ball using pan-tilt-zoom cameras, or using a collection of fixed cameras for sports such as tennis [1], cricket [1] and baseball [2], where the majority of the action is confined to a relatively small location. However, for the recorded footage to be useful to the coaching staff, the video must contain all players (from both teams) at all times. Furthermore, having information on the locations of all players at all times should improve the robustness of any tracking algorithm. In addition, the video sequence needs to be of sufficient quality so that players can be recognized automatically, and that the intricate details of one-on-one player interactions can also be reviewed.

To obtain a video recording of the entire field with suitable image quality for both the coaching staff and automated analysis (such as tracking the positions of the players and ball), our work focuses on fusing data from multiple cameras into a single multi-megapixel representation (see Figure 1). The camera system also includes a specialized viewer to provide the coaching staff with an intuitive interface to explore the multi-megapixel video stream. During playback, the user is able to pan and zoom within the stitched video sequence. As a result, members of the coaching staff are able to observe the strategic developments of the game when viewing the entire field, as well as particular actions of an individual player or group of players when necessary. Real-time performance is achieved by distributing the playback load over a network of computers.

A. Related Work

ProZone [3] is a commercially available solution for analysis of football matches. It requires multiple cameras placed around the field, each manually controlled by an operator [4]. The analysis is exceptionally thorough, but takes a significant amount of time to produce, as it is only partially automated. Installing ProZone is a complex and expensive task, which means the system can only be used in the stadium in which it was set-up. Alternative systems, such as [5], [6], [7], also rely on multiple pan-tilt-zoom cameras for input and are not suitable for our circumstances.

A research group at KTH [4], [8], [9], [10] combined the images from four DV cameras (located at the same vantage point) to produce a single high-resolution image of the entire football field. However, their primary focus was on off-line tracking, and avoided an investigation into “affordable methods to view [the] video sequences at their maximum resolution” [4]. Our implementation, on the other hand, directly addresses the need to view the unified multi-megapixel video sequence.

Pintaric *et al.* [11] developed a 1.6 MPixel portable panoramic video system. Their five camera system captured a $360^\circ \times 72^\circ$ field of view, which was mapped to the inside of a virtual cylinder and viewed interactively using a specialized head-mounted display. The bandwidth requirement was minimized by only loading the data that would be visible in

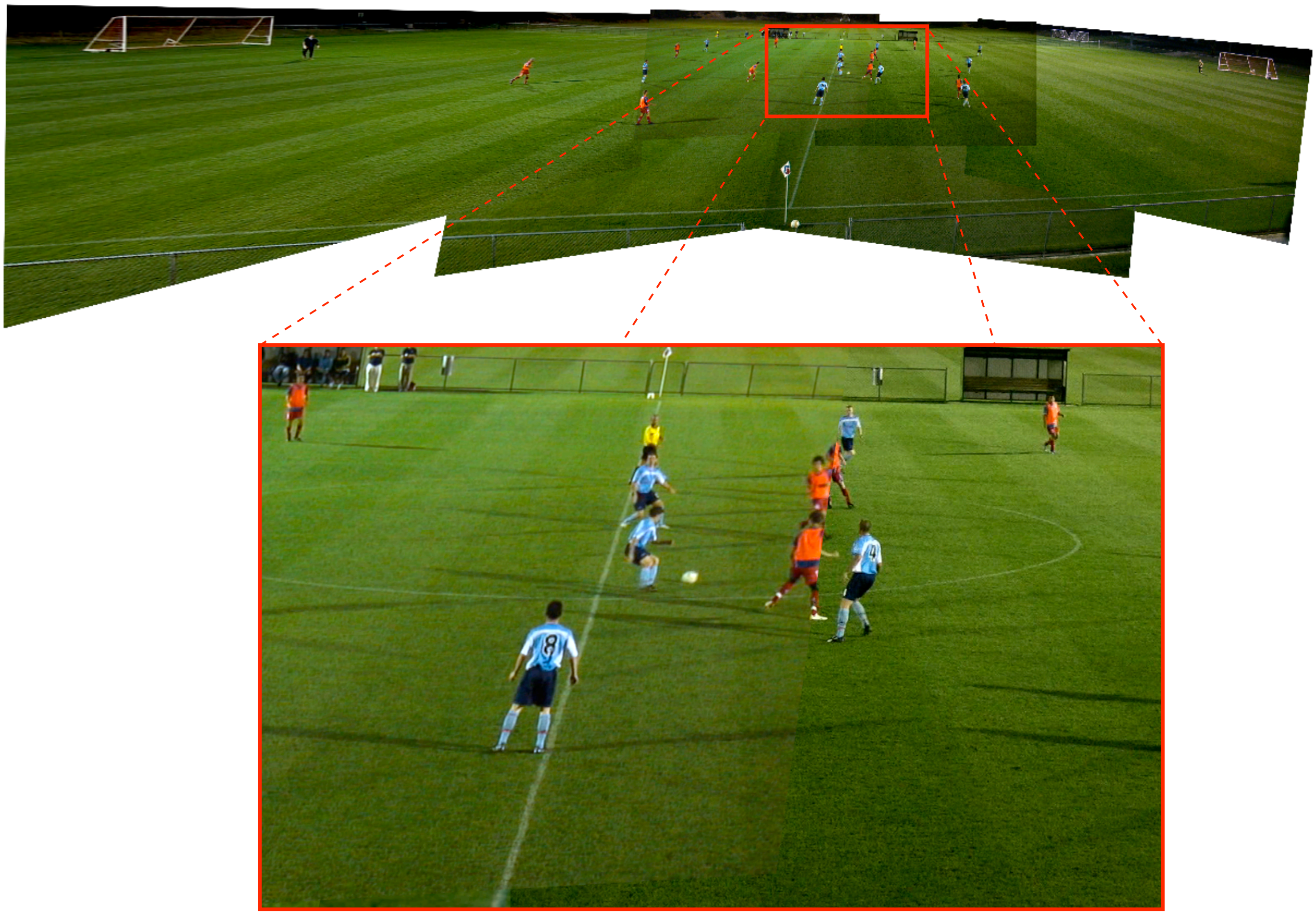

Fig. 1. *The individual camera images are aligned into a common co-ordinate frame producing a unified image of the field (top). The user can pan and zoom within the mosaiced video stream, effectively defining a region of interest within the canvas co-ordinate system (below).*

the user-defined region of interest. To achieve this, a post-processing stage was incorporated to stitch the individual video sequences together and then spatially segment the resulting mosaic into several smaller video streams.

Recent work at HP [12] has focused on constructing a mosaiced high resolution image from a collection of standard resolution cameras. However, custom FPGA hardware was used to transfer the contents of each video stream into a pre-allocated memory location within the single computer. Although the work mentions the need to specialize software for multiple cores, there is no mention of distributing tasks over multiple processors. Moreover, the concept of a scalable display/data access was identified as an area of future work.

Finally, the gigapixel image project at Microsoft Research [13] produces an extremely high-resolution composite image from many multi-megapixel images, captured from a specialized tripod. The system employs a similar panning and zooming strategy to allow the user to switch between the context of the full scene and arbitrary regions where intricate details are not visible in the larger view due to down-sampling. In their implementation, a significant amount of pre-processing was performed to stitch the individual images into a single mosaic. In addition, image pyramids were used to reduce the computational and bandwidth requirements of the computer running the viewer application. Although we also aim to provide fluid responses to panning and zooming actions, the methods of [13] are presently too computationally intensive for video.

II. Design

When travelling to other stadiums, the team's technical staff are usually only guaranteed access to a single vantage point. Although different perspectives would help overcome occlusions, a multi-perpsective system is prohibited by our design specifications. To compensate for the monocular limitation, we aim to record the game at a high image quality.

Our solution uses multiple static cameras located at a single vantage point, and produces a unified image of the entire field by registering each perspective to a common *canvas* co-ordinate system (see Figure 1). In addition, our interface allows the user to pan and zoom within the mosaiced video stream — effectively defining a region of interest (ROI). Such a design allows the user to switch rapidly between monitoring the entire

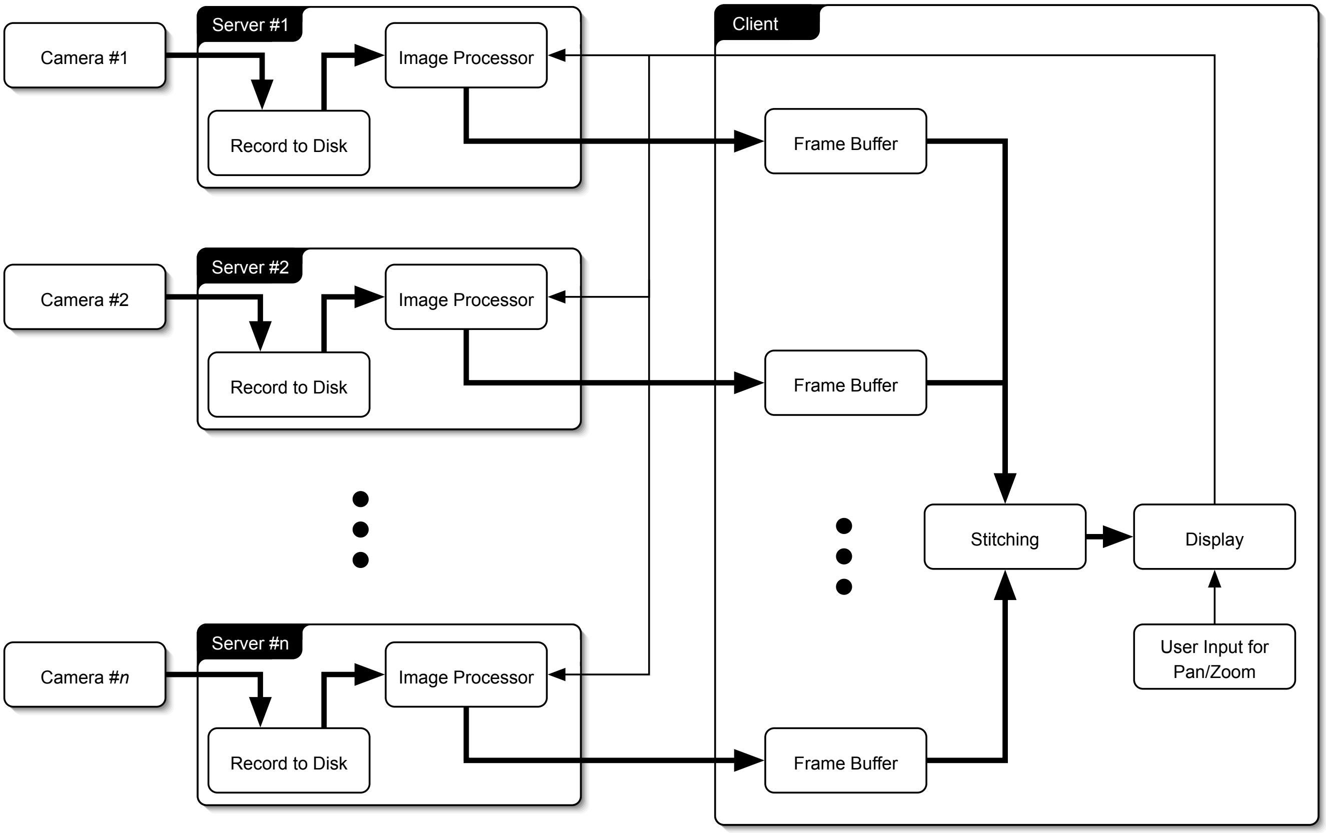

Fig. 2. *Each camera is augmented with a dedicated computer to record the captured frames, as well as managing playback bandwidth. The 'image processing' module crops and resamples each frame (with parameter values depending on the current user-defined region of interest on the canvas). Dark lines represent the flow of video data, and lighter lines represent control signals.*

field and/or individuals or groups of players. The recording and playback load is distributed over a network of computers (see Figure 2) and controlled from a single machine.

Like Pintaric *et al.* [11], our implementation makes efficient use of bandwidth by only transmitting the video streams which are visible in the user-defined region of interest. Whereas Pintaric *et al.* incorporated a post-processing step to sub-divide the video stream into spatial blocks, we incorporate a computer to manage each video stream, and crop the full frame images in real time. The advantage of this method is that is allows for greater bandwidth efficiency, as the required image sub-region to transmit is determined to the nearest pixel, and not the nearest block.

As the user-defined region of interest grows to encompass a wider view of the field, the resolution at which each camera image is displayed on the screen decreases. Kopf *et al.* [13] also handle changing region of interest scales, but their methodology is quite different from ours. Since they are dealing with a single image, a reasonable amount of off-line pre-processing can be invested to accelerate the on-line experience. For instance, their system pre-computes tiled image pyramids, and the viewing application only extracts pixels from the appropriate pyramid image level when generating the region of interest image. We, however, avoid image pyramids and instead dynamically resample the cropped video frame to match the desired display resolution.

A significant feature of our design is that the required bandwidth is primarily dependent on the output screen resolution (1440 × 900 in our case) — not the number of cameras. Although bandwidth requirements will grow as image overlap increases, our design easily incorporates additional cameras without requiring significant changes to the underlying infrastructure.

III. Hardware Implementation

A regular DV camera (720×480 pixels) was deemed to have sufficient image quality if the operator were allowed to pan, tilt and zoom the camera to follow a player. An HD camera (1920×1080 pixels) with a wide lens covering approximately 80% of the field (essentially observing all players except for the two goaltenders) provided suitable resolution for the coaching staff, but lacked sufficient resolution to estimate the identity of a player using standard computer vision methods (from either biometric features or the number on their jersey [14]). Our approach combines the data from six static HD cameras arranged to record the enitre field at a relatively

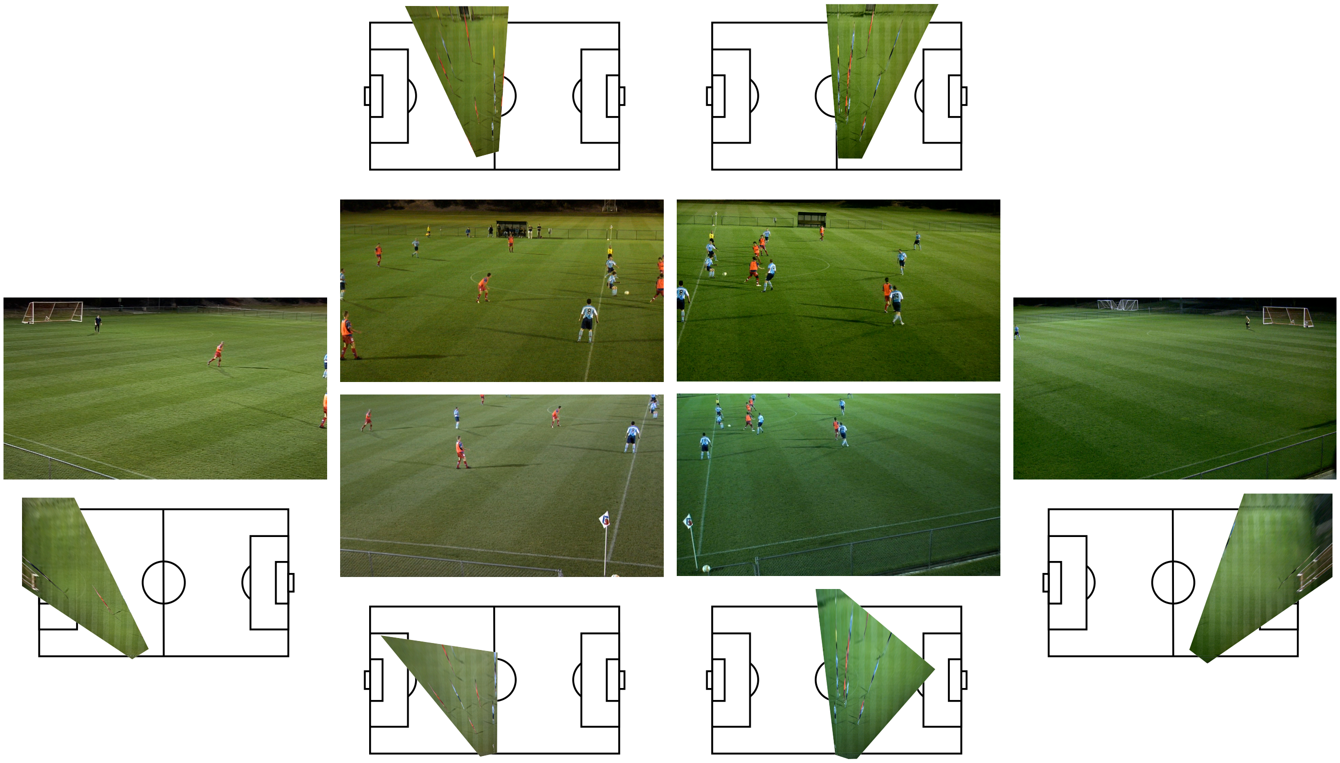

Fig. 3. *Six cameras are used to record the game from the same vantage point. Camera poses were chosen to produce consistent spatial resolution across the playing surface. Each pose is illustrated by projecting the resulting camera frustum onto a model of the playing surface.*

consistent spatial resolution (see Figure 3). Two cameras with short lenses are used to cover the front-half of the field, with the far side covered by four long lens cameras. The cameras are placed close together in an effort to minimize the distances between the centres of projection, which should reduce parallax artifacts when combining the different images.

HDV cameras are used to film the games, as they provide an appropriate compromise between image quality and cost. Although not true high-definition (the format is recorded in an anamorphic 1440×1080 format, and then stretched to 1920×1080 during playback [15]), the cameras still produce an image quality superior to regular DV at a minimal cost increase.

Although portability is an important aspect of the design requirements, our implementation also needs to operate within a reasonably quick-turnaround time. To avoid the lengthy offline process of transferring taped video to disk, we record directly to computer hard drives. We maintain portability by using Apple's Mac Mini[1] computers, as they are quite small, cheap and contain sufficient processing power for our recording and playback needs. A standard Gigabit ethernet switch is used to connect the computers together.

Finally, an Apple MacBook Pro[2] with a screen resolution of 1440×900 is used as the central controlling machine, and each Mac Mini is configured for remote administration. To enable OpenGL acceleration, dummy monitor connectors are attached to each Mac Mini.

IV. Software Implementation

Playback is managed through two separate software applications. The 'server' component is installed on each Mac Mini and controls access to each camera feed. The 'client' application is installed on the MacBook Pro, and is controlled by the user.

A. Image Processing

Image resampling is a major aspect of both applications' operation, and needs to be conducted efficiently to maximize system performance. As a result, our software implementation makes use of two accelerated image processing libraries included in OSX 10.4:

- *Core Image* [16] provides an API for common graphics processing unit (GPU) accelerated image operations, such as scaling, cropping and perspective transforms. Although the GPU is optimized for displaying results to the screen (as it has direct access to video memory), it is possible to leverage the GPU to accelerate image processing operations and transfer the results back to main memory [17]. However, there is usually a significant performance overhead obtaining the results from the GPU.
- Alternatively, *vImage* [18] implements vectorized versions of common image operations. Unlike *Core Image*,

[1] 2.0 GHz Intel Core 2 Duo, 2 GB RAM, Intel GMA 950

[2] 2.2 GHz Intel Core 2 Duo, 2 GB RAM, NVIDIA GeForce 8600M GT

vImage works in the same memory space as the CPU. Although there is still a minor overhead moving data to and from the registers in the vector processing unit (VPU), this is usually minimal relative to the time it takes to perform the actual image processing operations. Furthermore, the framework allows the image to be broken up into blocks and processed in parallel on multiple threads, if multiple processors and/or cores exist.

B. Server

Each Mac Mini is responsible for extracting a region of interest from its recorded video file and resampling it to the requested output resolution. Once this operation is complete, the data is transmitted to the central controlling computer (see Figure 4).

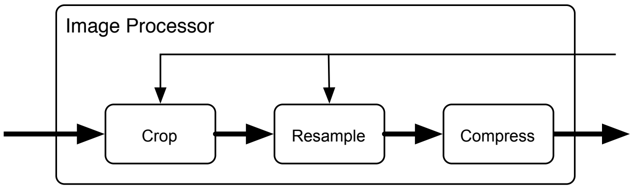

Fig. 4. Each server is responsible for transmitting the current video frame to the client. To conserve bandwidth, the portions of the image lying beyond the current region of interest in the canvas are cropped from the video frame. The resulting image is also resampled (downwards only) to match the resolution at which it will be displayed on the screen. Finally, the 24-bit RGB triplet is compressed into the 16-bit RGB 565 format.

Resampling: As mentioned previously, there are multiple ways to resample an image eficiently in OSX 10.4. Since the client application needs to send the resampled image over the network, GPU accelerated methods (if employed) will need to read the results back to main memory.

The integrated Intel GMA950 chipset on the Mac Mini (or the driver for it) appears to be optimised for synchronous data transfer between the GPU and CPU. Average throughput using OpenGL's synchronous transfer command *glReadPixels()* is approximately 40 MPixels/sec, whereas the equivalent asynchronous operation *glGetTexImage()* is only 7 MPixels/sec. Although *glReadPixels()* achieves greater performance, the operation blocks the current thread execution. To overcome this problem, we run the OpenGL operations on a separate dedicated thread.

Although the GPU may be faster for large images, the overhead to transfer data between video memory and main memory may make it slower than a VPU implementation for smaller images. To achieve maximum performance, we use both implementations and invoke the appropriate method depending on the current requested region of interest and video size (see Figure 5). The most intensive situation occurs when the entire 1920×1080 video frame needs to be resampled to 1440×810 to fill the screen of the central computer . Fortunately, the native anamorphic signal reduces the input to 1440×1080 (1.48 MPixels), and output to 1080×810 (0.83 MPixels), which takes approximately 30 ms on the Mac Minis.

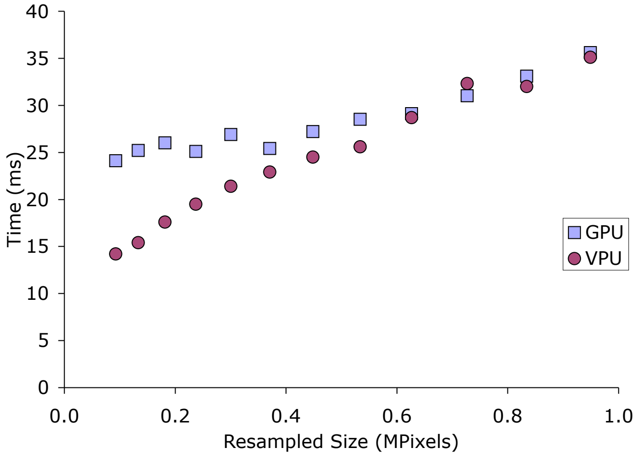

Fig. 5. *Performance resampling a 1.48 MPixel image using the VPU and GPU. The overhead of reading data from the GPU is evident for smaller image sizes.*

As Figure 5 illustrates, above 0.6 MPixels both methods have comparable performance. However, the VPU method consumes a significant portion of the total available computing resources. Off-loading the image processing tasks to the GPU, even if marginally slower, leaves the CPU available for other tasks (such as managing data transmission and responding to requests from the central controlling computer for new cropping and resampling settings).

Compression: Before the resampled image is transmitted, the 24 bit pixels are down-sampled to 16 bits (5 bits for red, 6 for green and 5 for blue) using the *vImage* framework, since the bandwidth of gigabit ethernet is insufficient to send more than 1.67 MPixels at 25 fps (and 3 bytes/pixel). In most cases, the cameras will overlap to some degree, which means that data for certain display pixels will be sent more than once (as the central computer decides which pixel data to display based on the user-defined layer order). As such, it is important to have sufficient bandwidth to transmit more than the final display screen resolution. Reducing to 16 bit pixels allows up to 2.5 MPixels to be transmitted for each frame.

In the worst case scenario, compressing a 1080×810 image, the Mac Mini takes approximately 5 ms to compress the image.

Recording: The HDV format encodes information in MPEG2 in order to maintain the same data rate as DV [15]. The downsides of this codec are that the decoding process is CPU intensive and temporal compression is employed. In our implementation, the user is given control over playback, and is able to jump to any frame in the image sequence (giving the coaching staff the ability to stop, rewind, play in slow motion, etc., for reviewing particular moments of the game). Therefore, we incorporate an off-line transcoding stage to convert to Apple's *intermediate codec* [15]. This codec provides a suitable trade-off between file size and decoding complexity, and does not incorporate temporal compression. As a result, it is well suited for playback from any point in a file, unlike MPEG2. The Mac Minis are able to perform the transcoding stage slightly faster than real-time ($\sim 0.9\times$).

C. Client

The client's two main tasks are to produce the appropriate portion of the unified view of the football field and update the transmission requirements of each video server when the user specifies a new region of interest (see Figure 7). Generating the consolidated view of the field requires aligning and colour correcting a set of synchronized frames from the individual video feeds (see Figure 1).

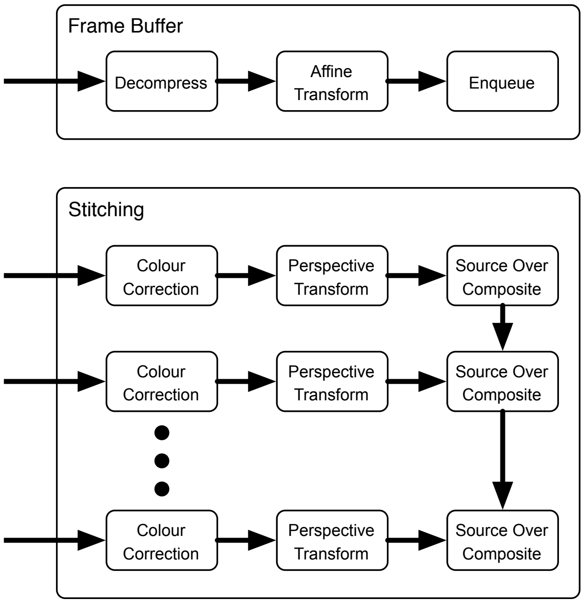

Fig. 7. The central controlling computer manages a frame buffer for each camera. As data is received, the 16-bit RGB 565 pixels are upsampled to 32-bit ARGB and transferred to the graphics processing unit. The pixels are then translated and scaled to occupy the region of the video frame from which they were originally extracted by the server. Finally, the frame is inserted into the appropriate position within the frame buffer.

At each screen refresh, the client extracts the appropriate image from each frame buffer. The images are colour corrected and warped using the estimated homographies. Finally, the composite image is produced by layering the images in a particular order (specified by the user during the set-up phase).

Homography Estimation: As video sources are discovered by the central machine, the user is prompted to assist in estimating the inter-camera homographies between the new camera and the previously registered ones. For a given pair of cameras, the controlling computer requests each camera to send its image in full resolution. The user is then asked to identify point correspondences on the ground planes in both images. When a suitable number of points have been defined, the homography is estimated using the direct linear transform [19]. Since no inter-camera homography can be established for the first camera, the co-ordinate system of that perspective is used to define the co-ordinate system of the canvas.

Compositing: The *Core Image* framework is leveraged to provide real-time compositing of the individual video feeds. At each screen refresh, the appropriate image from each frame buffer is colour corrected using an affine transformation [20] (with user defined parameters) and warped to its appropriate convex quadrilateral on the canvas using the previously described homography. The individual images are sequentially added to the canvas using the simple "source over" compositing method. Finally, the currently viewed region of the canvas is extracted and drawn to the screen.

Frame Buffer: In addition to the RGB 565 pixel data, each Mac Mini also encapsulates the region of interest (relative to the full camera frame) to which this pixel data belongs, as well as a local timestamp indicating when this frame should be displayed. The pixels are up-sampled to ARGB 8888 using *vImage*, and transferred to the GPU's video memory. The pixels are then scaled and translated using a GPU accelerated affine transformation to occupy the proper region of interest in the camera's local co-ordinate system. Finally, the frame is inserted into the appropriate location of the frame buffer (queued based on display time).

Synchronization: Since each camera is capturing at 25 fps, driven by its own internal clock, the timestamp assigned to each frame (essentially the frame number) will not be consistent across all cameras. Before playback can begin, the user is requested to specify the relative temporal offsets between the video files on each server. The client manages its own playback clock, which ensures the screen refreshes at 25 fps. The relative temporal offsets between the canvas clock and each video feed are then used to extract the appropriate frame from each frame buffer, as the frame buffers are temporally aligned to the original clock signal of the cameras.

Bandwidth Management: Given the user's current region of interest in the canvas, the client software calculates the sub-region visible in each camera by warping the bounds of the region of interest into the co-ordinate system of each camera using the inverse homography (see Figure 6). The bounding box of the resulting convex quadrilateral represents the required sub-region needed from that particular video source. The required display size of each camera is then approximated using the bounding box of the convex quadrilateral mapped back in the canvas co-ordinate system. As the user pans and zooms in the mosaiced image, the image requirements are revised and transmitted to the appropriate server.

V. Conclusions

We developed a prototoype camera system for post-game football analysis from a single vantage point. Our solution does not compromise on the design constraints of portability or cost. To achieve image quality suitable for both qualitative and quantitative analysis, our system fuses the data from multiple camera located at the same vantage point. Real-time playback is achieved by distributing the playback load across a network of computers. Along with being able to handle a variable number of cameras, our implementation provides an intuitive viewer to explore the multi-megapixel video stream. As a result, the coaching staff are able to monitor the development of the game over the entire field, as well as reviewing specific one-on-one player interactions.

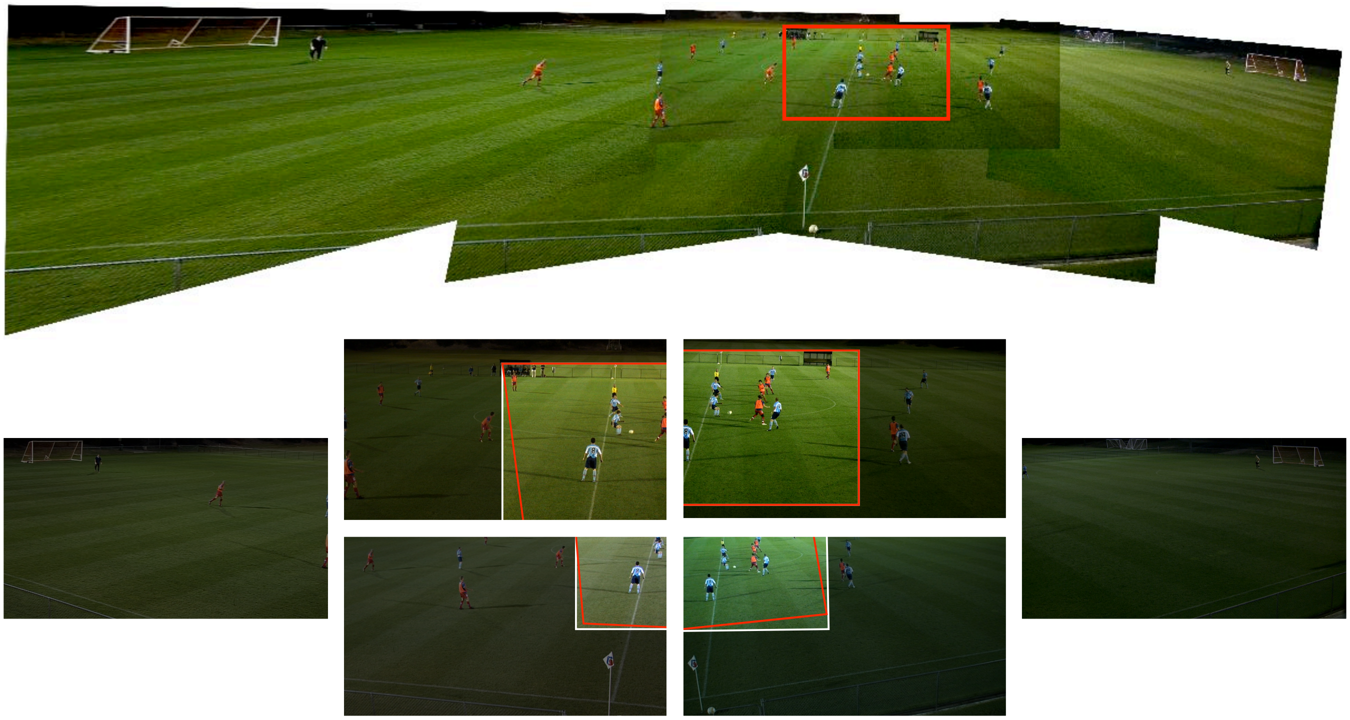

Fig. 6. *The region of interest within the canvas (top) is mapped into each camera's local co-ordinate system (bottom) using the inverse homography (mapping results shown in red). The bounding box of the resulting quadrilateral (white lines) defines the pixels that need to be transmitted. Dark regions of the images are not transmitted.*

The current prototype requires the user to specify correspondences between pairs of cameras, as well as the parameters for colour correction and temporal alignment. We plan to investigate methods for estimating these parameters automatically. Furthermore, a significant amount of bandwidth is presently devoted to transmitting background. With more powerful serving computers, one could extract the players and ball from the video and only transmit the foreground data.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] Hawk-Eye Innovations Ltd., Winchester, UK.

[2] A. Guéziec, "Tracking pitches for broadcast television," *Computer*, vol. 35, no. 3, pp. 38–43, March 2002.

[3] ProZone Sports Ltd., Leeds, UK.

[4] V. Lumikero, "Football tracking in wide-screen video sequences," Master's thesis, School of Electrical Engineering, Royal Institute of Technology, KTH, Stockholm, Sweden, 2004.

[5] LiberoVision Inc., Zurich, Switzerland.

[6] S. Gedikli, J. Bandouch, N. von Hoyningen-Huene, B. Kirchlechner, and M. Beetz, "An adaptive vision system for tracking soccer players from variable camera settings," in *ICCV*, 2007.

[7] J. B. Hayet, T. Mathes, J. Czyz, J. Piater, J. Verly, and B. Macq, "A modular multi-camera framework for team sports tracking," in *IEEE Conference on Advanced Video and Signal Based Surveillance*, 2005.

[8] D. Setterwall, "Computerised video analysis of football — technical and commercial possibilities for football coaching," Master's thesis, Department of Numerical Analysis and Computer Science, KTH, Stockholm, Sweden, 2003.

[9] J. Sullivan and S. Carlsson, "Tracking and labelling of ineracting multiple targets," in *ECCV*, vol. III, 2006, pp. 619–632.

[10] P. Nillius, J. Sullivan, and S. Carlsson, "Multi-target tracking — linking identities using bayesian network inference," in *CVPR*, 2006.

[11] T. Pintaric, U. Neumann, and A. Rizzo, "Immersive panoramic video," in *ACM Multimedia*, 2000.

[12] D. Tanguay, H. H. Baker, and D. Gelb, "Achieving high-resolution video using scalable capture, processing and display," in *International Conference on Computer Vision Theory and Applications*, 2006.

[13] J. Kopf, M. Uyttendaele, O. Deussen, and M. Cohen, "Capturing and viewing gigapixel images," in *ACM SIGGRAPH*, vol. 26, no. 3, 2007.

[14] Q. Ye, Q. Huang, S. Jiang, Y. Liu, and W. Gao, "Jersey number detection in sports video for athlete identification," in *SPIE Visual Communications and Image Processing*, 2005.

[15] "About HDV and the Apple Intermediate Codec," Apple Inc., Cupertino, CA, USA, 2005.

[16] "Core Image Programming Guide," Apple Inc., Cupertino, CA, USA, 2007.

[17] "OpenGL Performance Optimization: The Basics," Apple Inc., Cupertino, CA, USA, Technical Note TN2093, 2004.

[18] "vImage Programming Guide," Apple Inc., Cupertino, CA, USA, 2007.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Oxford University Press, 2003.

[20] G. Y. Tian, D. Gledhill, D. Taylor, and D. Clarke, "Colour correction for panoramic imaging," in *IEEE Conference on Information Visualisation*, 2002.