EUROGRAPHICS 2012 / P. Cignoni, T. Ertl
(Guest Editors)

Volume 31 (2012), Number 2

Novel-View Synthesis of Outdoor Sport Events Using an Adaptive View-Dependent Geometry

Marcel Germann[1], Tiberiu Popa[1], Richard Keiser[2], Remo Ziegler[2], Markus Gross[1]

[1]ETH Zurich, Switzerland. [2]LiberoVision AG, Switzerland

Figure 1: *Novel view rendering of a soccer scene using only the two input cameras shown to the left and the right.*

Abstract

We propose a novel fully automatic method for novel-viewpoint synthesis. Our method robustly handles multi-camera setups featuring wide-baselines in an uncontrolled environment. In a first step, robust and sparse point correspondences are found based on an extension of the Daisy features [TLF10]. These correspondences together with back-projection errors are used to drive a novel adaptive coarse to fine reconstruction method, allowing to approximate detailed geometry while avoiding an extreme triangle count.
To render the scene from arbitrary viewpoints we use a view-dependent blending of color information in combination with a view-dependent geometry morph. The view-dependent geometry compensates for misalignments caused by calibration errors. We demonstrate that our method works well under arbitrary lighting conditions with as little as two cameras featuring wide-baselines. The footage taken from real sports broadcast events contains fine geometric structures, which result in nice novel-viewpoint renderings despite of the low resolution in the images.

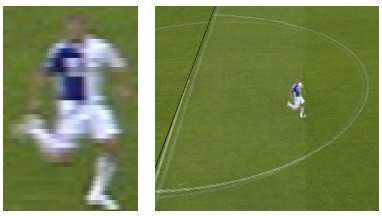
Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing Algorithms

1. Introduction

Novel-view video synthesis is an area of active research in computer graphics and vision to produce views of a scene from a virtual camera position. It has a large variety of applications including 2D to 3D conversion, telepresence, games, movie production, interactive TV sports broadcasts, etc. Although they all share the same fundamental problem, the nature of each individual application and setup constraints can yield to a very different set of challenges and requirements that may limit the range of suitable approaches. In this work we focus on conventional sports broadcasts footage [HGK*11, Lib]. The goal is to extend the creative freedom of an editor or director by providing the possibility to place a virtual camera in the stadium without having to change or add anything in the already existing sparse physical camera setup. This allows to have the perfect perspective and therefore perfect shot at any given time.

As noted in [GKH09, GPZ*11, HGK*11] this setup is very challenging due to several factors. There are typically only few moving cameras available that cover the interesting part of the scene and can be calibrated. In soccer they are positioned only on one side of the stadium. Although the cameras provide high resolution images, they are usually set to be wide-angle for editorial reasons. Therefore, an individual player covers only a height between 50 and 200 pixels [HGK*11] (Figure 2(a)). In sports broadcasts the player motion is usually fast and thus often results in motion blur. Methods for per frame automatic calibration in such setups [Tho06] suffer from errors and typically contain no radiometric calibration. All these factors have to be circumvented in order to create a convincing novel-view synthesis.

Due to these challenges the most popular approaches in this field still use simple planar billboards [HS06] which invariably produce unwanted visual artifacts like ghosting. A significant visual improvement can be achieved using articulated billboards [GHK*10]. Unfortunately, they rely on



(a) (b)

Figure 2: *(a) Resolution of input images (b) Example with calibration errors visible on the ground as ghosting of lines.*

fairly accurate pose estimation, which is cumbersome to do in these setups. Current methods [GPZ*11] require a large database of poses and high quality silhouettes while manual interaction is still required.

In contrast, we present a fully automatic method for novel-view synthesis using an adaptive reconstruction technique and view dependent rendering. Our method is robust to calibration errors and low resolution of the players. Central to our novel view synthesis method is a robust reconstruction step that adaptively recovers a geometry using as few as two wide-baseline cameras. The rendering method consists of a view-dependent geometry interpolation as well as a view-dependent texture blending, leading to convincing novel view synthesis results even in the challenging video setups of conventional TV broadcasts.

2. Related Work

Beginning with the pioneering work of Kanade et. al [KNR95], a lot of research on novel-view synthesis has been done in computer graphics for the last 15 years. Some methods were presented that work directly in 2D with morphs [YIS05] or layers [CR03]. They are only able to interpolate between images which is a camera flight along only a single line. Our method allows novel views in the entire area of the input cameras. Lipski et al. [LLB*10] extend the interpolations to a sphere. Their setup consists of cameras with less than 10° baseline and does not allow to fly into the scene because only interpolations on the sphere surface are possible. Other methods estimate depth from stereo obtained from a large set of video cameras [KRN97, MP04]. This approach works well for dense camera setups, which are not available for sports broadcast setups. If only few cameras are available, it is necessary to imply a geometric proxy with 3D information about the scene that improves the interpolation [BBM*01, SL04].

For rendering articulated characters such as players in a sport game, it is possible to use pose estimation to obtain a proxy geometry [CTMS03, VBMP08, dAST*08, GHK*10]. However, as noted in [GKH09, GPZ*11, HGK*11] pose estimation in such setups is difficult to do automatically and still requires too much user interaction to be practical for long sequences.

An alternative way to retrieve 3D geometric information is to exploit the 2D silhouettes in all the camera views and construct a visual hull of the scene, which can be used to render the scenes from an arbitrary viewpoint with the video camera images as textures [Lau94, MBR*00, LMS04, GTH*07, PLM*10]. Visual hull methods are suitable for novel view synthesis of single objects. In crowded scenes, unless a large number of cameras are employed, extraneous so-called phantom geometry is generated [FB09]. Additionally, when using distant cameras, small calibration errors can cause entire arms or legs to be cut off. Guillemaut et al. [GH11, GKH09, HGK*11] addresses many challenges for free-viewpoint video in sports broadcasting by jointly optimizing scene segmentation and player reconstruction. Their approach is leading to a more accurate geometry than the visual hull, but still requires a fairly large number of cameras (6-12). Inamoto et al. [IS02] also use silhouettes and match dense correspondences on epipolar lines. These correspondences, will suffer from the same drawbacks as the visual hull when working with weak calibration. Also their method does only interpolate on the path between cameras whereas we allow arbitrary viewpoints in the area of the two cameras.

Rather than relying on silhouettes alone to reconstruct a proxy geometry, an alternative way is to use dense geometry reconstruction techniques [MP04, BPS*08, BBPP10]. The results are very convincing, but dense reconstruction is difficult to do in wide-baseline camera setups. An alternative approach suitable for setups where dense feature matching is difficult to accomplish, is patch based reconstruction [FSB06, SLAM08, SSS09, GFP10]. Patch based reconstruction methods only require a sparse set of correspondences, but are limited to objects with a strong planarity assumption, and thus not suitable for reconstructing players.

Since our wide-base camera setup does not allow dense reconstruction, but it is possible to get reliable sparse matches [TLF10], we propose a technique in the spirit of the patch-based reconstruction methods. However, instead of a bottom-up approach that tries to cluster pixels in planar patches, we propose a top-down approach that starts by reconstructing the scene with a few large planar pieces and recursively subdivides the geometry and gradually adds geometric detail up to resolution limitations.

Reconstructing the geometry is only half of the problem. To synthesize novel views, usually the geometry is rendered using projective texturing from the original video footage. The pixel colors from the video cameras are blended using view-dependent weights [BBM*01]. However, inherent calibration errors will yield rendering artifacts. A popular method to address this problem is floating textures [EDDM*08]. The geometry is rendered and shaded independently from the first camera and from the second camera. Then, in image space, using optical flow, the two images are locally aligned to eliminate the ghosting and blurring artifacts. Unfortunately, optical flow will not provide accurate results in our setup. The exact view-dependent visual hull [MH06] renders only a view-dependent subsample of the visual hull similar to primary rays in raytracing. How-

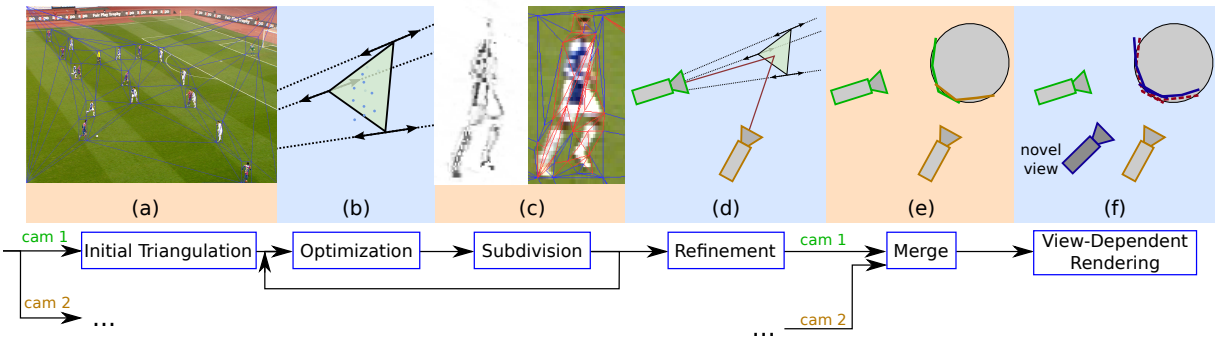


Figure 3: *Overview of the algorithm: (a)-(e) The adaptive reconstruction. (f) The view-dependent rendering.*

ever, the geometry of the scene is not view-dependent. It is a static visual hull that will still cut off entire arms or legs in our setup due to calibration errors. We propose a view-dependent rendering and texturing method that corrects the local misalignments by morphing the 3D geometry using sparse feature correspondences in the camera images.

3. Overview

The backbone of our method is a top-down adaptive reconstruction technique that is able to retrieve a 2.5D triangulation of the players in each camera even in challenging setups consisting of only two wide-baseline cameras. The reconstruction starts with a simple triangulation (figure 3(a)). The triangles are placed according to a sparse 3D point cloud (figure 3(b)), which is generated using an extended version of the Daisy features [TLF10]. If the triangles are too large and do not represent the shape of the object accurately (red triangles in figure 3(c)), they are subdivided. This process is repeated until the triangles are just small enough to approximate the shape of the object. This way the reconstruction method inherently adapts to the geometric complexity as well as to the resolution of the represented object. In an additional refinement step the vertex depths of those triangles that do not contain any features are set to optimal depth values of neighboring triangles including random perturbation tests. The adaptive level of detail and the reconstruction lead to a robust geometry. To cover also parts of the scene that are occluded in one camera, we repeat the reconstruction for each camera as a *base camera* and merge these 2.5D reconstructions into a final 3D geometry (figure 3(e)).

We render this geometry from an arbitrary viewpoint and blend the color information from the available cameras. However, inherent calibration errors will yield rendering artifacts such as ghosting. Therefore, we propose a method that morphs the geometry based on the viewing positions to compensate for calibration errors (figure 3(f)).

In section 4 the adaptive reconstruction is elaborated in detail. The view-dependent geometry morph and rendering is described in section 5. Our proposed method is evaluated and discussed in section 6 and section 7.

4. Adaptive Reconstruction

We chose to represent the geometry as a per camera triangle soup. The advantage of a triangle soup as opposed to a connected mesh is that it allows us to place these triangles independently solving implicitly for depth discontinuities. However, to avoid discontinuities on connected surface parts, we connect the triangles that are close together before the rendering stage. Our reconstruction algorithm proceeds as follows (Figure 3):

1. *Initial Triangulation:* an initial set of triangles is created from the image of one of the cameras. The triangulation of this base camera is illustrated in figure 3(a). The triangles are aligned with the view of the base camera such that the only degree of freedom is the depth of its vertices in camera coordinates. This allows for a low-degree of freedom optimization that facilitates the process of positioning them in 3D, without sacrificing precision.
2. *Triangle Optimization:* in this step each triangle of the current set is positioned independently in 3D by optimizing the depth of its vertices only. As a result the projection of the triangle onto the base camera never changes. The optimization process uses robust sparse feature correspondences from pixels inside the projection of the triangle to determine the depth of each vertex.
3. *Subdivision:* the 3D triangles may not approximate well the local geometry. For instance, in figure 3(a) the players are initially approximated only by two triangles. In this case we subdivide the triangle if the error evaluated by a robust error metric of texture re-projection is too big. Figure 3(c) shows triangles to be subdivided in red. We repeat the optimization and subdivision step until the re-projection error is small for all triangles.
4. *Refinement:* due to occlusions and the low resolution of the image, it is not always possible to find image features in every triangle. If a triangle has no features it inherits its vertex depths from the previous subdivision. However, these depths could be wrong and as a result we might have rendering artifacts such as missing parts of the players. To further refine the position of such triangles, we employ a heuristic to determine their 3D location based on depth guesses from the neighboring triangles combined with random perturbations.

5. *Merge:* We reconstruct a 2.5D geometry the same way for every input camera as base camera and build the union of these resulting in a final 3D reconstruction (figure 3(e)).

4.1. Inital Triangulation

The initial triangulation is done in 2D using a delaunay triangulation of the four corners of the image and the corners of the bounding boxes of each player (Figure 3(a)). The bounding boxes of each player are computed automatically [FBLF07]. Note that we do not rely on an accurate bounding box detection. As one can see in figure 3(a), players who are occluding each other are generally classified into one box. Also the goal keeper's box is shifted. This does not create any problems for the algorithm since it is only used as an initial triangulation. This procedure gives us a set of 2D triangles in a base camera, which we project to the ground plane to obtain an initial 3D triangle soup. The ground plane can be found automatically by the calibration method according to pitch lines such as described in [Tho06].

4.2. Triangle Optimization

The main goal of this step is to optimize the 3D position of the triangles in the current set. The triangles are optimized independently while the only degree of freedom is the depth along the camera ray of their respective vertices as illustrated in Figure 3(b). Our optimization technique combines two criteria: one is based on a background color model and one based on robust feature matches.

Background Color Model. Triangles are part of the background if more than 95% of the pixels are classified as such by the background color model. An alternative way to accomplish this test is using a background subtraction technique as described in [ZPB07]. If the triangle is classified as background, its vertices are pushed to the ground.

The depth of non-background triangles relies on feature matching of pixels contained in the corresponding 2D triangles in the camera image. We first compute a robust set of matching pairs according to section 4.2.1. Each 2D pair of matches represents a point in 3D that can be computed as the intersection of two rays. Due to calibration errors these rays usually do not intersect, and we use the point on the base camera's ray that is closest to the other ray, as explained in section 5. The reconstructed set of 3D points belongs to the scene geometry of the base camera. By applying a RANSAC technique [FB81] we can accurately and robustly fit a plane to the set of 3D points. Once the best fitting plane is determined, we can solve for the depths values of every vertex, such that the optimized triangle lies in the plane.

4.2.1. Feature matching

In order to position a triangle in 3D we rely on image feature matching between the views. We selected Daisy features [TLF10], which are designed for wide-baselines. However, due to the low resolution of each player and the lack of features on the pitch, the feature detection contains a lot of wrongly matched pairs. Therefore, we added more constraints to the matching operator to get more reliable, albeit fewer, matches. We find robust matches in three steps:

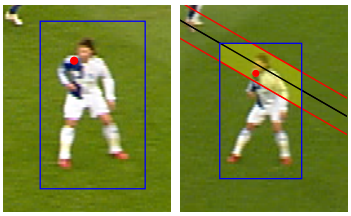


Figure 4: *Feature correspondence search for a pixel in the left image within a cropped epipolar stripe in the image of the other camera.*

1. For every pixel in the base camera we restrict the search space of possible matches in the second camera. This search space is defined by a band of $d = 20$ pixels around the epipolar line, whereof only pixels lying inside the bounding box are considered. This is illustrated as the yellow area in figure 4.
2. Only matches with a Daisy error value below 1.4 and below 0.8 times the average of the matches within this stripe are considered.
3. We verify that the match is symmetric. That is, if p_{c_0} is a pixel in the base camera and p_{c_1} is its corresponding Daisy match in the second camera, the Daisy match of pixel p_{c_1} has to lie within five pixels of p_{c_0}.

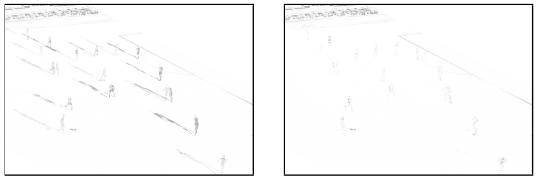
The resulting matches are a relatively sparse set of reliable matches that we use in the triangle optimization process (depending on the scene, it varies from 20 to 300 per player).

4.3. Adaptive Subdivision

If the scene geometry represented by a triangle significantly differs from a planar surface, a further subdivision of the triangle is required. The triangles are only refined if the projective textures from the respective source cameras do not match. The resulting triangle soup only features small triangles where the underlying 3D geometry requires a refinement.

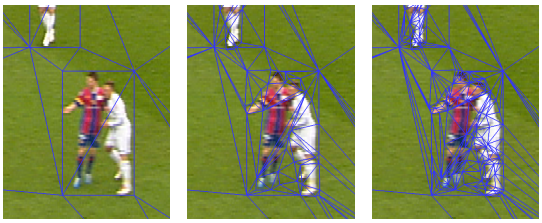
Error metric. For every triangle a color error value is computed to decide if the triangle has to be subdivided or not. To do so, the current geometry is used to reproject the textures from all cameras into the base camera. The error of a single triangle is then computed as the average of all pixel differences, which are not occluded in any of the cameras. To have comparable image sources, the appearances of the images are roughly adjusted by adding a compensating color shift as described in section 5. An example of the initial per pixel error viewed from the base camera is shown in Figure 5(a) where the entire geometry is approximated as the ground plane.

Inaccurate calibration will inherently introduce a bias in



(a) (b)

Figure 5: *Example for an initial color error when using the ground plane as the geometric proxy. The errors appear at pixels with wrong depth values.*



(a) (b) (c)

Figure 6: *(a) Initial triangulation. (b) First subdivision based on player's silhouettes. (c) Final subdivided geometry.*

this error metric as the projection of a 3D point into the cameras suffers from calibration errors. We address this problem by adding a geometric shift to each triangle vertex. This view-dependent geometry is described in detail in section 5.

Subdivisions The subdivision is a simple splitting of the triangle into two halves at the longest edge, including splitting of neighbors sharing the same edge. This directly inherits the position in 3D to the children. Therefore, even if no feature point is available for a child (e.g. at occlusions), it still inherits a most plausible 3D position and orientation. In the first iteration step, we use the background color model and a blob detection to get contour lines. These contour lines guide the first subdivision of the initial triangulation as shown in figure 6(b). This speeds up the reconstruction and avoids high triangle counts, since it adds edges at potential depth discontinuities.

Figure 6(c) shows a final subdivision after 3 iteration steps of optimization and subdivision.

4.4. Refinement

Due to a lack of Daisy features, a small subset might not be positioned correctly in 3D and could also not be recovered

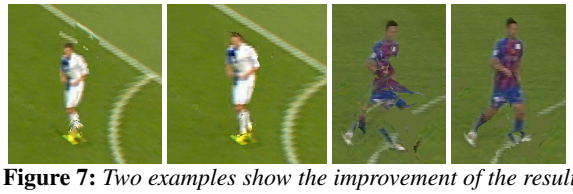


Figure 7: *Two examples show the improvement of the result before and after applying the refinement step.*

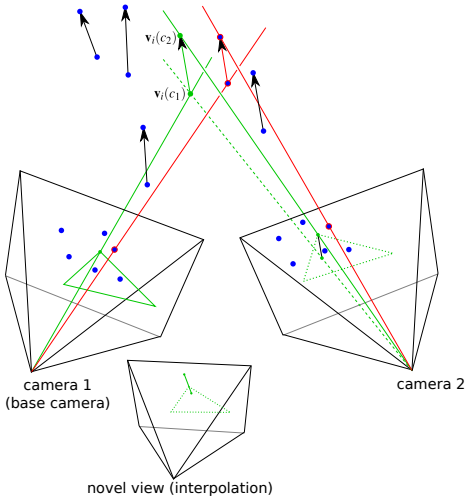


Figure 8: *Sketch of the view dependent geometry. The arrows indicate the view dependent geometric shift.*

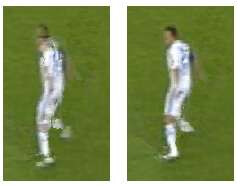
by the inheritance of the subdivision. This may lead to rendering artifacts as shown in figure 7. To resolve these, we assume that neighboring triangles tend to have similar 3D depths values.

We try several positions using depth values from the neighboring triangles adding random perturbations. Finally, we select the one minimizing the color error.

Similar to the generalized PatchMatch method [BSGF10] this is done iteratively over all triangles 10 times with 10 random perturbations per vertex and iteration. Figure 7 shows the improvement of this step.

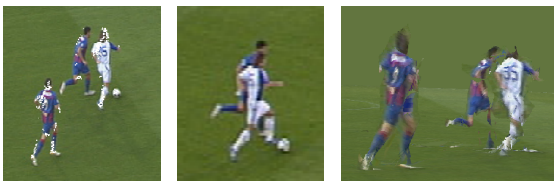
5. View-dependent Geometry and Rendering

Our representation could be rendered as simple triangles in 3D with the color values from projective texturing of the source camera images, using some smart blending function [BBM*01]. Due to the inherent errors in the calibration process, the projection errors - even on a perfectly reconstructed geometry - result in rendering artifacts such as blurriness and ghosting. This problem is illustrated in figure 8. The blue dots are reliable feature matches from one camera to the other. If the calibration is correct then the corresponding rays (the red lines) would intersect. This is generally not the case. Wherever we position the 3D point, in at least one camera it will not project back to the 2D position where the feature point was detected. To solve this, we shift (morph) this point in 3D along the line of the shortest path between the two rays (the red arrow) when changing the viewing position. We call this shortest path a displacement. These displacements describe the geometric 3D morph function from every feature point in the base camera to the corresponding feature point in the other camera. To calculate the morph function of any view-dependent vertex (green dot), we interpolate all the displacements of neighboring features using



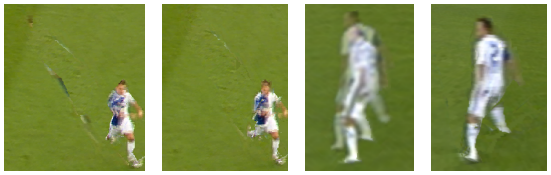
(a) (b)

Figure 9: *(a) Rendering with one geometry. (b) Rendering with a view-dependent geometry.*



(a) (b) (c)

Figure 10: *Difficult occlusion. (a) View in camera 1 with feature points shown as white dots. (b) View in camera 2 where one player is almost entirely occluded. (c) Novel view.*



(a) (b) (c) (d)

Figure 11: *(a) Reconstruction with triangulation in camera 1. The ghostings on the ground are parts not seen and thus not reconstructed in camera 1. (b) Merge of the reconstructions of camera 1 and camera 2. (c) A rendering using billboarding. (d) The same view using our method.*

a weighted average (the green arrow). The weighting function is a gaussian, while only features lying in a radius of 1m of the vertex are considered being neighbors. This view-dependent rendering (figure 3(f)) is not a 2D morph but a morph of the 3D geometry and should not be confused with the merge of the 2.5D reconstructions (figure 3(e)). It resolves most of the rendering artifacts as demonstrated in figure 9.

More formally, let $\mathbf{v}_i(c)$ be the view-dependent position of vertex $\mathbf{v}_i$ in camera $c \in C$ as illustrated in figure 8. Let $\mathbf{v}_i(\hat{c})$ be the 3d position that we need to compute to render the vertex $\mathbf{v}_i$ from a novel viewpoint $\hat{c}$. This view-dependent position can be interpolated between its corresponding positions of all cameras C.

$$\mathbf{v}_i(\hat{c}) = \sum_{c \in C} \lambda_c(\hat{c}) \mathbf{v}_i(c) \tag{1}$$

where the weights λ_c correspond to the blending weights as described in [GHK*10]. The angles α_c used for the blending are defined as the angle between the vector from $\mathbf{v}$ to the viewers position and the vector from $\mathbf{v}$ to the position of camera c, with $\mathbf{v} = \frac{1}{|C|} \sum_{c \in C} \mathbf{v}_i(c)$.

For the texture blending, we use projective textures. However, it is important that for the projection from a source camera c we do not use the interpolated geometry (vertex $\mathbf{v}_i(\hat{c})$) but the geometry relating to camera c (vertex $\mathbf{v}_i(c)$). The color values are blended with the same weights λ_c used above. However, at occlusions, the λ_c of any camera c not covering this pixel is set to 0 and the weights are renormalized.

The cameras are typically not radiometrically calibrated. Therefore, we also compute the color shifts between cameras: an average color R_c, G_c, B_c is computed per camera c. It is the average of the color values of those pixels of the image c that project onto 3D positions covered by more than one camera. The per-pixel average color for the interpolated view rendering is then given by

$$R_{\hat{c}} = \sum_{c \in C} \lambda_c(\hat{c}) R_c, \quad G_{\hat{c}} = \sum_{c \in C} \lambda_c(\hat{c}) G_c, \quad B_{\hat{c}} = \sum_{c \in C} \lambda_c(\hat{c}) B_c.$$

For every source camera, the RGB values $R_{\hat{c}} - R_c$, $G_{\hat{c}} - G_c$, $B_{\hat{c}} - B_c$ are added to the texture values of camera c before interpolating the color of a pixel.

This method for geometry interpolation and texturing is also used for the computation of the color error in section 4.3.

6. Results

We demonstrate our method on footage of original TV broadcasts of several soccer scenes. All computations were done on a standard desktop computer (Core i7 CPU, 3.20GHz). All results shown here and in the video are reconstructed using only two source cameras.

Despite low resolutions (figure 2(a)) and calibration errors (figure 2(b)), our algorithm fully automatically reconstructs plausible novel views as shown in figure 11(d). Figure 9 illustrates the effect of the view dependent geometry that is able to reduce calibration error artifacts.

With our improved Daisy feature matching, the feature matches are reduced to a set of reliable correspondences. Sometimes this set contains only a few matches per player, but nevertheless our adaptive reconstruction method recovers a good approximation of the players geometry as shown in figure 10.

Our method performs a reconstruction for each camera. This can be used to recover parts occluded in one camera but visible in another. The result of merging two reconstructions is shown in figure 11, where figure 11(a) shows a novel view using the reconstruction of camera 1 only and figure 11(b) shows the same view with the unified reconstruction of the two cameras. With the reconstruction of only one camera, parts of players visible only in the other camera are not reconstructed and are thus projected onto the ground. With the

merged reconstruction we also get a valid depth for these parts allowing to reconstruct occlusions (figure 3(e)). Figure 10 demonstrates this with another example.

Figures 11(c) and 11(d) show a comparison to billboarding in a view half-way between the two source cameras. While billboarding shows ghosting (more than two legs) due to the planar representation, our method preserves the pose of the player in the novel view.

Figure 12 depicts a ground truth comparison to a third camera which was not used in the reconstruction. It demonstrates that the result of our reconstruction is correct and with similar quality as the input images. More results are shown in figure 13. The first two images are always the input images followed by the novel views. They show views not lying directly in between the cameras, e.g. top views or views from the height of the players heads.

The accompanying video shows several flights in different scenes. To demonstrate a possible application, it contains also examples of flights over time. Despite not using any temporal information, coherence or smoothing, these dynamic scenes still result in plausible renderings. Artifacts are visible mostly when players come into or leave a cameras view range. Since we reduce the search for Daisy feature matches to the bounding box found by the player detection, a player that is not detected will not be reconstructed. An example for this can be seen in the video in the first dynamic scene where the goal keeper in the green uniform as well as the ball are not reconstructed.

The rendering of novel views is done in real-time, i.e., more than 60 frames per second for HD resolution. Our fully automatic algorithm for reconstruction takes on average 1 minute per frame. The exact timing depends on the scene complexity, but none of our examples required more than 2 minutes to reconstruct. About 19 seconds of this time are Daisy feature vector computation and our feature matching. The rest of the time is spent about equally for positioning, and for the refinement. The time required per frame could be reduced by parallelization in several ways. First, the reconstruction for each camera is done completely independently (figure 3). Second, the refinement is done independently per triangle. Third, the feature matching could be parallelized.

7. Limitations and Future Work

Although our approach produces convincing results for a two wide-baseline camera setup, one can still spot some minor visual artifacts. Many of these are caused by inherent limitations by the fact that we only use two cameras. For instance, subjects too close to the front will have nearly no overlapping parts and, therefore, cannot be reconstructed. Also, some subjects are visible only in one camera and cannot be reconstructed. Due to lack of good robust matches occasionally triangles are not positioned optimally causing cracking artifacts. Although our method can recover a plausible depth in most occlusion cases where parts are only visible in one camera, there are situations where too few or no features were found in the neighborhood and thus leading to visual artifacts. Simply adding more cameras to the setup will automatically improve these issues. The camera weights λ, the view dependent morph (equation (1)) and the color shifts can be computed for any arbitrary number of cameras.

Our method processes every frame independently, while ignoring temporal coherence. The quality of this straight forward application resulting in only occasional flickering shows the big potential our fully automatic method has. In the future, we would like to add temporal coherence to our system. For instance, optical flow could be used to initialize the geometry in the next mesh or to smooth the geometry in the temporal domain. This will not only yield better results, but it will also increase the efficiency of our method.

8. Conclusions

In this paper we presented a fully automatic novel-view synthesis method suitable for conventional TV sport broadcasts. The setup consists of only two wide-baseline video cameras. The main ingredient of our method is an adaptive and view-dependent reconstruction technique that can reconstruct players even at very low resolutions. The geometry is reconstructed in a top-down fashion. Geometric detail is added gradually in the areas where it is needed based on a reliable sparse feature matching technique. It also robustly handles areas with no image features by inferring the position based on the neighboring triangles and back-projection error. The geometry is then rendered from a novel viewpoint using a view-dependent geometry morphing and texture interpolation technique that alleviates rendering artifacts stemmed from calibration errors. We proved the visual quality and reliability of our technique by applying it to footage of soccer broadcasts.

9. Acknowledgments

The data is courtesy of Teleclub and LiberoVision. This project is supported by CTI Switzerland and by NSERC.

References

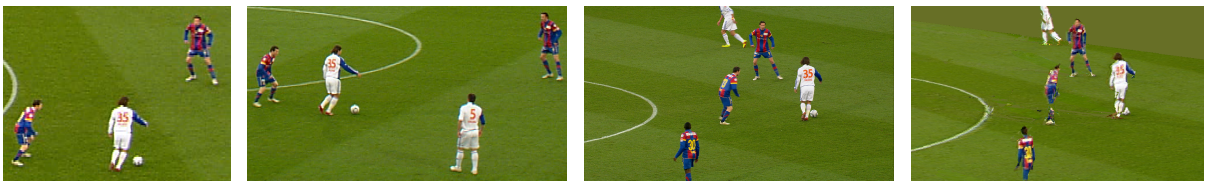
[BBM*01] BUEHLER C., BOSSE M., MCMILLAN L., GORTLER S., COHEN M.: Unstructured lumigraph rendering. In *SIGGRAPH* (2001). 2, 5

[BBPP10] BALLAN L., BROSTOW G. J., PUWEIN J., POLLEFEYS M.: Unstructured video-based rendering: interactive exploration of casually captured videos. In *SIGGRAPH* (2010). 2

[BPS*08] BRADLEY D., POPA T., SHEFFER A., HEIDRICH W., BOUBEKEUR T.: Markerless garment capture. In *SIGGRAPH* (2008). 2

[BSGF10] BARNES C., SHECHTMAN E., GOLDMAN D. B., FINKELSTEIN A.: The generalized patchmatch correspondence algorithm. In *ECCV* (2010). 5

[CR03] CONNOR K., REID I.: A multiple view layered representation for dynamic novel view synthesis. In *BMVC* (2003). 2



(a) (b) (c) (d)

Figure 12: *Leave-one-out example: (a) and (b) Closeups of the two source camera images used. (c) High resolution ground truth from a third (not used) camera that zoomed in. (d) Our reconstruction of the same view*

[CTMS03] CARRANZA J., THEOBALT C., MAGNOR M. A., SEIDEL H.-P.: Free-viewpoint video of human actors. In *SIGGRAPH* (2003). 2

[dAST*08] DE AGUIAR E., STOLL C., THEOBALT C., AHMED N., SEIDEL H.-P., THRUN S.: Performance capture from sparse multi-view video. In *SIGGRAPH* (2008). 2

[EDDM*08] EISEMANN M., DE DECKER B., MAGNOR M., BEKAERT P., DE AGUIAR E., AHMED N., THEOBALT C., SELLENT A.: Floating textures. In *Eurographics* (2008). 2

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24* (June 1981), 381–395. 4

[FB09] FRANCO J.-S., BOYER E.: Efficient polyhedral modelling from silhouettes. In *PAMI* (2009). 2

[FBLF07] FLEURET F., BERCLAZ J., LENGAGNE R., FUA P.: Multi-camera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007). 4

[FSB06] FRAUNDORFER F., SCHINDLER K., BISCHOF H.: Piecewise planar scene reconstruction from sparse correspondences. *Image and Vision Computing 24*, 4 (2006), 395 – 406. 2

[GFP10] GALLUP D., FRAHM J.-M., POLLEFEYS M.: Piecewise planar and non-planar stereo for urban scene reconstruction. In *CVPR* (2010). 2

[GH11] GUILLEMAUT J.-Y., HILTON A.: Joint multi-layer segmentation and reconstruction for free-viewpoint video applications. *IJCV 93* (2011), 73–100. 2

[GHK*10] GERMANN M., HORNUNG A., KEISER R., ZIEGLER R., WÜRMLIN S., GROSS M.: Articulated billboards for video-based rendering. In *Eurographics* (2010). 1, 2, 6

[GKH09] GUILLEMAUT J.-Y., KILNER J., HILTON A.: Robust graph-cut scene segmentation and reconstruction for free-viewpoint video of complex dynamic scenes. In *ICCV* (2009). 1, 2

[GPZ*11] GERMANN M., POPA T., ZIEGLER R., KEISER R., GROSS M.: Space-time body pose estimation in uncontrolled environments. In *3DIMPVT* (2011). 1, 2

[GTH*07] GRAU O., THOMAS G. A., HILTON A., KILNER J., STARCK J.: A robust free-viewpoint video system for sport scenes. In *3DTV* (2007). 2

[HGK*11] HILTON A., GUILLEMAUT J., KILNER J., GRAU O., THOMAS G.: 3d-tv production from conventional cameras for sports broadcast. *Broadcasting, IEEE Transactions on 57*, 2 (june 2011), 462 –476. 1, 2

[HS06] HAYASHI K., SAITO H.: Synthesizing free-viewpoint images from multiple view videos in soccer stadium. In *CGIV* (2006). 1

[IS02] INAMOTO N., SAITO H.: Intermediate view generation of soccer scene from multiple videos. In *ICPR* (2002). 2

[KNR95] KANADE T., NARAYANAN P., RANDER P.: Virtualized reality: concepts and early results. In *IEEE Workshop on Representation of Visual Scenes* (1995). 2

[KRN97] KANADE T., RANDER P., NARAYANAN P.: Virtualized reality: constructing virtual worlds from real scenes. *Multimedia, IEEE 4*, 1 (jan-mar 1997), 34 –47. 2

[Lau94] LAURENTINI A.: The visual hull concept for silhouette-based image understanding. In *PAMI* (1994). 2

[Lib] LIBEROVISION:. www.liberovision.com. 1

[LLB*10] LIPSKI C., LINZ C., BERGER K., SELLENT A., MAGNOR M.: Virtual video camera: Image-based viewpoint navigation through space and time. 2

[LMS04] LI M., MAGNOR M., SEIDEL H.-P.: A hybrid hardware-accelerated algorithm for high quality rendering of visual hulls. In *Graphics Interface* (2004), pp. 41–48. 2

[MBR*00] MATUSIK W., BUEHLER C., RASKAR R., GORTLER S. J., MCMILLAN L.: Image-based visual hulls. In *SIGGRAPH* (2000). 2

[MH06] MILLER G., HILTON A.: Exact view-dependent visual hulls. In *ICPR* (2006). 2

[MP04] MATUSIK W., PFISTER H.: 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *SIGGRAPH* (2004). 2

[PLM*10] PETIT B., LESAGE J.-D., MENIER C., ALLARD J., FRANCO J.-S., RAFFIN B., BOYER E., FAURE F.: Multicamera real-time 3D modeling for telepresence and remote collaboration. *Intern. Journ. of Digital Multi. Broadcasting* (2010). 2

[SL04] SIU A. M. K., LAU R. W. H.: Image-based modeling and rendering with geometric proxy. In *ACM Multimedia* (2004). 2

[SLAM08] STICH T., LINZ C., ALBUQUERQUE G., MAGNOR M.: View and time interpolation in image space. 2

[SSS09] SINHA S. N., STEEDLY D., SZELISKI R.: Piecewise planar stereo for image-based rendering. In *ICCV* (2009). 2

[Tho06] THOMAS G.: Real-time camera pose estimation for augmenting sports scenes. In *CVMP* (2006). 1, 4

[TLF10] TOLA E., LEPETIT V., FUA P.: Daisy: an efficient dense descriptor applied to wide baseline stereo. In *PAMI* (2010). 1, 2, 3, 4

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIĆ J.: Articulated mesh animation from multi-view silhouettes. In *SIGGRAPH* (2008). 2

[YIS05] YAMAZAKI S., IKEUCHI K., SHINAGAWA Y.: Plausible image matching: determining dense and smooth mapping between images without a priori knowledge. 2

[ZPB07] ZACH C., POCK T., BISCHOF H.: A globally optimal algorithm for robust tv-l1 range image integration. In *ICCV* (2007). 4

Figure 13: *Results: the first row (little images) always shows the input camera views.*