Virtual camera synthesis for soccer game replays

N. Papadakis[1], A. Baeza[1], I. Rius[2], X. Armangué[2], A. Bugeau[1], O. D'Hondt[1], P. Gargallo[1], V. Caselles[3] and S. Sagàs[2]

[1] Barcelona Media, Avenida Diagonal 177, 08017 Barcelona, Spain
[2] Mediapro, Avenida Diagonal 177, 08017 Barcelona, Spain
[3] Universitat Pompeu Fabra, Carrer de Roc Boronat 138, 08018 Barcelona, Spain

Abstract

In this paper, we present a set of tools developed during the creation of a platform that allows the automatic generation of virtual views in a live soccer game production. Observing the scene through a multi-camera system, a 3D approximation of the players is computed and used for the synthesis of virtual views. The system is suitable both for static scenes, to create bullet time effects, and for video applications, where the virtual camera moves as the game plays.

Keywords: Novel view synthesis, soccer game, depth estimation, image inpainting.

1 Introduction

The creation of virtual viewpoint in soccer game scenario is an interesting problem for two main reasons: first, for its large viewer base and potential industrial impact and second, for the many scientific challenges it presents. Hence, a collaboration between industries and universities, through the Spanish project CENIT-2007-1012 i3media, has lead us to investigate such an application. This paper presents an overview of the system that has been developed during this collaboration. The system is able to automatically produce videos taken from a virtual, moving camera given the video streams of multiple real cameras.

Before going into the details of the system, let us first mention the motivation for building such a system from an industrial and from a technical point of view.

Industrial applications

Recent technological advances on multimedia content coding, image processing and computer vision algorithms along with the higher quality and lower cost of imaging hardware are having a large impact on the multimedia and broadcast industry. Nowadays, this environment is rapidly changing by the introduction of sophisticated tools for media creation, and the proliferation of new and more compelling forms of media contents and platforms: high definition television (HDTV), fully digital cinema pipeline, rich multimedia internet sites, immersive online video games, spectacular visual effects and more recently the explosion of 3D stereo content and exhibition systems.

Nevertheless, even though the 3D stereo content brings more realism to the spectator, there is still a large gap with respect to the immersivity of the spectator within the scene. For the future, industry aims at significantly reducing this gap by introducing the concept of live free-viewpoint media content. It relies on the development of the necessary technologies for capturing, processing and delivering truly interactive photorealistic content to a variety of professional and home remote users.

Towards this end, virtual camera synthesis technologies would allow to interact and freely explore a live-action 3D scene, thus allowing the viewers to control the focus of attention rather than being restricted to the views offered by a director, or bringing the chance to directors to offer the action from novel and impossible points of view where a physical camera can not be placed. For instance, one could watch a soccer match from the point of view of a particular player or even following the ball. In addition, replays could benefit from these technologies to highlight a particular aspect of the game by choosing the best viewpoint for it. Summarizing, the scene could be observed from novel, unique and compelling viewpoints, thus engaging the spectator in a much richer and immersive experience.

The technical challenges

The creation of virtual view for soccer games relies on numerous problems. Cameras are located in outdoor environments, often in uncontrolled and changing lighting conditions, they are located in a wide-baseline disposition, and may be moving. Besides the calibration problems in such environments, the relatively low resolution and the complexity of the scene are considerable. Indeed, two teams of 11 players dressed alike play in a very large arena outdoors plus several referees running freely in the field, bumping into each other and creating all kinds of occlusions are not a simple scenario.

The problem has been addressed using a variety of tools of stereo reconstruction techniques like using billboards that produce a flat 3D reconstruction of a scene [10, 11, 12, 21, 24], using visual hulls or photohulls [10, 11, 12], depth computations using graph cuts [12] or joint segmentation and depth estimation using graph cuts to optimize an energy function combining multiple image cues with strong priors [13]. The separation of the soccer field and the players has been used as a fundamental tool in order to speed up or to facilitate the computation of correspondences [10, 11, 12, 13, 17, 18].

978-0-7695-4268-3/10 $26.00 © 2010 IEEE
DOI 10.1109/CVMP.2010.20

Still, there are several aspects of the problem that need to be improved to generate quality videos for retransmission. To get a correct parallax effect we need to compute the depth of the players in the scene. The complexity of the scenes needs to use depth computation or stereo reconstruction methods that take into account the visibility constraints and the handling of occlusions. On the other hand, to synthesize video sequences one needs to compute time coherent depths or at least to guarantee a time coherent synthesis. Finally, one needs to fill-in the holes during the synthesis with inpainting methods and to correct the eventual artifacts with a suitable video filtering.

Overview of the system

The system presented in this paper is composed of different steps from image capture to post-processing. Figure 1 summarizes the complete pipeline.

- The first step is to capture the images. We present the multicamera acquisition system used for the application in section 2. It currently uses four static, synchronized cameras.
- Next, as a pre-process, the color of the images between the different cameras is equalized, and the cameras geometrically calibrated.
- The core of the process, presented in section 3, consists in computing the geometry of the players. The players are first segmented via background substraction. Then, their 3D geometry is estimated by computing the depth of each pixel in the images via a graph-cut technique.
- With the computed depth maps, a novel view synthesis algorithm, presented in section 4, creates the image that would have been seen from a virtual camera viewpoint.
- Finally, two post-processing algorithms are used to improve the quality of the synthesis (section 5). First, artifacts around depth discontinuities are removed via an inpainting algorithm. Then, temporal coherence is enforced by a temporal filter.

Details on the experimentations are given in section 6.

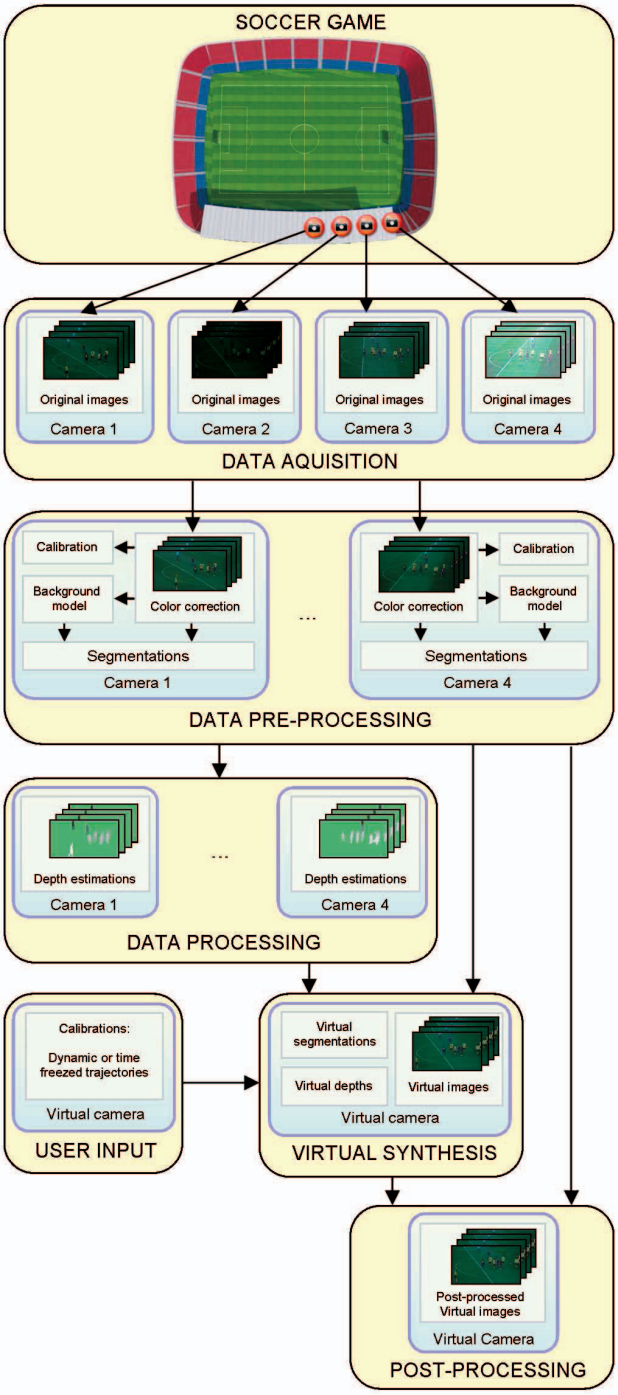


***Figure 1:** Description of the overall process.*

2 Data acquisition

A specific acquisition system has been built for this application. The system is composed of 4 cameras and is capable of recording synchronized multiple-views from the same scene in FullHD resolution. In the following we will detail the hardware and software composing the system, its features and give an overview about its operation and output.

The multiview capture system is composed of 4 JAI/PULNIX TMC 2030GE color cameras. These cameras have a 1" CCD sensor which can produce 1920x1080 images at 32 fps. In addition, each pixel can be 8,10 or 12-bits valued. They can be controlled via the GigE Vision Ethernet interface which enables a cable length of 100m per camera. Finally, all cameras can be synchronized at a shutter level via an external sync cable.

Each pair of cameras is connected to a PC with the following specifications:

- Intel Core 2 Quad CPU Q6600@2.40 Ghz with 3.6 Gbytes of DDR2 RAM memory.
- 2 x Giga Ethernet Intel Pro/1000 PT Dual card.
- 2 x SATA hard drive of 10000 rpm (WD1500ADFD).
- Windows XP Professional Service Pack 3 OS.

Regarding the acquisition procedure, first we aim at setting the zoom level for each camera. Subsequently, camera's gain, shutter speed and aperture is configured on each camera depending on the lighting conditions. The goal is to obtain uniformly contrasted images, without motion blur artifacts at the possible lowest noise level. Finally, the focus on each camera is adjusted to obtain sharp images.

Each camera is produces a raw 8-bit 1920x1080 video stream at 25 fps, which is stored in a fast hard drive unprocessed. This results in a data bandwidth of 50 Mbytes/sec. Then, the Adaptive homogeneity-directed (AHD) Bayer demosaicing algorithm [16] is used to compute high quality color images from the raw data. Then, output images are color corrected so color constancy is kept among different views of the same object. Finally, each camera is calibrated optically by detecting the soccer field lines and matching them against a known 3D model. In the following, we describe further the color correction and calibration methods.

2.1 Color correction

A color correction step is performed over the final acquired images in order to make uniform the color between views. Its ultimate goal is that an object which is seen from several views has similar color values among all its images. Hence, color coherence constraints can be exploited by the subsequent computing stages, and the final synthesized images look realistic and free of color artifacts when combining multiple image sources.

Towards this end, 2 complementary approaches are considered. The first approach $(A1)$ is preferred if we have physical access to the field before a match. It consists of inserting a pattern of known color into the scene and recording a color calibration sequence. Then, we use correspondences between images to estimate the parameters of the color correction algorithm. Alternatively, the second approach $(A2)$ extracts SIFT correspondences automatically between views to obtain these pairs of color matches. In Figure 2 we show an example of an input image pair.

In both approaches, we use the color correspondences to fit a parametric model of the color transformation between the images. We assume a simple, per channel affine transformation

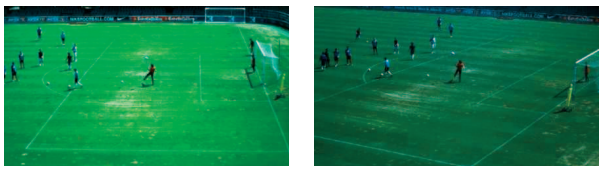


Figure 2: Pair of input images without color correction.

of the form

$$\begin{aligned} R_2 &= a_R R_1 + b_R, \\ G_2 &= a_G G_1 + b_G, \\ B_2 &= a_B B_1 + b_B, \end{aligned}$$

where R_2, G_2, B_2 correspond to the red, green and blue component of the color-corrected images. The fitting is done using least squares method. Then the color transformation is applied to all the input images. A result obtained using $(A1)$ is displayed in Figure 3. For a result obtained using $(A2)$ we refer to the top images in Figure 7.

The use of this simple model is justified since; a) it does not require a color calibration pattern, e.g. Macbeth (this is important if we need to do the color correction in the middle of a match), and b) its computational simplicity and easy interpretation make it easily adjustable by an external operator. Hence it can be integrated in a production system.

More sophisticated linear models can be found in [9, 22, 27]. One could also use histogram matching techniques as in [28, 27].

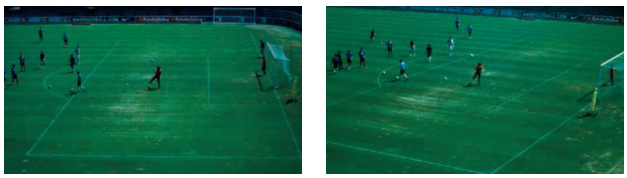


Figure 3: First image (left) corrected to match the colors of the second image, and second image (right) used as reference

2.2 Camera calibration

Camera calibration is a crucial problem for further metric scene measurement. This section presents an overview of the techniques used to calibrate the multicamera setup on a soccer scene using a minimum number of visible field lines or circle [1]. Strategies differ slightly depending on whether or not the camera can be previously calibrated in the laboratory. The camera model we use throughout this work is called the projective model simple "pinhole model" including a model of radial lens distortion.

In case the camera can be precalibrated beforehand in the laboratory, the calibration procedure is as follows:

1. Use a calibration pattern to compute the lens distortion and all the camera intrinsic parameters, including the focal length.

2. Bring the camera to the stadium and take pictures of the pitch (without changing the zoom level).

3. Automatically estimate the position and rotation of the camera in space by detecting white lines and the central circle.

4. If the zoom of the camera is varying, it can be recomputed leaving the other intrinsic parameters constant.

Alternatively, if the camera can not be accessed beforehand in the laboratory the calibration steps are:

1. Take an image of the soccer field using any kind of camera.

2. Default values for the intrinsic camera parameters are assumed, i.e. square pixels and optical center at the center of the image, and calculate the distortion model parameters.

3. Automatically estimate the position, rotation and focal length of the camera in space by detecting white lines and the central circle.

In addition, in case we know that the camera set shares some parameters in common, we run an extra global optimization step to improve the calibration accuracy. The parameters considered are: a common focal length, the height of all cameras, the lens distortion model, the pixel aspect ratio and the optical center.

The accuracy of a camera calibration using the lines and circles present in a soccer field can be affected by various circumstances. We have observed that inaccurate results can be obtained when the number of primitives visible in the images is low, or when the lines are not well drawn in the field, or even because they do not have the proportions and measures they should have officially. Also, the curvature of the field is a source of errors. These failures are particularly relevant in applications such as 3D reconstruction of the scene that require a high precision calibration. Thus we introduced a final calibration enhancement step that uses point matches between different images to improve calibration. The procedure is as follows:

1. An initial calibration of the group of cameras is computed as described above by detecting lines and circles on the soccer field.

2. Points of interest in the scene are automatically detected in each image of the scene using the Harris algorithm [14].

3. Right matchings between the detected 2D points are selected and their 3D point coordinates are reconstructed. The points should be visible in at least two of the cameras and they usually correspond to typically visible points in the scene such as the head of a player or points in the goal.

4. The information from these 3D points is incorporated in the calibration of the cameras minimizing their reprojection error in the different camera views via bundle adjustment [3].

3 3D representation of the players

From the multi-camera system, we get a set of 4 synchronized video sequences observing the scene from different point of views. We now describe how these data are used to compute a 3D reconstruction of the players. Such an estimation is obtained in two steps. First, we automatically segment the players from the field in each camera. Then, we use a multi-view stereo algorithm to estimate the depth of the pixels inside the segmentation masks.

3.1 Background learning and segmentation

In order to estimate the segmentation mask of the players, we rely on background substraction methods. Since we consider fixed cameras, the background, composed of the field and the stadium, can be modeled independently for each camera. We estimate a background image for each camera. To learn this background image, even when the players are on the field, techniques such as running average, Gaussian mixtures [30] or Kernel density estimations [8] are used. Such approaches aim at modeling each pixel of the background using the redundant temporal information of the sequence at its location, while discarding the outliers (i.e. the players). In Figure 4 we show an example of background modeling obtained using the Kernel density estimation method in [8].

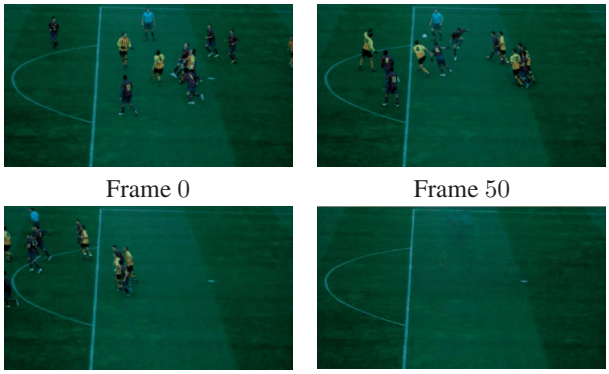


Figure 4: ***Background estimation. The background model is estimated from the different images available for a single camera. We here show such images at frames*** 0***,*** 50 ***and*** 100 ***of a real sequence.***

From the estimated background, a first estimate of the

segmentation mask of the players is obtained by thresholding the difference between the current image and the reference background one. Then, in order to compute the final mask of the players, this estimate is combined with a spatial regularization term using a Graph-cut approach [4]. Some examples of the obtained segmentation masks are given in Figure 5.

Frame 0 Frame 50 Frame100

Figure 5: Segmentation. Illustration of the mask of players obtained at frames 0, 50 ***and*** 100.

3.2 Depth estimation

Given the foreground segmentation masks of all cameras, the goal is now to compute the depth of each pixel in the masks for each frame in the sequence. We describe here the procedure we use to compute depths that are coherent between the different cameras and between the different frames of the sequence.

We define the depth estimation problem as a multi-label problem. We consider a set of candidate depth planes, and associate one of these planes to each pixel inside the segmentation masks. The pixel-to-plane association is realized by minimizing via graph-cuts a suitable energy function.

For reasons of space, we briefly describe the energy, for more details we refer to [26]. To fix our notation, we assume that we have $N \geq 2$ cameras and we consider N color images $I_i : \Omega_i \rightarrow \mathbb{R}^3$, where $\Omega_i \subseteq \mathbb{R}^2$ denotes the domain of I_i, $i = 1, \cdots, N$. We identify Ω_i as the foreground mask of image i. We denote by $\mathcal{I} = \{(i,j) \in \{1, \cdots, N\}^2, i \neq j\}$ the set of pairs of images.

Recall that a plane of the 3D scene induces an homography between each pair of images, and the projections of image points which lie on the plane are matched by this homography. Thus, we may sweep the scene with a family of parallel ordered planes Π_λ, each one labeled by its depth $\lambda > 0$, and only the points of the scene lying on Π_λ will be matched in two images I_i and I_j, $i \neq j$, by the corresponding induced homography $\mathcal{H}^{ij}_\lambda$ (see [15]). The homography $\mathcal{H}^{ij}_\lambda$ can be written as $\mathcal{H}^{ij}_\lambda = (\mathcal{H}^j_\lambda)^{-1}\mathcal{H}^i_\lambda$ where $\mathcal{H}^k_\lambda$ is the homography between the k image plane and Π_λ, $k = 1, \cdots, N$. Obviously, the matching can only be done if the point is visible in both cameras. We assume here that the family of sweeping planes is given.

We want to find the depth, denoted by $\boldsymbol{\lambda}(p)$, associated to each $p \in \Omega_i$, $i = 1, \cdots, N$, and the sets of occluded pixels in each image. We note that these sets of occluded pixels can be estimated from the knowledge of the depth. As the problem will be solved in a discrete framework via a graph cut approach, we assume that $\boldsymbol{\lambda}(p)$ takes its values in a predefined discrete set of possible depths contained in the range $[\lambda_{min}, \lambda_{max}]$.

The pixel-to-plane association is realized by considering an energy function containing five terms:

• a photoconsistency matching cost for couples of corresponding pixels in pairs of images:

$$\mathcal{E}_{\text{ph}}(\boldsymbol{\lambda}) := \sum_{(i,j)\in\mathcal{I}} \sum_{p\in\Omega_i} D(p, \mathcal{H}^{ij}_{\boldsymbol{\lambda}(p)}p) T[\boldsymbol{\lambda}(p) = \boldsymbol{\lambda}(\mathcal{H}^{ij}_{\boldsymbol{\lambda}(p)}p)],$$

where $T[\cdot]$ is 1 if its argument is true and 0 otherwise. This energy considers the scene points $\mathcal{H}^i_{\boldsymbol{\lambda}(p)}p$, with $p \in \Omega_i$, that are visible in both images I_i and I_j. The matching cost $D(p,q)$ measures the difference between the colors $I_i(p)$ and $I_j(q)$ of the pixels $p \in \Omega_i$ and $q \in \Omega_j$ (see [26]).

• a penalty on the occluded pixels that prevents all pixels to become occluded:

$$\mathcal{E}_{\text{occ}}(\boldsymbol{\lambda}) := \gamma \sum_{(i,j)\in\mathcal{I}} \sum_{p\in\Omega_i} T[\boldsymbol{\lambda}(p) > \boldsymbol{\lambda}(\mathcal{H}^{ij}_{\boldsymbol{\lambda}(p)}p)],$$

where $\gamma > 0$ is the occlusion parameter.

If $p \in \Omega_i$ and $q = \mathcal{H}^{ij}_{\boldsymbol{\lambda}(p)}p$, $j \neq i$, the visibility constraint is given by $\boldsymbol{\lambda}(q) \leq \boldsymbol{\lambda}(p)$. This constraint means that a pixel p of an image i can be occluded by a pixel q in image j if and only if the depth of q is smaller than the depth of p. If $\boldsymbol{\lambda}(q) < \boldsymbol{\lambda}(p)$, then the scene point $q = \mathcal{H}^i_{\boldsymbol{\lambda}(p)}p$ is occluded by the scene point $\mathcal{H}^j_{\boldsymbol{\lambda}(q)}q$ in the image I_j. The cost γ penalizes these occlusions. If $\boldsymbol{\lambda}(q) = \boldsymbol{\lambda}(p)$, then the 3D point $\mathcal{H}^i_{\boldsymbol{\lambda}(p)}p$ is visible in both images. In that case we compute the photoconsistency matching cost.

• a term that enforces the visibility constraint to ensure the coherence between the depth maps of the different cameras:

$$\mathcal{E}_{\text{vis}}(\boldsymbol{\lambda}) := U \sum_{(i,j)\in\mathcal{I}} \sum_{p\in\Omega_i} T[\boldsymbol{\lambda}(p) < \boldsymbol{\lambda}(\mathcal{H}^{ij}_{\boldsymbol{\lambda}(p)}p)],$$

where $U \rightarrow \infty$ is a large scalar that prevents the solution to violate the visibility constraint.

• a visual hull constraint that forces the reprojection of the reconstructed points to lie inside the foreground masks of the other images:

$$\mathcal{E}_{\text{vh}}(\boldsymbol{\lambda}) := \beta \sum_{(i,j)\in\mathcal{I}} \sum_{p_i\in\Omega_i} T[\mathcal{H}^{ij}_{\boldsymbol{\lambda}(p)}p \in \Omega_j],$$

where $\beta > 0$.

• a regularization term that forces the neighboring pixels with similar intensity level or color to have similar depth labels:

$$\mathcal{E}_{\text{reg}}(\boldsymbol{\lambda}) := \alpha \sum_{i=1}^{N} \sum_{p_i\in\Omega_i} \sum_{q_i\in\mathcal{N}_{p_i}} F(I_i, p_i, q_i) |\boldsymbol{\lambda}(p_i) - \boldsymbol{\lambda}(q_i)|,$$

where $\alpha > 0$ is the regularization parameter, and $\mathcal{N}_{p_i}$ is 8-neighborhood of p_i. The function $F(I_i, p_i, q_i)$ has the form

$\tilde{F}(|I_i(p_i) - I_i(q_i)|, |p_i - q_i|)$ and weights the discontinuities of the labeling taking into account the image gradient, encouraging depth discontinuities to coincide with edges. The function $\tilde{F}$ is big when its first argument $|I_i(p_i) - I_i(q_i)|$ is small. The second argument $|p_i - q_i|$ enables weighting the influence of pixel q_i, the neighbor of pixel p_i, with respect to their Euclidean distance. The larger $|p_i - q_i|$, the smaller $\tilde{F}$.

The term $|\boldsymbol{\lambda}(p_i) - \boldsymbol{\lambda}(q_i)|$ can be replaced by the Potts model $\min(|\boldsymbol{\lambda}(p_i) - \boldsymbol{\lambda}(q_i)|, 1) = T(\boldsymbol{\lambda}(p_i) \neq \boldsymbol{\lambda}(q_i))$, which does not weight the amplitude of the depth discontinuities. In any of these cases the regularization term (and the whole energy) in an α-expansion move is regular and it can be efficiently minimized using graph-cuts [20].

For reasons of space we have briefly described the energy terms, for more details we refer to [26].

The total energy $\mathcal{E}$ is defined as:

$$\mathcal{E}(\boldsymbol{\lambda}) := \mathcal{E}_{\text{ph}}(\boldsymbol{\lambda}) + \mathcal{E}_{\text{occ}}(\boldsymbol{\lambda}) + \mathcal{E}_{\text{vis}}(\boldsymbol{\lambda}) + \mathcal{E}_{\text{vh}}(\boldsymbol{\lambda}) + \mathcal{E}_{\text{reg}}(\boldsymbol{\lambda}).$$

The energy is minimized for all the cameras simultaneously using graph cuts with α-expansion [19, 26].

For dynamic scenes, in order to enforce the temporal consistency of the depth estimation between successive frames, we add an additional constraint. The idea is to force the estimation at frame $t + 1$ to be similar to the estimation at frame t. We use the optical flow [25] between the captured images, to transfer the depth estimation computed at t to the location of the players at frame $t + 1$. Then, we add a constraint that forces the depth estimation at $t+1$ to be close to this prediction. This constraint could be enhanced through the scene flow computation [31]. Indeed, the scene flow permits considering the variations of the depth of a player between successive frame instants which yields to a more exact the temporal constraint.

Figures 6 and 7 show examples of the depth maps estimated by this procedure in a laboratory and a real scene.

Since the computational cost of the graph-cut labeling method increases with the number of candidate depth planes, it is important to properly choose these candidates to be close to where the players are, in order not to waste candidates where there are no players. In the toy example of Figure 6, the 3 players are spatially close, and it is and we can use a dense depth discretization. Unfortunately, for real examples as the one presented in Figure 7, the players are sparsely distributed around the field. In this case, we consider the visual hull of all the players together in order to define a bounding box that includes all the players. The candidate depth planes are then chosen to be distributed inside this bounding box.

This is clearly not optimal, since it would be better to create a bounding box for each player. Nevertheless, it is good enough, and yield results where each player is represented by more than a single depth plane. This can be seen from the depth discontinuities observable in column (c) of Figure 12. Compared to simpler approaches that represent each player via a single depth plane, our method is able to produce the parallax between different parts of the player, and not only between different players.

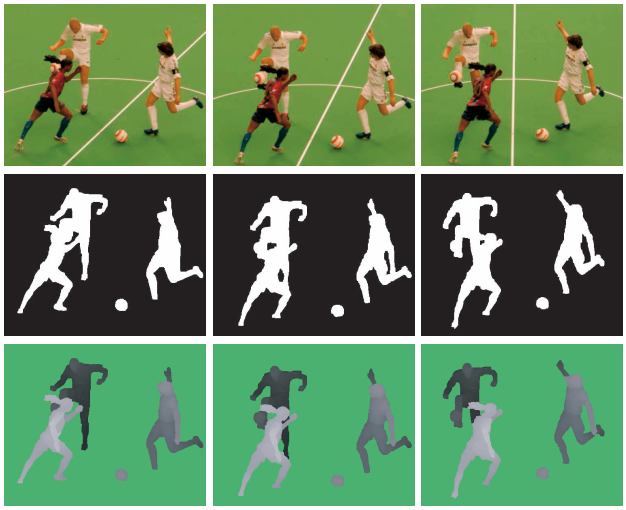


Figure 6: *Depth estimation for a toy example. The original images are presented in the first line for the three cameras. The corresponding segmentations and the depth estimations are then shown in the second and third lines. The depth colors correspond to the depth planes: the darker color, the closer to the cameras. These images are courtesy of Luis Álvarez (AMI group, University of Las Palmas of Gran Canaria) and MEDIAPRO.*

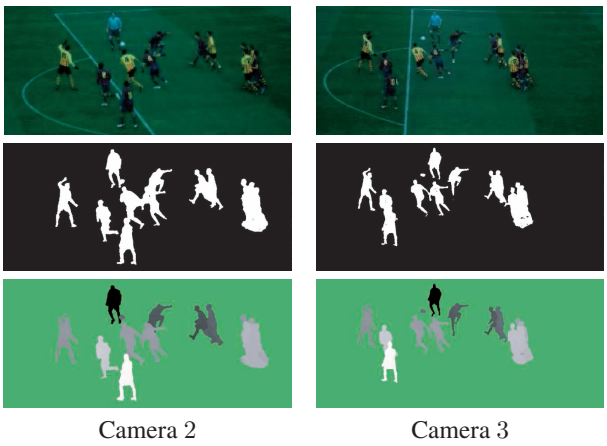


Figure 7: *Depth estimation at frame 50. The original images are presented in the first line for the central real cameras 2 and 3. The corresponding segmentations and the depth estimations are then shown in the second and third lines. The depth colors correspond to the depth planes: the darker color, the closer to the cameras.*

To speed things up, during the development of the method proposed above, we also considered computing the depth of each image independently (i.e, without including visibility constraints between the different estimations) through a plane-sweep approach [6]. This leads to faster algorithms that can be parallelized efficiently on GPU [2]. However, with such

approaches, the consistency between the different estimations is lost and a volumetric approximation of the different depth estimations must be done [32] to prevent from decreasing the accuracy of the results.

4 Virtual view synthesis

Given the depth estimation of the real cameras, we now want to generate a virtual camera observing the scene from a novel point of view. In this section, we describe the process allowing to create such virtual views. First of all, the user define the wanted trajectory of the virtual camera. Then, the corresponding virtual view is synthesized.

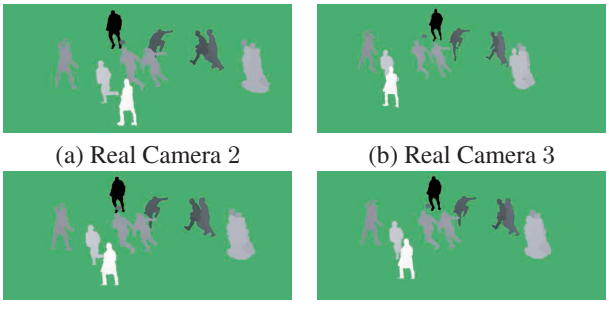
4.1 Virtual calibrations

To synthesize a virtual camera, it is necessary to define its calibration: its position and orientation with respect to the 3D world coordinates. These informations thus need to be given by a user. In practice, we have proposed to automatically generate virtual mappings between real cameras. More precisely, we have addressed two applications. The first one consists in creating a virtual camera that warps the real cameras at a fix frame. This leads to a time freezed sequence equivalent to the matrix effect. The second application considers a virtual camera moving along frames, observing the dynamic scene from different point of views.

4.2 Image synthesis

Once the calibration of the virtual camera has been defined, we use this information and the computed depth maps to synthesize the corresponding virtual image. To this end, we use a two step method related to the one presented in [29]. It is based on the estimation of the depth map of the virtual camera and a further estimation of the colors of the virtual image.

In the first step, the virtual camera depth map is obtained by forward mapping the depth estimated for each real camera to the virtual camera. An example of such transfer is presented in Figure 8.



(a) Real Camera 2 (b) Real Camera 3

(c) Virtual Camera 1 (d) Virtual Camera 2

***Figure 8:** Creation of the virtual depth at frame 50. The depth estimated on the real cameras (a-b) are transferred into two virtual cameras (c-d).*

In the second step, the colors of the virtual image are computed using by backward mapping the pixels from the virtual camera to the real ones with respect to their depth. A mean of the colors obtained from the different real cameras is used in order to smooth the virtual image and avoid artefacts coming from potential depth estimation errors.

Finally, the pixels outside the virtual players are processed using three particular depth planes, as illustrated in Figure 9.

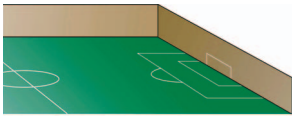
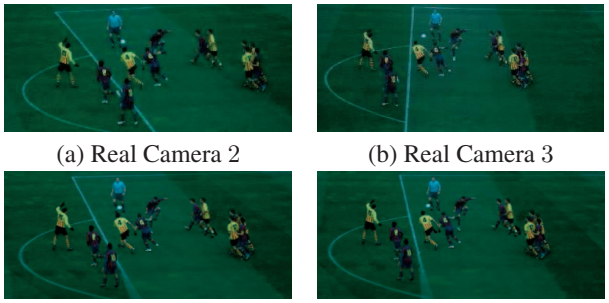


***Figure 9:** Background model. Three specific planes are considered for background synthesis: the groundfield, the goal and the stands.*

More precisely, we use a first plane to model the ground. We assume the groundfield to be flat, which is not true in reality and does yield some blurring effects in the non flat parts of the groundfield. We use two more planes to model the goal and the stands. These form a very rough approximation of the real geometry. The real images backprojected into this planes clearly do not match. To palliate the visual artifacts of this approximation we use view dependent texture mapping [7] so that only the camera that is closer to the virtual camera is used to produce the texture of these planes. When moving from one real camera to another, a short transition zone is used, and the images produced by both cameras are bended to avoid a sudden change in the texture. An example of synthesis including the goal and the stands is shown in Figure 10.

***Figure 10:** Virtual views from a moving camera observing the dynamic scene. The goal and the stands have been synthetized using the specific planes illustrated by the Figure 9.*

We present in Figure 11 some synthesized virtual images. Note that we do not directly transfer the colors from the real cameras to the virtual one. Indeed, such a direct approach would generate non integer coordinates and require heavy interpolations to provide a good synthesis. In general, the synthesis gives better results when dealing with back-transfer of colors (see for instance the inverse trifocal tensor of [23]).



(a) Real Camera 2 (b) Real Camera 3

(c) Virtual Camera 1 (d) Virtual Camera 2

Figure 11: ***Synthesis of the virtual images at frame*** 50. ***The color information of the real cameras (a-b) is transferred into two virtual cameras (c-d).***

5 Post-processing

The synthesized virtual images can contain some visual artifacts due to depth estimation errors, occluded parts of the field or temporal inconsistency. In order to correct them, we propose to use two post-processing tools: a spatial correction of the image artifacts through image inpainting and a temporal filtering of the inpainted images along the trajectories of the points of the scene, using the optical flow of the virtual inpainted sequence.

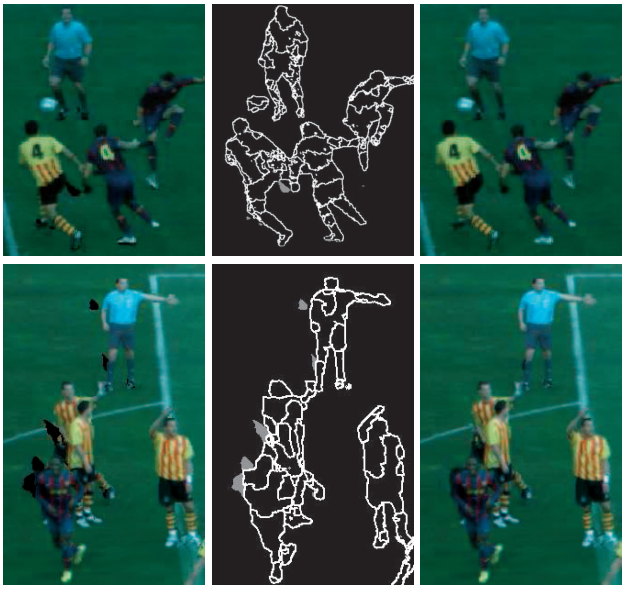
5.1 Image inpainting

Image inpainting methods aim at repairing the missing or erroneous areas of an image. Such processes are thus suitable to correct eventual visual artifacts in the virtual synthetic images generated by the procedure described in the previous section. Indeed, we observed some visual artifacts on the boundaries of depth discontinuities, where the errors of depth estimation are more frequent. Moreover, the areas of the field that are occluded by the players lead to missing parts on the synthetic images. Fortunately, from the estimated synthetic depth maps, we are able to define with enough accuracy the areas of the synthetic images we want to correct (i.e. areas around the depth discontinuities). Examples of such areas are shown in Figure 12.

(a) (b) (c)

Figure 12: ***Creation of the inpainting masks. (a) Synthetic images. (b) Virtual depth maps. (c) Areas to inpaint.***

The correction is then performed with an image inpainting algorithm [5] based on image color patches comparison and copy. The method also relies on search trees (*kd-trees*), in order to speed up the search of similar patches. In Figure 13, we present examples of inpainting correction.



(a) (b) (c)

Figure 13: ***Image inpainting. (a) The synthetic images. (b) The corresponding areas to inpaint composed by the union of the missing part of the field (in gray) and the virtual depth discontinuities areas (in white). (c) Final inpainted images.***

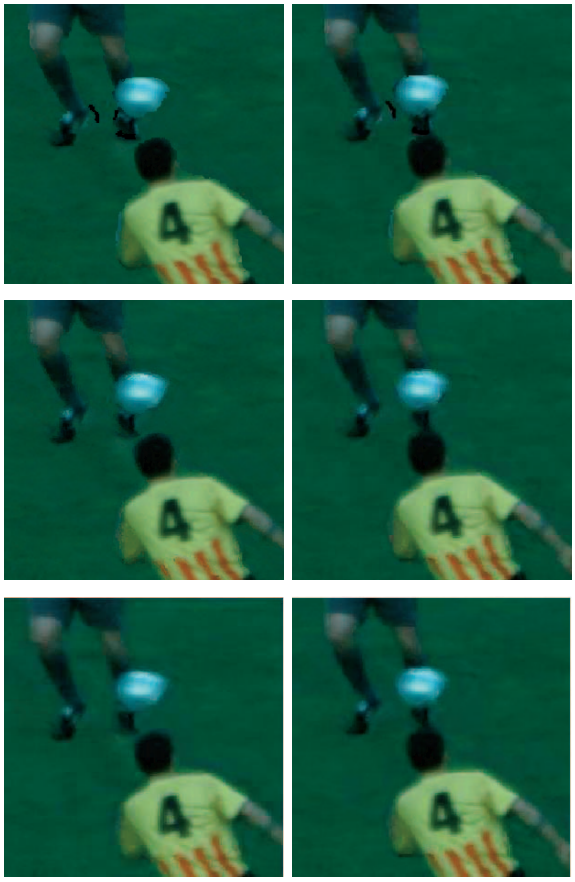
5.2 Temporal filtering

The inpainted images are finally filtered temporally. To this end, we realize a weighted mean of the colors along pixel trajectories. Trajectories are first obtained through the computation of the optical flow [25] on the inpainted sequence. Hence, the value of the color of a pixel is filtered by considering a temporal window along its trajectory. The values are in practice weighted with respect to the inverse of the norm of the optical flow. Indeed, when a pixel is moving fast, there is no need to filter its color value since it will not imply temporal artifacts. On the contrary, when a pixel corresponds to a nearly static 3D point of the scene, the color value must be filtered in order to smooth the synthesis and obtain some temporal consistency. As illustrated by Figure 14, this process allows to smooth the inpainted images in order to attenuate the potential temporal artefacts.

6 Experimentation details

In this section, we give some details on the experimentations we realized. We provide two videos[1] corresponding to two different sequences. The first one presents a time freezed virtual cameras warping two real cameras. The second video shows a dynamic virtual camera observing the moving game.

[1] available on `http://sites.google.com/site/nicolaspapadakis/videos_cvmp`



(a) Virtual frame t (b) Virtual frame $t+1$

***Figure 14:** Temporal filtering. A crop of the result is shown for two consecutive virtual frame time t (column a) and $t+1$ (column b). The original synthesized images are presented in the first line. The inpainting results are shown in the second line. The third line illustrates the final filtered images.*

The videos have been obtained automatically without user interaction or parameter tuning.

These results have been obtained using a standard desktop computer, running linux, with an Intel Core2 Quad CPU, and a NVIDIA GTX280 graphics card. The process required between 5 and 10 minutes of a single core per frame depending on the size of the players. Most of this time was spend computing the depth maps.

Indeed, half of the intern processes are real time: segmentation, synthesis and filtering. In our current implementation, these programs spend more time for input reading and output writing than the processes themselves. However, the depth estimation, optical flow and inpainting steps have a huge computational cost, especially considering the size of the processed data that are FullHD images with a size of 1920x1080 pixels.

Let us discuss these three algorithms more precisely. Concerning the depth estimation, as we impose visibility constraints between the different cameras in a graph representation, the graph include some complex connexions. Moreover, as mentioned in subsection 3.2, the definition of the set of possible depth planes is not optimized. Considering a reduced and accurate set of depth planes would improve the estimation and reduce the computational time of the process. When estimating the depth independently for each camera, the process is real time, but the results presents a poor quality. A merging of the different depths is then necessary to filter these noisy estimations.

Even if nearly-real time optical flow estimators exist in the literature, we investigated a different approach. The optical flow computation is done through a parallelized implementation of a convexified energy [25]. Such a process is potentially real time, but our current implementation takes up to 30 seconds our NVIDIA graphic card.

Finally, the last time consuming process is the inpainting step. Originally, we used a simple patch-based approach, looking for similar patches in the original images. This first method presented a very high computational cost due to the size of the images (a huge number of patches have to be tested for each pixel to inpaint). Hence, the use of *kd-trees* allowed us to speed-up the search process and reduce the inpainting running time. The consuming part now comes from the construction of the tree of possible candidate patches, which is done on each frame instant independently. Defining a common global tree from a database of possible patches would allow to reduce drastically the computational cost of the inpainting step. We are also currently investigating the implementation of search trees in GPU in order to speed-up even more the search part of the process.

7 Conclusion

In this paper, we have presented a framework for the automatic synthesis of virtual views in the case of soccer games observed by a system of 4 cameras. For the future, the different steps of the process will be optimized, mainly in terms of computational cost. One other important point concerns the field, which is assumed to be flat. Better approximations including the field curvature should be taken into account to enhance the calibration and the synthesis steps. Note also that the present work only uses the information contained in the acquired images. To improve the visual quality of the synthesized results, we plan to couple the synthesis of players with a predefined virtual model of the stadium and the goal.

Acknowledgements

This work was partially funded by Mediapro through the Spanish project CENIT-2007-1012 i3media and by the Centro para el Desarrollo Tecnológico Industrial (CDTI), within the Ingenio 2010 initiative. Aurélie Bugeau, Olivier D'Hondt, Pau Gargallo and Nicolas Papadakis acknowledge support from the Torres Quevedo program of the Ministerio de Educación y Ciencia in Spain. Vicent Caselles also acknowledges partial support by MICINN project, reference MTM2009-08171, by

GRC reference 2009 SGR 773 and by "ICREA Acadèmia" prize for excellence in research, the last two funded by the Generalitat de Catalunya.

References

[1] L. Álvarez and V. Caselles. Procedimiento para la calibración de una cámara de video y tv. *Spanish Patent No. EP-0140. Patent Pending.*, 2009.

[2] A. Baeza, P. Gargallo, V. Caselles, and N. Papadakis. A narrow band method for the convex formulation of discrete multi-label problems. *Preprint, available at* `http://www.dtic.upf.edu/~vcaselles/technicalreport2Web.pdf`, 2010.

[3] T. Bill, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In *Vision Algorithms: Theory and Practice*, volume 1883, pages 298–372, 2000.

[4] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proc. IEEE Int. Conf. on Comp. Vis. (ICCV'01)*, volume 1, pages 105–112, 2001.

[5] A. Bugeau, M. Bertalmío, V. Caselles, and G. Sapiro. A comprehensive framework for image inpainting. *to appear in IEEE Trans. on Image Processing*, 2010.

[6] R.T. Collins. A space-sweep approach to true multi-image matching. In *Proc. IEEE Comput. Vis. and Pat. Recogn. (CVPR '96)*, pages 358–363, 1996.

[7] P. Debevec, Y. Yu, and G. Boshokov. Efficient view-dependent image-based rendering with projective texture-mapping. Technical report, University of California at Berkeley, 1998.

[8] A.M. Elgammal, D. Harwood, and L.S. Davis. Non-parametric model for background subtraction. In *Proc. European Conf. on Comput. Vis. (ECCV'00)*, volume 2, pages 751–767, 2000.

[9] G.D. Finlayson, S.D. Hordley, and R. Xu. Convex programming colour constancy with a diagonal-offset model. In *IEEE Int. Conf. on Image Processing*, page 948951, 2005.

[10] O. Grau, A. Hilton, J. Kilner, G. Miller, T. Sargeant, and J. Starck. A free-viewpoint video system for visualisation of sport scenes publication details. *BBC Research White Paper WHP 142*, 2006.

[11] O. Grau, GA Thomas, A. Hilton, J. Kilner, and J. Starck. A Robust Free-Viewpoint Video System for Sport Scenes. In *3DTV Conference, 2007*, pages 1–4, 2007.

[12] J.Y. Guillemaut, A. Hilton, J. Starck, J. Kilner, and O. Grau. A bayesian framework for simultaneous matting and 3d reconstruction. In *3-D Digital Imaging and Modeling, 2007. 3DIM'07. Sixth International Conference on*, pages 167–176, 2007.

[13] J.Y. Guillemaut, J. Kilner, and A. Hilton. Robust graphcut scene segmentation and reconstruction for free-viewpoint video of complex dynamic scenes. In *IEEE International Conference on Computer Vision*, 2009.

[14] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vis. Conf.*, pages 147–151, 1988.

[15] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press New York, NY, USA, 2003.

[16] K. Hirakawa and T.W. Parks. Adaptive homogeneity-directed demosaicing algorithm. *IEEE Trans. on Image Processing*, 14:360–369, 2005.

[17] N. Inamoto and H. Saito. Fly through view video generation of soccer scene. In *Entertainment computing: technologies and applications*, page 109. Springer, 2002.

[18] N. Inamoto and H. Saito. Immersive observation of virtualized soccer match at real stadium model. In *The Second International Symposium on Mixed and Augmented Reality (ISMAR03)*, pages 188–197. Citeseer, 2003.

[19] V. Kolmogorov and R. Zabih. Multicamera scene reconstruction via graph cuts. In *Proc. European Conf. on Comput. Vis. (ECCV'02)*, pages 82–96, 2002.

[20] V. Kolmogorov and R. Zabih. What Energy Functions Can Be Minimized via Graph Cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147, 2004.

[21] T. Koyama, I. Kitahara, and Y. Ohta. Live mixed-reality 3d video in soccer stadium. In *International Symposium on Mixed and Augmented Reality*, pages 178–186, 2003.

[22] P. Latorre Carmona, R. Lenz, F. Pla, and J. Sotoca. Affine Illumination Compensation for Multispectral Images. *Image Analysis*, pages 522–531, 2007.

[23] H. Li and R. Hartley. Inverse tensor transfer with applications to novel view synthesis and multi-baseline stereo. *Signal Processing: Image Communication*, 21(9):724 – 738, 2006.

[24] Y. Ohta, I. Kitahara, Y. Kameda, H. Ishikawa, and T. Koyama. Live 3D video in soccer stadium. *International Journal of Computer Vision*, 75(1):173–187, 2007.

[25] N. Papadakis, A. Baeza, P. Gargallo, and V. Caselles. Polyconvexification of the multi-label optical flow problem. In *Proc. IEEE Int. Conf. on Image Processing (ICIP'10)*, 2010.

[26] N. Papadakis and V. Caselles. Multi-label depth estimation for graph cuts stereo problems. *To appear in Journal of Mathematical Imaging and Vision*, 2010.

[27] F. Pitie and A. Kokaram. The linear Monge-Kantorovitch linear colour mapping for example-based colour transfer. In *4th European Conference on Visual Media Production, 2007. IETCVMP*, pages 1–9. Citeseer, 2007.

[28] F. Pitié, A.C. Kokaram, and R. Dahyot. Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding*, 107(1-2):123–137, 2007.

[29] J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In *SIGGRAPH '98: Proc. of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, 1998.

[30] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. IEEE Comput. Vis. and Pat. Recogn. (CVPR '99)*, volume 2, pages 252–258, 1999.

[31] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):475–480, 2005.

[32] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l1 range image integration. In *Proc. IEEE Int. Conf. on Comp. Vis. (ICCV'07)*, 2007.