Virtual camera synthesis for soccer game replays

N. Papadakis[*][‖], A. Baeza[†], A. Bugeau[‡], O. D'Hondt[†], P. Gargallo[†], V. Caselles[§]
X. Armangué[¶], I. Rius[¶], S. Sagàs[¶]

[*]CNRS, Laboratoire Jean Kuntzmann, 51 rue des Mathématiques, 38041 Grenoble, France
email: `nicolas.papadakis@imag.fr`

[‖]Moise team (INRIA Rhône-Alpes/LJK)

[†]Fundació Barcelona Media, Avinguda Diagonal 177, 08018 Barcelona, Spain
email: {`antonio.baeza,olivier.dhondt,pau.gargallo`}`@barcelonamedia.org`

[‡]LABRI, 351 cours de la Libération F-33405 Talence, France
email: `aurelie.bugeau@labri.fr`

[§]Universitat Pompeu Fabra, Carrer de Roc Boronat 138, 08018 Barcelona, Spain
email: `vicent.caselles@upf.edu`

[¶]Mediapro, Avinguda Diagonal 177, 08018 Barcelona, Spain
email: {`xarmangue,irius,ssagas`}`@mediapro.es`

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.

First presented at the 8th European Conference on Visual Media Production 2010, extended and revised for JVRB

1 Introduction

The creation of virtual viewpoints in soccer games is an interesting problem for two main reasons: first, for its large base audience and potential industrial impact and second, for the many scientific challenges it presents. Hence, a collaboration between industries and universities 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 development of post-production tools facilitating the change of viewpoint for sport scenes has received much attention in the last years. The Eyevision system [KNR95, KOS$^+$01] is probably the first practical attempt to give the possibility of observing a dynamic sport scene from different angles. The viewpoint is changed simply by selecting one of the multiple possible views that are recorded by a set of cameras pointing to the same target and with homogenized properties (zoom, focus, pan and tilt); no intermediate, novel views are generated. Some commercial products with higher capabilities have been developed since then. Piero[1] is limited to a single camera input, thus restricting the position of the virtual camera. LiberoVision[2] and SportsVision[3] offer semi-automatic systems that let an expert user to freely move a virtual camera, while only using the existing broadcast cameras as input. The IMEC Virtual Camera[4] is a fully automatic system, but requires closer camera positions than the standard broadcasts set ups. Albeit the existence of these commercial solutions, the generation of novel viewpoints is still an active field, as demonstrates the amount of research projects devoted to the topic. We can mention, for example, the European Projects FINE[5] and FASCINATE[6] and the iView project leaded by BBC[7].

The technical challenges The creation of virtual view for soccer games requires to address numerous problems. Cameras are capturing outdoor environments, often in uncontrolled and changing lighting conditions. Also, they are located in a wide-baseline disposition, and may be moving. Besides the calibration problems in such environments, the different cameras may have a different color response that has to be normalized, and the relatively low resolution and the complexity of the scene are considerable. Indeed, two teams of 11 players dressed alike playing in a very large arena outdoors and bumping into each other, added to several referees running freely in the field creates all kinds of occlusions. It is therefore not a simple scenario.

Although all these problems have to be addressed in order to solve the problem of view synthesis for sport events, in particular for soccer games, we shall assume that the camera calibration and color normalization have been solved and we concentrate on the problem of depth computation, view synthesis and its post-processing (filtering and inpainting) without forgetting the problem of background subtraction.

Let us briefly review the main approaches to multi-view scene reconstruction.

Related work on multi-view scene reconstruction We focus essentially on techniques that allow a 3D reconstruction of the scene since they are more adapted to free view-point video rendering, which is our main interest in this paper.

The 3D reconstruction of a shape from its binary projected masks, or silhouettes, in the different views is called shape from silhouette. The interest of this technique lies in the fact that it provides a fast and simple 3D reconstruction, not as accurate as the one obtained by multi-view stereo, but sufficient for some computer vision applications such as human motion analysis or 3D localization. It may also be useful as a first approximation to a more accurate reconstruction. Traditionally [Bau74, Lau94], the shape is com-

[1]`http://www.bbc.co.uk/rd/projects/virtual/piero`

[2]`http://www.liberovision.com`

[3]`http://www.sportsvision.org`

[4]`http://www2.imec.be`

[5]`http://www.projectfine.eu`

[6]`http://www.fascinate-project.com`

[7]`http://www.bbc.co.uk/rd/projects/iview`

puted as the maximum volume consistent with all the silhouettes, the Visual Hull. In real applications the visual hull may present errors due to occlusions, moving background, illumination changes, color similarities between foreground and background or calibration errors. Several methods have been recently proposed in order to make the computation of the visual hull robust to missing parts and to segmentation errors [FB05, SVZ00, LPC08, HP10].

Another useful approximation may be obtained by using photo-hulls [SD99, KS00] defined as the maximal volume which is photo-consistent with the set of input views. Although this technique may produce noisy reconstructions (specially when we do not have many images) the result obtained may be useful as a first approximation to be refined with other methods that we will explain below.

The problem of free viewpoint visualization in the case of sport scenes presents a set of characteristics that require the development of accurate and robust methods matching the quality required for retransmission. The wide baseline disposition of the cameras in an environment with non controlled illumination and the distance to the players may generate occlusions and the violation of the ordering constraint when viewing players from different cameras. Efficient numerical techniques for multi-view stereo reconstruction that incorporate visibility constraints are based on the use of graph cuts [BVZ98, BVZ01, KZ01, KZ02, KZ04] or belief propagation [SZS03, TF03]. They permit to obtain state of the art results. Other powerful methods are based on the computation of reliable correspondences using correlation methods, forming a set of seed points that are later increased to neighboring areas until they arrive to a dense reconstruction [HK07, FP10]. In [GCS06] the authors follow a similar strategy by computing individual depth maps using a window-based voting approach that returns only good matches, merging the depth maps into a single mesh using a volumetric approach. Let us mention the work [SH07] where the combination of stereo matching cues with visual hull information in an energy based formulation permits to obtain good results in cases where there is little texture or strong variation in appearance, as it may be the case when the cameras are in a wide baseline disposition. We refer to [SCD$^+$06] for a survey of multi-view reconstruction methods.

Other strategies based on joint segmentation and depth estimation are also adapted to the case of sport events, since it may be convenient not to explicitly reconstruct the background, using instead a virtual one onto which real textures are projected. The background is indeed difficult to compute it if its texture is poor. In the case of soccer, images can then be separated into foreground layers (corresponding to players) and a background one. This is the approach followed in [GKH09, GH11] and [BBPP10]. This strategy was introduced in [KCB$^+$06] for background substitution, considering thus only two layers. Let us also mention the work [ZKU$^+$04] which pioneered the use of layered representations for disparity computation and used it for free-viewpoint video interpolation in the case of a narrow baseline camera configuration. Similar recent systems produce very good results in real-time [LLB$^+$10, JNS10].

From the exhaustive list of works dealing specifically with sport scenes, we can mention the ones using billboards that produce a flat 3D reconstruction of a scene [GHK$^+$06, GTH$^+$07, GHS$^+$07, KKO03, OKK$^+$07]. In order to finally obtain realistic volumetric reconstructions, more sophisticated tools based on visual hulls or photohulls have been proposed [GHK$^+$06, GTH$^+$07, GHS$^+$07]. A last class of works, closely related to the method we propose, concerns the use of Graph-Cut to optimize an energy function combining multiple image cues with strong priors. Such an approach has been successfully applied to depth computation [GHS$^+$07] and joint segmentation and depth estimation [GKH09, GH11]. In all these works, the separation of the soccer field and the players has been is a fundamental step in order to speed up and facilitate the computation of correspondences [GHK$^+$06, GTH$^+$07, GHS$^+$07, GKH09, GH11, IS02, IS03].

Thus, there are still several aspects of the problem that require improvements to generate quality videos for retransmission. To get a correct parallax effect we need to compute a (sufficiently) dense depth map of the players in the scene. The complexity of the scenes then requires the use of depth computation or stereo reconstruction methods that take into account the visibility constraints and the handling of occlusions. In the case of video sequences synthesis, one also needs to compute time coherent depths or at least to guarantee a time coherent synthesis. As there may be still a lack of precision around boundaries of players, leading to holes in the synthetic views, a post-processing step is therefore necessary. Such holes have to be filled-in with inpainting methods and the eventual temporal artifacts should be treated with a suitable video filtering.

Let us finally mention that from the works quoted [GKH09, GH11] are the closer to our approach, the main difference being that those authors consider a joint segmentation and depth estimation approach while we consider both stages separately. On the other hand, our depth estimation energy takes into account occlusion and visibility constraints.

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 subtraction. 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.

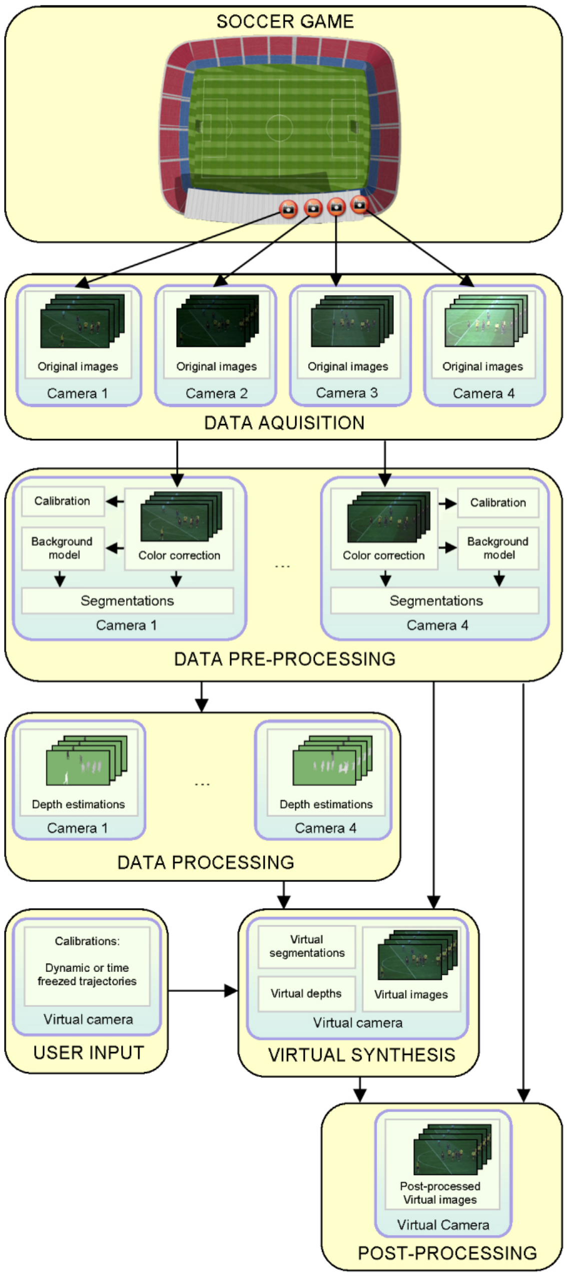


Figure 1: Description of the overall process.

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. The cameras were placed on the stands on one side of the field, at the same height, separated 10 meters from each other and looking to one goal (see the top view in Figure 1). 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.
- Windows XP Professional Service Pack 3 OS.

Regarding the acquisition procedure, we first 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 produces a raw 8-bit 1920x1080 video stream at 25 fps, which is stored unprocessed in a fast hard drive. This results in a data bandwidth of 50 Mbytes/sec. Next, the Adaptive Homogeneity-Directed (AHD) Bayer demosaicing algorithm [HP05] 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 with more detail the color correction and calibration methods. Figure 2 shows an example of an input image pair. We use

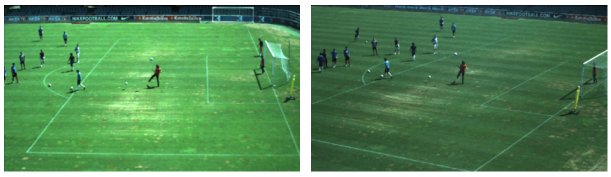


Figure 2: Pair of input images without color correction.

the color correspondences computed via sift matching [Low04] to fit a parametric model of the color transformation between the images. We assume a simple, per channel affine transformation 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, whose result is depicted in Figure 3.

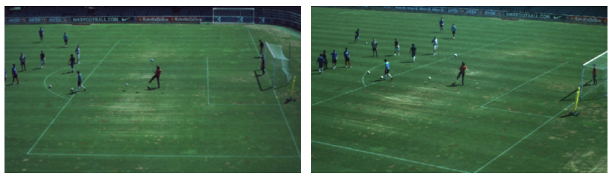


Figure 3: First image (left) corrected to match the colors of the second image, and second image (right) used as reference.

2.1 Camera calibration

Camera calibration is a crucial problem for further metric scene measurement. This section presents an overview of the techniques we used to calibrate the multicamera setup on a soccer scene using a minimum number of visible field lines or circles [AC09]. For related strategies we refer to [LL05] where the authors also exploit the information contained in the central circle, the central line and lateral lines in order to compute intrinsic and extrinsic parameters, to [YZ07] which needs five views of the scene in order to compute this set of parameters, or to [SCG01] which requires to view at least four lines near the goal area.

Strategies differ slightly depending on whether the camera can be previously calibrated in the laboratory or not. In this work we use the simple pinhole camera model, including radial lens distortion.

If 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 [AC09].
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 with the camera system.
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 [AC09].

In addition, in case we know that the cameras share some parameters, 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:

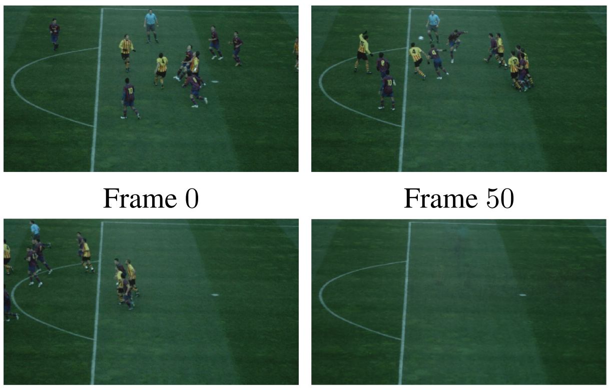
1. An initial calibration of the group of cameras is computed as described above by detecting lines and circles on the soccer field.
2. Points of interest in the scene are automatically detected in each image of the scene using the Harris algorithm [HS88].
3. Right matchings between the detected 2D points 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 the goals.
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 [TMHF00].

3 3D representation of the players

From the multi-camera system, we get a set of $M = 4$ 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 subtraction methods. Since we consider fixed cameras, the background, composed by the field and the stadium, can be modeled independently for each camera. We estimate a background image B_i for each camera $i = 1, \cdots, M$. To learn this background image, even when the players are on the field, techniques such as running average, Gaussian mixtures [SG99] or Kernel density [EHD00] estimations 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). An example of background modeling is shown in Figure 4.



Estimated background

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 estimation of the background, the segmentation mask of the players Ω_i, $i = 1, \cdots, M$, is finally obtained by thresholding the difference between the current image and the reference background one. This difference is computed on each pixel and completed with a spatial regularization of the masks.

Denoting as B_i^j the background image of camera i for the color channel j and I_i^j the current image of camera i for the color channel j, we considered the following data cost to segment players:

$$c_i(x) = \sum_j \left| \frac{I_i^j}{\sum_k I_i^k} - \frac{B_i^j}{\sum_k B_i^k} \right| + \beta \sum_j Max(0, I_i^j(x) - B_i^j(x)).$$

In the first term, we considered normalized colors in order to take into account the possible illumination changes that may occur along the game, namely with weather variation, nightfall and spot lightning. The second term of the data cost, weighted by the parameter $\beta \geq 0$, allows considering the moving shadows in the field. Indeed, in the players shadow areas, the grass color of the current image I_i will be darker than the one of the reference background image B_i. With this model, the data cost does not penalize such shadow areas.

The binary segmentation mask of players Ω_i is finally obtained using a Graph-Cut approach [BJ01]:

$$\Omega_i = \arg\min_S \int S(x) Max(\alpha - c_i(x), 0) + \gamma \int |\nabla S|, \tag{1}$$

where the parameter $\alpha \geq 0$ is used for thresholding the data cost and $\gamma \geq 0$ weights the influence of the regularization term that allows discarding the small areas from the segmentation. In order to justify this segmentation model, we realized some comparisons with ideal segmentations made by hand. More precisely, we looked at N_1, the number of false positives, and N_2, the number of pixels not detected. Denoting as N_0 the number of pixels contained in the manual segmentation, we have computed the percentages of over- and sub-segmentations for different segmentation models (corresponding to different values of β and γ, for $\alpha = 0.1$). In these experiments, each of the 5 ideal segmentations tested contains around $N_0 = 50000$ pixels to detect. The results, illustrated in Table 1 , show the good performance of the proposed model. Indeed, we obtain small over-segmentation rates N_1/N_0, which is interesting to have fast depth estimations. Moreover, we also have small subsegmentation rates N_2/N_0 with the full model, so that it ensures good quality of the estimations. Note that most of these last errors occur at player's boundaries and will not affect the final results thanks to the inpainting post-processing step.

In Figure 5, some illustrations of the obtained segmentation masks are given.

	$\beta = 0$ $\gamma = 0$	$\beta = 1$ $\gamma = 0$	$\beta = 0$ $\gamma = 0.1$	$\beta = 1$ $\gamma = 0.1$
N_1/N_0	3.8%	57.1%	4.6%	9.2%
N_2/N_0	24,1%	7,1%	14,5%	7,9%

Table 1: Rates of false positives N_1 and pixels not detected N_2 with respect to the N_0 ideal pixels segmented by hand on test examples.

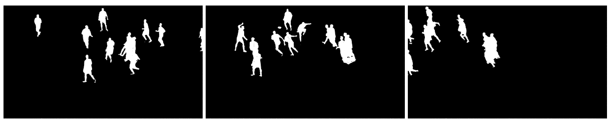


Figure 5: Segmentation. Illustration of the mask of players obtained at frames 0, 50 and 100.

3.2 Depth estimation

Given the foreground segmentation masks of all cameras, the goal is now to compute, for each frame t in the sequence, the depth of each pixel in the set $\Omega_i(t)$. 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.

The depth estimation problem is defined 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 a suitable energy function [KZ02, KZ04] via Graph-Cut.

We now describe the energy, for more details we refer to [PC10]. To fix our notation, we assume that we have $M \geq 2$ cameras and we consider M color images $I_i : \Omega_i \to \mathbb{R}^3$, where $\Omega_i \subseteq \mathbb{R}^2$ denotes the domain of I_i, $i = 1, \cdots, M$. We identify Ω_i as the foreground player mask of image i. We denote by $\mathcal{I} = \{(i,j) \in \{1, \cdots, M\}^2, i \neq j\}$ 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 $\lambda > 0$. The points of the scene lying on Π_λ will only be correctly matched in two images I_i and I_j, $i \neq j$ with the corresponding induced homography $\mathcal{H}_\lambda^{ij}$ (see [HZ03]). The homography $\mathcal{H}_\lambda^{ij}$ can be written as $\mathcal{H}_\lambda^{ij} = (\mathcal{H}_\lambda^j)^{-1}\mathcal{H}_\lambda^i$ where

$\mathcal{H}_\lambda^k$ is the homography between the image k and the plane and Π_λ. Obviously, the matching can only be done if the point is visible in both cameras.

We want to find the depth, denoted by $\boldsymbol{\lambda}(p)$, associated to each $p \in \Omega_i$, $i = 1, \cdots, M$, 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 $[\lambda_{min}, \lambda_{max}]$. We assume that the family of sweeping planes is given with respect to a reference camera. The planes representation is illustrated in figure 6.

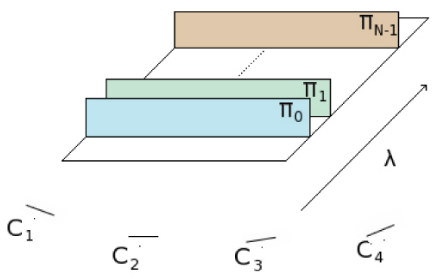


Figure 6: Representation of the sweeping planes. The N planes used to represent the depth are defined from the reference camera C_2.

Remember that the depth estimation must respect the visibility constraint. 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.

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

$$T[x] = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise.} \end{cases}$$

This energy concerns 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 [PC10]).

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

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

Following the visibility constraint, if $\boldsymbol{\lambda}(q) < \boldsymbol{\lambda}(p)$, then the scene point $X_p = \mathcal{H}^{i}_{\boldsymbol{\lambda}(p)}p$ is occluded by the scene point $X_q = \mathcal{H}^{j}_{\boldsymbol{\lambda}(q)}q$ in the image I_j. The occlusion parameter $\gamma > 0$ 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 and the photoconsistency matching cost is taken into account.

• 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 $U \to \infty$ 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 $\tilde{F}(|I_i(p_i) - I_i(q_i)|, |p_i - q_i|)$. It weights the discontinuities of the labeling by 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 the neighbor q_i 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. The total energy $\mathcal{E}$ is defined as:

$$\mathcal{E}(\boldsymbol{\lambda}) := \mathcal{E}_{\text{ph}}(\boldsymbol{\lambda}) + \mathcal{E}_{\text{occ}}(\boldsymbol{\lambda}) + \mathcal{E}_{\text{vis}}(\boldsymbol{\lambda}) + \mathcal{E}_{\text{vh}}(\boldsymbol{\lambda}) + \mathcal{E}_{\text{reg}}(\boldsymbol{\lambda}). \quad (2)$$

In any of the previously described cases the whole energy is regular [KZ04], and it can be efficiently minimized in an α-expansion move using Graph-Cut [KZ02, PC10].

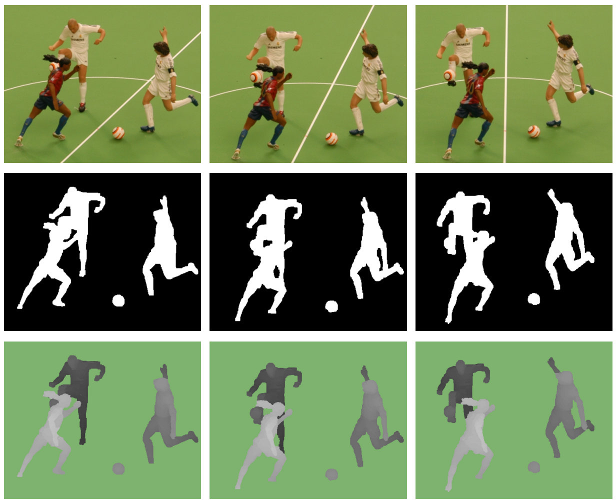


Figure 7: Depth estimation on a toy example. The original images are presented in the first row for three cameras. The corresponding segmentations and the depth estimations are then shown in the second and third rows. The depth colors correspond to the depth planes: the lighter color, the closer to the cameras. These images are courtesy of Luis Álvarez (AMI group, University of Las Palmas of Gran Canaria) and MEDIAPRO.

Figures 7 and 8 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. They have 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 7, the 3 players are spatially close, and we can use a dense depth discretization. Unfortunately, for real examples as the one presented in Figure 8, 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 yields results where each player is represented by more than one depth plane. This can be seen from the depth discontinuities observable in column (c) of Figure 13. 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.

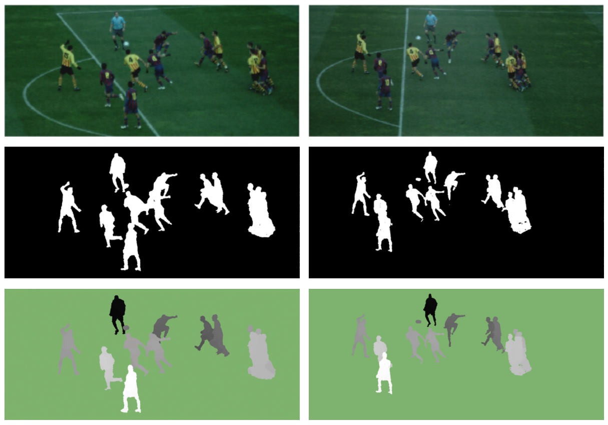


Figure 8: Depth estimation at frame 50. The original images are presented in the first row for the central real cameras 2 and 3. The corresponding segmentations and the depth estimations are then shown in the second and third rows. The depth colors correspond to the depth planes: the lighter color, the closer to the cameras.

3.3 Discussion on velocity and accuracy

In term of computational cost, our process provides results in almost 1 minute for a set of 20 depth planes and about 50000 pixels in each of the four masks Ω_i. 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 [Col96]. This leads to fast algorithms that can be parallelized efficiently on GPU and almost reaching real time estimation. However, with such approaches, the consistency between the different estimations is lost and an additional volumetric approximation of the different depth estimations must be done [ZPB07] to prevent from decreasing the accuracy of the results. This volumetric approximation can also be parallelized. However, we observed that this nearly real time reconstruction does not give accurate results with our $4-$camera setup. The visibility constraints are then necessary to ensure good depth estimations with a small number of cameras.

Note that the accuracy of our depth estimation method can be checked on the Middlebury reference benchmark website[8]. Our method is ranked in the first third of the the evaluated methods, obtaining a mean pixel error rate of 6%. This means that 6% of the estimated pixels have a disparity error superior or equal to 1 pixel. As a comparison, one can check that the best methods provide a mean pixel error rate of 4%, whereas the median error rate of the evaluated approaches is about 8%.

3.4 Temporal consistency

For dynamic scenes, in order to enforce the temporal consistency of the segmentation and the depth estimation between successive frames, we add an additional constraint. The idea is to force the estimation at frame $t+1$ to be related to the estimation at frame t. We use the optical flow v_i^t [PBGC10] between the captured images of camera i, to transfer the estimations computed at frame t to the location of the players at frame $t+1$. Then, we add a constraint that forces the estimations at $t+1$ to be close to these predictions. For the segmentation step at time $t+1$, it implies to first compute $\tilde{\Omega}_i^{t+1}(x + v_i^t(x)) = \Omega_i^t(x)$ and then add a new constraint $\int |S - \tilde{\Omega}_i^{t+1}|$ to the segmentation energy (1). In the depth estimation step, it leads to computing the predicted depth map $\tilde{\boldsymbol{\lambda}}^{t+1} = \{\tilde{\lambda}_i^{t+1}\}$ where $\tilde{\lambda}_i^{t+1}(x + v_i^t(x)) = \lambda_i^t(x)$ for each camera $i = 1, \cdots, M$ and considering the new energy term

$$\mathcal{E}_{pred}(\boldsymbol{\lambda}^{t+1}) = \sum_{i=1}^{M} \sum_{x \in \Omega_i^{t+1}} |\tilde{\boldsymbol{\lambda}}^{t+1} - \boldsymbol{\lambda}^{t+1}|,$$

which is added to the energy $\mathcal{E}(\boldsymbol{\lambda}^{t+1})$ introduced in expression (2).

This last temporal depth constraint is in practice enhanced through the scene flow computation from the optical flows estimated on each camera [VBR+05]. For a fixed frame t, the scene flow $F^t : \mathbb{R}^3 \to \mathbb{R}^3$ is a vector field that models the 3D displacements of the objects in a scene during one frame. The corresponding optical flow seen by a camera is simply the projection of the scene flow on the camera plane. Therefore, in contrast to the optical flow, the scene flow considers the object movement in the direction perpendicular to the camera plane, which yields a more exact temporal constraint. In this case the depth prediction $\tilde{\boldsymbol{\lambda}}^{t+1}$ can be computed as follows. Let X be the position of a 3D point at time t and let $F^t(X)$ be its estimated scene flow, so that its position at time $t+1$ is $X + F^t(X)$. If P_i denotes the projection matrix of camera i we compute

$$\tilde{\lambda}_i^{t+1}(P_i(X + F^t(X))) = \mathrm{dist}(X + F^t(X), C_i),$$

where $\mathrm{dist}(X + F^t(X), C_i)$ is the Euclidean distance between the 3D point $X + F^t(X)$ and the center of camera i.

[8] `http://vision.middlebury.edu/stereo/eval/`

4 Virtual view synthesis

Given the depth estimation of the real cameras, we now want to generate the view of 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. This step requires the trajectory of the virtual camera as an input given by the user.

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 fixed frame. This leads to a time frozen sequence equivalent to the matrix effect. The second application considers a virtual camera moving along frames, observing the dynamic scene from different points of view. Assuming that the calibration of a virtual camera C_v is known, we then want to estimate the virtual image I_v seen from this new camera. To that end, we first need to determine λ_v, the depth of this virtual image.

4.2 Virtual depth 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 [SGHS98]. 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. This process is independent of the number N of considered depth planes so that it leads to fast parallelizable synthesis algorithm.

We here focus on the virtual depth map of players, since a specific process for the background synthesis will be detailed afterwards. The virtual player depth map creation is obtained by mapping the player depth map of each real camera to the virtual camera. If several pixels (of the same or of different cameras) are transferred in the same location in the virtual image, the closest one is kept.

Hence, we aim at estimating Ω_v the virtual player mask and λ_v, its corresponding depth map. Using the previously defined notations, the depth of each pixel $\lambda_i(x)$ inside the foreground area Ω_i is projected into the virtual depth map λ_v. To compute these projections, we must first define the homographies H_λ^{iv} going from the real image I_i to the virtual image I_v with respect to the plane $\Pi_{\lambda_i(x)}$. These homographies give $x' = H^{iv}_{\lambda_i(x)}x$, the pixel of image I_v that corresponds to the pixel x of the real image I_i along the plane $\Pi_{\lambda_i(x)}$. An initial estimation $\overline{\Omega}_v$ of the virtual player mask Ω_v is computed as the intersection of the transfer of the segmentation masks Ω_i to the virtual image via the homographies $H^{iv}_{\lambda_i}$. For each pixel x' of the virtual segmentation mask $\overline{\Omega}_v$, we select, from all the depth planes "coherent" with the input data, the one $\Pi_{\lambda_v(x)}$ that is closest to the virtual camera (see Figure 6).

The final depth estimation process is summarized in algorithm 4.1.

Algorithm 4.1 *Virtual depth estimation*

```
Initialize $\lambda_v(x) = +\infty$, $\overline{\Omega}_v(x) = 0$, $\forall x \in I_v$.
For $i = 1, \cdots, M$
    For all $x \in \Omega_i$
        $x' = [H^{iv}_{\lambda_i(x)}x]$
        $\overline{\Omega}_v(x') = 1$
        if $\lambda_v(x') > \lambda_i(x)$, $\lambda_v(x') = \lambda_i(x)$
```

Note that we considered the integer approximation $\lfloor . \rfloor$ to obtain the pixel position x' in the previous algorithm. As a consequence, with this direct transfer of depth, the virtual mask $\overline{\Omega}_v$ can contain holes. To circumvent this limitation, we realize the transfer of depth in a small 3×3 neighborhood of the pixel x'. This step allows filling the holes but also dilates the segmentation mask of players $\overline{\Omega}_v$. These small errors on the players boundaries will nevertheless be handled by the color synthesis and inpainting steps.

We finally realize a last step of mask refinement in order to remove the small areas of $\overline{\Omega}_v$ that are due to depth estimation errors. To that end, we simply regularize the binary mask $\overline{\Omega}_v$ by solving:

$$\text{Minimize}_X P(X) + \alpha|X - \overline{\Omega}_v|, \tag{3}$$

where $P(X)$ denotes the perimeter of X, $\alpha > 0$, and the minimization is taken with respect to all sets of finite perimeter in the image domain (hence, all sets in the discrete case). Let χ_X be the indicator function of the set X, i.e. $\chi_X(x) = 1$ if $x \in X$ and $= 0$ otherwise. Recall that $P(X)$ coincides with the total variation of χ_X. To solve (3) we solve the relaxed problem [NEC06]

$$S_v = \arg\min_S \int |\nabla S| + \alpha \int |S - \overline{\Omega}_v|$$

and find a solution S taking values in $[0, 1]$. Any level set $\{S \geq \gamma\}$, $\gamma > 0$ then gives a solution of (3) (in practice we take $\gamma = \frac{1}{2}$). We define the final virtual mask as $\Omega_v := \overline{\Omega}_v \cap \{S \geq \frac{1}{2}\}$, since we want the final mask to be included in the transferred mask $\overline{\Omega}_v$ to remove the small artifacts. This step allows obtaining a good compromise between the transferred depth areas and their size. An example of such transfer is presented in Figure 9.

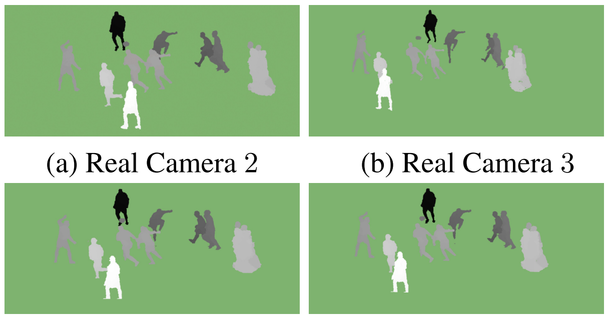

(a) Real Camera 2 (b) Real Camera 3
(c) Virtual Camera 1 (d) Virtual Camera 2

Figure 9: 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).

Remark Note that better approximations of the virtual depth map can be obtained with a backward transfer of depths. This can be done by testing all the possible depth planes for all the pixels of the virtual image. Such an approach could be formalized by an energy function (and eventually be completed by a depth spatial regularization term). However, the computational cost of this backward transfer depends on the number of considered depth planes N and is therefore prohibitive with respect to the small gain of quality in the results.

4.3 Player synthesis

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

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 [LH06]).

In practice, once the depth map λ_v has been estimated, we can recover the color of the virtual image I_v, using homographies. Thus for all $x \in \Omega_v$, we look in all the other images for the values of $I_i(H^{vi}_{\lambda_v(x)}x)$, $i = 1, \cdots, M$, to construct the image value $I_v(x)$. We also use this step to refine the virtual player mask Ω_v. Indeed, if the pixel $x \in \Omega_v$ is not seen from the other cameras (this may happen since we used a direct transfer to estimate Ω_v), we will have $\Omega_i(H^{vi}_{\lambda_v(x)}x) = 0$, $i = 1, \cdots, M$ so that the pixel x is part of the background. This synthesis process is summed up in the algorithm 4.2.

Algorithm 4.2 *Player synthesis*

```
Initialize $p(x) = 0$, $I_v(x) = 0$, $\forall x \in \Omega_v$.
For all $x$ such that $\Omega_v(x) = 1$
    For $i = 1, \cdots, M$
        $x_i = H^{vi}_{\lambda_v(x)}x$
        if $\Omega_i(x_i) = 1$
            $c = exp(-|\lambda_i(x_i) - \lambda_v(x)|)$
            $I_v(x) = I_v(x) + cI_i(x_i)$ ,
            $p(x) = p(x) + c$.
    if $p(x) > 0$, $I_v(x) = I_v(x)/p(x)$
    else $\Omega_v(x) = 0$
```

The weight $p(x)$ is used to represent the influence of the different image data. Note also that the values $\lambda_i(H^{vi}_{\lambda_v(x)}x)$ are interpolated.

4.4 Playground and stadium synthesis

The two previous algorithms are used to generate the virtual players. However, the playground and the stadium are not treated with such an approach. Hence, the pixels outside the virtual players are processed using three particular depth planes, as illustrated in Figure 10.

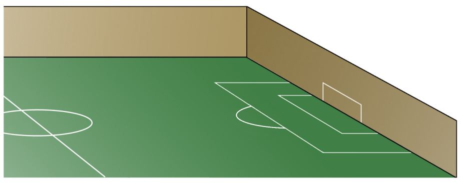


Figure 10: Background model. Three specific planes are considered for background synthesis: the groundfield, the goal and the stands.

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 is the cause of some blurring effects that appear in the non flat parts of the groundfield of the virtual view. Two additional planes model the goal and the stands. These form a very rough approximation of the real geometry. The real images back-projected into these planes clearly do not match. To palliate the visual artifacts of this approximation, we considered the following processes.

For the stand and goal synthesis, we use view dependent texture mapping [DYB98] so that only the camera that is closer to the virtual camera is used to produce the texture of the goal and stand planes. When moving from one real camera to another, a short transition zone is used, and the images produced by both cameras are blended to avoid a sudden change in the texture. Using this process, we are able to create a virtual background image, for a given calibration corresponding to a virtual camera. This virtual background image can also be decomposed into three areas corresponding to the stands, the goal and the groundfield.

The groundfield colorization is realized in a different way. As we want the moving shadows of the players to be synthesized, we can not use a reference texture of the field. From the three planes stadium model, we know the binary mask F_v corresponding to the field in the virtual image I_v. The area to be synthesized is then defined by $D_v(x) = 1$ if $F_v(x) = 1$ and

$\Omega_v(x) = 0$, i.e. if it is ground field and if there are no players. The same areas D_i can be defined for each real camera i. The backward color transfer is then used between the virtual image and the real images with respect to the field plane homographies H_F^{vi} corresponding to the field plane Π_F. If the ground field at pixel $x \in D_v$ is occluded in all the real cameras (i.e. $D_i(H_F^{vi}x) = 0$, $i = 1, \cdots, M$), then the pixel x is added to the mask M_v of pixels to be inpainted in the filling-in post-processing step. The virtual ground field synthesis is shown in Algorithm 4.3.

```
**Algorithm 4.3** *Playground synthesis*
Initialize $p(x) = I_v(x) = M_v(x) = 0, \forall x \in D_v$.
For all $x$ such that $D_v(x) = 1$
    For $i = 1, \cdots, M$
        $x_i = H_F^{vi} x$
        if $D_i(x_i) = 1$
            $I_v(x) = I_v(x) + I_i(x_i)$,
            $p(x) = p(x) + 1$.
    if $p(x) > 0$, $I_v(x) = I_v(x)/p(x)$
    else $M_v(x) = 1$
```

Remark: Finally note that if, for a pixel x of image I_v, the virtual plane obtained by Algorithm 4.1 is behind the three planes stadium model, we then remove the pixel x from the virtual players segmentation mask by setting $\Omega_v(x) = 0$.

An example of synthesis including the goal and the stands is shown in Figure 11. We also present in Figure 12 some synthesized virtual images.

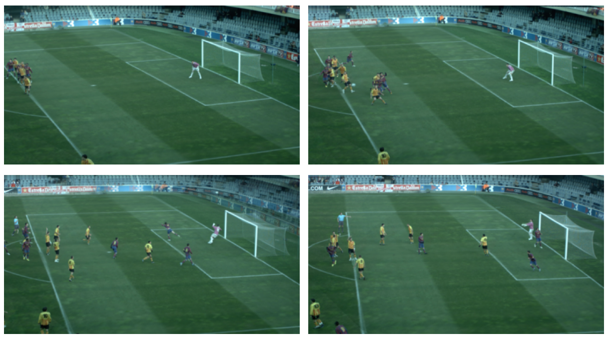


Figure 11: Virtual views from a moving camera observing the dynamic scene. The goal and the stands have been synthesized using the specific planes illustrated by the Figure 10.

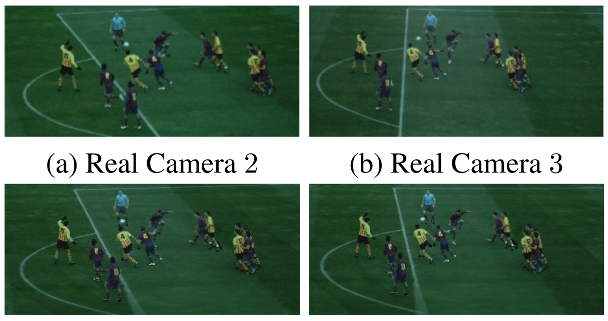

(a) Real Camera 2 (b) Real Camera 3
(c) Virtual Camera 1 (d) Virtual Camera 2

Figure 12: 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 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).

We first recall that M_v, the mask of pixels to inpaint, has been initialized with the occluded part of the groundfield in Algorithm 4.3. The mask is then completed by adding the pixels located at the discontinuities of the virtual depth map λ_v. Without loss of generality, we consider that the depth of the pixels belonging to the groundfield, stands and goal areas are respectively represented by the planes $\lambda_v = N$, $N+1$

and $N + 2$. Hence, for each pixel x of the virtual image, if $\exists y \in \mathcal{N}(x)$, such that $\lambda_v(y) \neq \lambda_v(x)$ and $x \in \Omega_v$ or $y \in \Omega_v$, then $M_v(x) = 1$. The neighborhood $\mathcal{N}(x)$ here represents a 3×3 patch around the pixel x. The mask contains pixels inside the players but also the pixels at players boundaries with the groundfield, stands and goal areas.

Examples of such areas are shown in Figure 13.

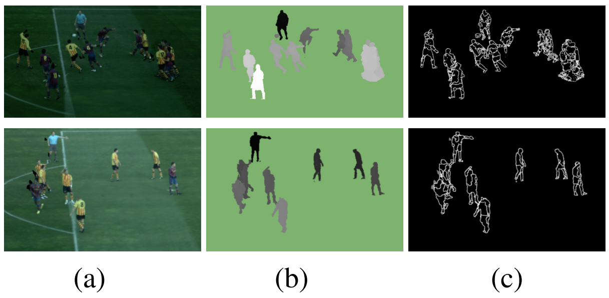


Figure 13: Creation of the inpainting masks. (a) Synthetic images. (b) Virtual depth maps. (c) Areas to inpaint.

The correction is performed with an image inpainting algorithm [BBCS10] based on image color patches comparison and copy, although a more simple method [CPT04] is sufficient in the present case. Let us detail the basic ideas behind exemplar-based inpainting methods [EL99, CPT04].

A database of N_p candidate color patches P_k, $k = 1, \cdots, N_p$, is first created using the color information available in the real images $I_i(t)$, $i = 1, \cdots, M$, around the segmentation masks Ω_i^t at all the time t of the sequence. Then, for each pixel of the current inpainting mask $x \in M_v(t)$, the process searches into the database the patch that is the most similar to the color patch of the virtual image $I_v(t)$ centered at x. This search is realized through the computation of a similarity measure. Let h be the parameter defining the patch size, we then consider patches of size $(2h+1) \times (2h+1)$. The similarity measure between the current patch centered on pixel $x = (i, j)$ and a candidate patch P_k is then defined as:

$$s(x,k) = \sum_{m=-h}^{h} \sum_{n=-h}^{h} \| I_v(i+m, j+n) - P_k(h+m, h+n) \|.$$

For the sake of clarity, we omit the sum on the color channels on the last expression. The measure $\|.\|$ contains three terms, one of texture synthesis, one for diffusion and one for the spatial consistency (see [BBCS10] for details). We select the patch P_{k^*} that minimizes this measure by taking $k^* = \arg\min_k s(x,k)$. The color of all the pixels y of the patch centered on x that belong also into the inpainting mask M_v is then replaced by the corresponding color of the selected patch P_{k^*} and, once processed, these pixels y are removed from the inpainting mask.

In practice, as we considered $h = 4$, we have to deal with color patches of size $3 \times 9^2 = 243$. The dimension of the data base is also very large, since the number of candidate patches N_p becomes superior to 10^5 by considering only 10 frames. A dimension reduction is realized in the patch space through Principal Component Analysis (PCA) and only the most significant 10% of information is kept for the further searches. Our search method also relies on search trees (*kd-trees*), in order to speed up the search of similar patches. Then, each patch P centered on x is projected on the PCA basis and its nearest neighbor patch is found on the tree.

In Figure 14, we present examples of inpainting correction.

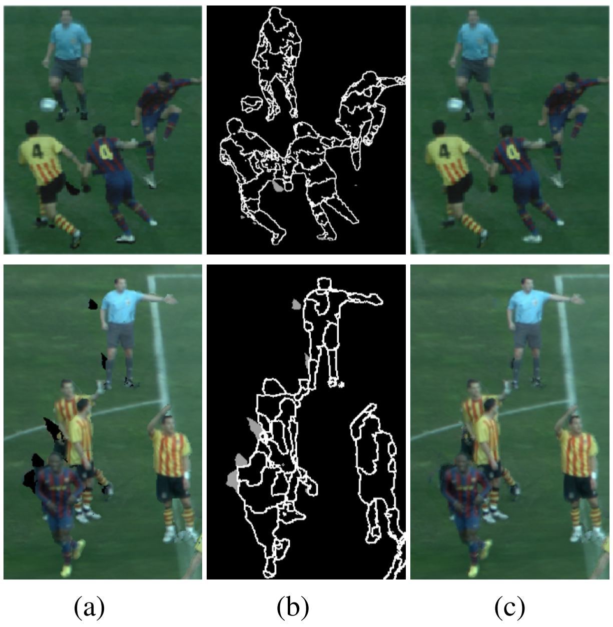


Figure 14: Image inpainting. (a) The synthetic images. (b) The corresponding areas to inpaint composed by the union of the missing part of the field (in gray) and the virtual depth discontinuities areas (in white). (c) Final inpainted images.

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 [PBGC10] on the inpainted sequence. Hence, the value of the color of a pixel is filtered by considering a temporal window along its trajectory.

More precisely, let us consider the virtual image sequence $I_v(x,t)$, $x \in D$ (where D denotes the virtual image domain) and $t_0 \leq t \leq t_f$. We first compute the forward v_+^t and backward v_-^t optical flows on this virtual sequence. In order to filter $I_v(x,t)$, the color of the pixel x of the virtual image at time t, we select all the colors of the pixels on the trajectory of x on the temporal window $[t-T, t+T]$. Denoting as $x_k = x$, this trajectory is obtained as $x_{k+1} = x_k + v_+^k$ for $t \leq k \leq t+T$ and $x_{k-1} = x_k + v_-^k$ for $t \geq k \geq t-T$. The interpolated colors of the virtual images at points $I_v(x_k, k)$, $t-T \leq k \leq t+T]$ are then used to estimate the final color of $I_v(x,t)$. One direct way to filter this set of colors is to take their mean or median value. Such an approach is nevertheless very sensitive to the quality of the optical flow estimations. On the boundary of players with grass, we do not want to merge player and grass colors and have to deal with this problem carefully. Namely, we have observed that taking the mean value of colors leads to significant blurring effects when the players move quickly, as the amplitude of the optical flow vectors and the corresponding errors are larger. As a consequence, we choose a weighted mean of the colors $I_v(x_k,k)$ given by:

$$I_v(x,t) = \frac{\sum_{k=t-T}^{t+T} w(k) I_v(x_k,k)}{\sum_{k=t-T}^{t+T} w(k)},$$

the weight $w(k)$ being given by the corresponding optical flow amplitude at time k:

$$w(k) = \begin{cases} exp(-||v_-^{k+1}||^2/\sigma^2) & \text{if } k < t \\ 1 & \text{if } k = t, \\ exp(-||v_+^{k-1}||^2/\sigma^2) & \text{if } k > t \end{cases}$$

where σ weights the optical flow amplitude. The color values are then weighted with respect to the inverse of the norm of the optical flows. 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 temporal consistency.

A sketch of the temporal filtering process is presented in Figure 15. As illustrated by Figure 16, this process allows to smooth the inpainted images in order to attenuate the potential temporal artifacts.

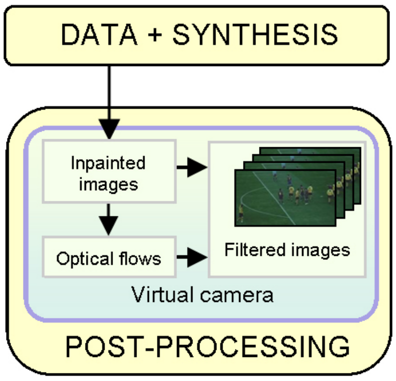
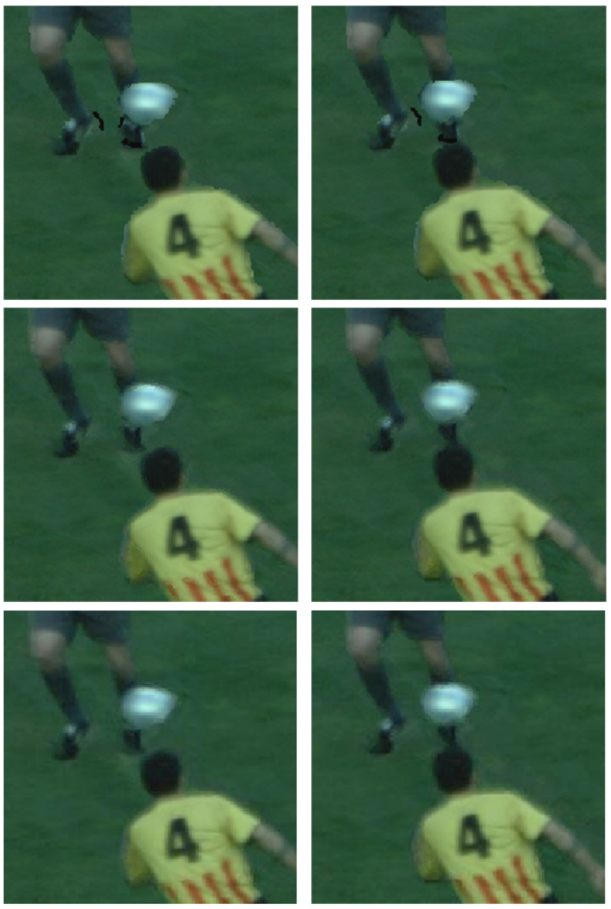


Figure 15: Temporal filtering. The inpainted image are filtered temporally using optical flow trajectories to impose consistency and provide virtual sequences without temporal artifacts.



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

Figure 16: 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 row. The inpainting results are shown in the second row. The third row illustrates the final filtered images.

6 Experimentation details

In this section, we give some details on the experimentations we realized. We provide two videos [9] corresponding to two different sequences. The first one presents a time frozen virtual camera warping two real cameras. The second video shows a dynamic virtual camera observing the moving game. The videos have been obtained automatically without user interaction or parameter tuning, some illustration are presented in Figures 17 and 18.

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 auxiliary 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 in more detail. Concerning the depth estimation, as we impose visibility constraints between the different cameras in a graph representation, the graph includes some complex connections. 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 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 [PBGC10]. Such a process is potentially real time, but our current implementation takes up to 30 seconds in 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 time 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.

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

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 cameras (we used 4 in our experiments). We have presented experiments showing the quality of the generated sequences. There are however limitations in the current prototype that need to be addressed in the future.

In terms of robustness, the systems relies heavily in the correct extraction of the players silhouette, which might be difficult to obtain automatically depending on the lighting conditions. Also, the current implementation generates views that are in between the real cameras; the quality of the synthesis degrades as the virtual camera moves away from the real ones. Therefore, up to 10 cameras would be required to cover the whole field from one side. Finally, the system takes to much time to generate novel views for being used in live events.

In a future work, the different steps of the process will be optimized, mainly in terms of computational cost. In particular, we will develop a GPU implementation of depth and optical flow computation following the algorithm proposed in [BCGP10]. Let us also mention another important point which concerns the field, which is assumed to be flat. A better approximation taking into account the field curvature should be considered in order to improve the calibration and the synthesis steps. Note also that the present work only used 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.

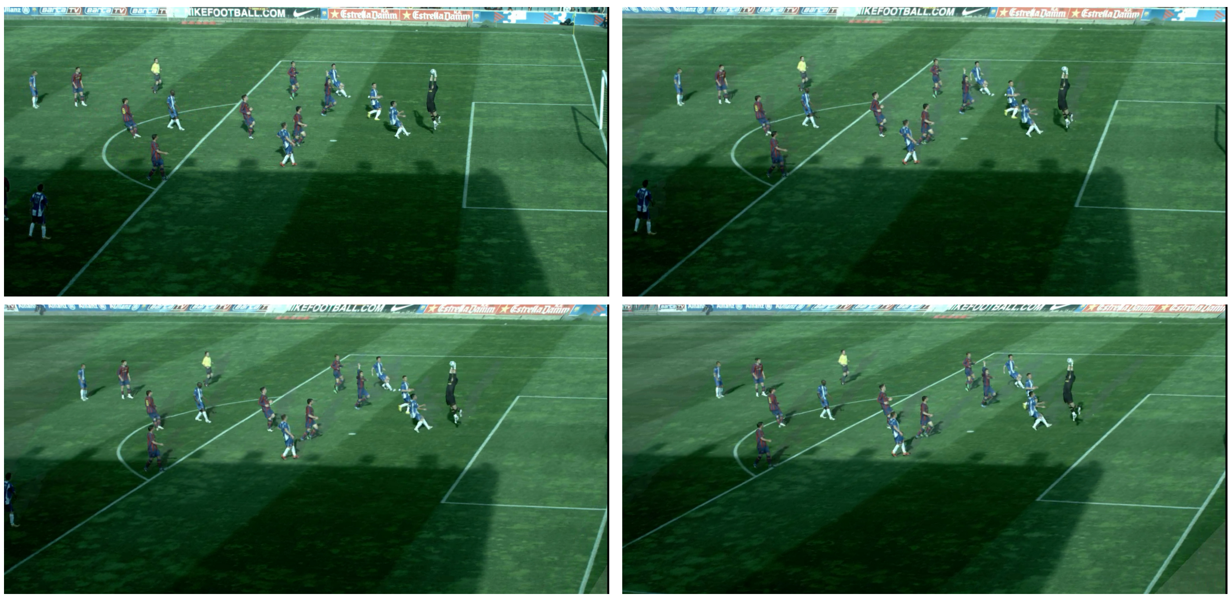


Figure 17: Static synthesis. The four images have been synthesized from a time frozen virtual camera warping two real cameras.

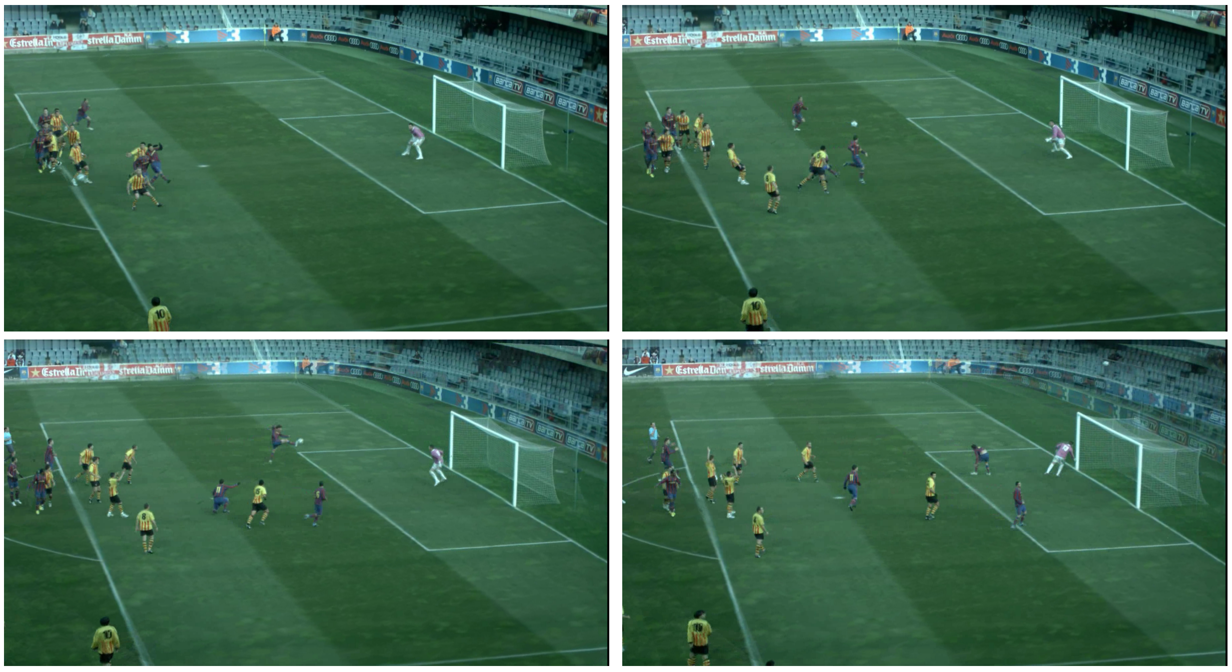


Figure 18: Dynamic synthesis. The four images have been synthesized from a dynamic virtual camera observing the moving game.

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. We also acknowledge partial support by the FP7-ICT-2009-4 project ”Free-viewpoint Immersive Networked Experience” (FINE) financed by EC. Pau Gargallo 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

[AC09] L. Álvarez and V. Caselles, *Procedimiento para la calibración de una cámara de video y TV*, Spanish Patent No. EP-0140. Patent Pending. (2009).

[Bau74] B. G. Baumgart, *Geometric Modeling for Computer Vision*, Tech. report, Department of Computer Science, Stanford University, California, 1974.

[BBCS10] A. Bugeau, M. Bertalmío, V. Caselles, and G. Sapiro, *A Comprehensive Framework for Image Inpainting*, IEEE Trans. on Image Processing **19** (2010), no. 10, 2634–2645, ISSN 1057-7149.

[BBPP10] L. Ballan, G.J. Brostow, J. Puwein, and M. Pollefeys, *Unstructured video-based rendering: interactive exploration of casually captured videos*, ACM SIGGRAPH 2010 papers (New York, NY, USA), SIGGRAPH '10, ACM, 2010, pp. 87:1–87:11, ISBN 978-1-4503-0210-4.

[BCGP10] A. Baeza, V. Caselles, P. Gargallo, and N. Papadakis, *A narrow band method for the convex formulation of discrete multi-label problems*, SIAM Journal on Multiscale Modeling & Simulation **8** (2010), 2048–2078, ISSN 1540-3459.

[BJ01] Y. Boykov and M. P. Jolly, *Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images*, Proc. IEEE International Conference on Computer Vision (ICCV'01), vol. 1, 2001, pp. 105–112, ISBN 0-7695-1143-0c.

[BVZ98] Y. Boykov, O. Veksler, and R. Zabih, *Markov random fields with efficient approximations*, Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'98), 1998, pp. 648–655, ISBN 0-8186-8497-6.

[BVZ01] Y. Boykov, O. Veksler, and R. Zabih, *Fast approximate energy minimization via graph cuts*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **23** (2001), no. 11, 1222–1239, ISSN 0162-8828.

[Col96] R.T. Collins, *A Space-Sweep Approach to True Multi-Image Matching*, Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '96), 1996, pp. 358–363.

[CPT04] A. Criminisi, P. Pérez, and K. Toyama, *Region filling and object removal by exemplar-based image inpainting*, IEEE Transactions on Image Processing **13** (2004), no. 9, 1200–1212, ISSN 1057-7149.

[DYB98] P. Debevec, Y. Yu, and G. Boshokov, *Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping*, 1998.

[EHD00] A.M. Elgammal, D. Harwood, and L.S. Davis, *Non-parametric Model for Background Subtraction*, Proc. European Conference on Computer Vision (ECCV'00), vol. 2, 2000, pp. 751–767.

[EL99] A.A. Efros and T.K. Leung, *Texture synthesis by non-parametric sampling*, Proc. IEEE International Conference on Computer Vision (ICCV'98), 1999, pp. 1033–1038, ISBN 0-7695-0164-8.

[FB05] J.S. Franco and E. Boyer, *Fusion of multiview silhouette cues using a space occupancy grid*, Proc. IEEE International Conference on Computer Vision (ICCV'05), vol. 2, 2005, pp. 1747–1753, ISBN 0-7695-2334-X.

[FP10] Y. Furukawa and J. Ponce, *Accurate, dense, and robust multiview stereopsis*, IEEE transactions on pattern analysis and machine intelligence **38** (2010), no. 8, 1362–1376, ISSN 0162-8828.

[GCS06] M. Goesele, B. Curless, and S.M. Seitz, *Multi-view stereo revisited*, Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 2, 2006, pp. 2402–2409, ISBN 0-7695-2597-0.

[GH11] J.Y. Guillemaut and A Hilton, *Joint multi-layer segmentation and reconstruction for free-viewpoint video applications*, International Journal of Computer Vision **93** (2011), no. 1, 73–100, ISSN 1573-1405.

[GHK+06] 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*, Proc. International Broadcasting Convention, 2006.

[GHS+07] J.Y. Guillemaut, A. Hilton, J. Starck, J. Kilner, and O. Grau, *A bayesian framework for simultaneous matting and 3d reconstruction*, Proc. International Conference on 3-D Digital Imaging and Modeling (3DIM'07), 2007, pp. 167–176.

[GKH09] J.Y. Guillemaut, J. Kilner, and A. Hilton, *Robust graph-cut scene segmentation and reconstruction for free-viewpoint video of complex dynamic scenes*, Proc. IEEE International Conference on Computer Vision (ICCV'09), 2009, pp. 809–816, ISBN 978-1-4244-4420-5.

[GTH+07] O. Grau, GA Thomas, A. Hilton, J. Kilner, and J. Starck, *A Robust Free-Viewpoint Video System for Sport Scenes*, Proc. 3DTV Conference, 2007, 2007, pp. 1–4, ISBN 978-1-4244-0721-7.

[HK07] M. Habbecke and L. Kobbelt, *A surface-growing approach to multi-view stereo reconstruction*, Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07), 2007, pp. 1–8, ISBN 1-4244-1179-3.

[HP05] K. Hirakawa and T.W. Parks, *Adaptive homogeneity-directed demosaicing algorithm*, IEEE Trans. on Image Processing **14** (2005), no. 3, 360–369, ISSN 1057-7149.

[HP10] G. Haro and M. Pardàs, *Shape from incomplete silhouettes based on the reprojection error*, Image and Vision Computing **28** (2010), no. 9, 1354–1368, ISSN 0262-8856.

[HS88] C. Harris and M. Stephens, *A Combined Corner and Edge Detection*, Proc. of The Fourth Alvey Vis. Conf., 1988, pp. 147–151.

[HZ03] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision 2*, Cambridge University Press New York, NY, USA, 2003, ISBN 0521540518.

[IS02] N. Inamoto and H. Saito, *Fly through view video generation of soccer scene*, Proc. Entertainment computing: technologies and applications, Springer, 2002, p. 109, ISBN 9781402073601.

[IS03] N. Inamoto and H. Saito, *Immersive observation of virtualized soccer match at real stadium model*, Proc. International Symposium on Mixed and Augmented Reality (ISMAR03), Citeseer, 2003, pp. 188–197, ISBN 0-7695-2006-5.

[JNS10] S. Jarusirisawad, V. Nozick, and H. Saito, *Real-time video-based rendering from uncalibrated cameras using plane-sweep algorithm*, Journal of Visual Communication and Image Representation **21** (2010), 577–585, ISSN 1047-3203.

[KCB+06] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, *Probabilistic fusion of stereo with color and contrast for bilayer segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006), no. 9, 1480–1492, ISSN 0162-8828.

[KKO03] T. Koyama, I. Kitahara, and Y. Ohta, *Live mixed-reality 3d video in soccer stadium*, Proc. International Symposium on Mixed and Augmented Reality, 2003, pp. 178–186, ISBN 0-7695-2006-5.

[KNR95] T. Kanade, P.J. Narayanan, and P.W. Rander, *Virtualised reality: concepts and early results*, Proc. of IEEE Workshop on Representation of Visual Scenes, 1995, pp. 69–76, ISBN 0-8186-7122-X.

[KOS+01] I. Kitahara, Y. Ohta, H. Saito, S. Akimichi, T. Ono, and T. Kanade, *Recording multiple videos in a large-scale Space for large-scale virtualized reality*, Proc. of International Display Workshops (AD/IDWf̂01), 2001, pp. 1377–1380.

[KS00] K.N. Kutulakos and S.M. Seitz, *A theory of shape by space carving*, International Journal of Computer Vision **38** (2000), no. 3, 199–218, ISSN 0920-5691.

[KZ01] V. Kolmogorov and R. Zabih, *Computing Visual Correspondence with Occlusions via Graph Cuts*, Proc. IEEE International Conference on Computer Vision (ICCV'01), vol. 2, 2001, pp. 508–515, ISBN 0-7695-1143-0.

[KZ02] V. Kolmogorov and R. Zabih, *Multicamera Scene Reconstruction via Graph Cuts*, Proc. European Conference on Computer Vision (ECCV'02), 2002, pp. 82–96, ISBN 3-540-43746-0.

[KZ04] V. Kolmogorov and R. Zabih, *What Energy Functions Can Be Minimized via Graph Cuts?*, IEEE Transactions on Pattern Analysis and Machine Intelligence **26** (2004), no. 2, 147–159, ISSN 0162-8828.

[Lau94] A. Laurentini, *The visual hull concept for silhouette-based image understanding*, IEEE Transactions on Pattern Analysis and Machine Intelligence **16** (1994), no. 2, 150–162, ISSN 0162-8828.

[LH06] H. Li and R. Hartley, *Inverse tensor transfer with applications to novel view synthesis and multi-baseline stereo*, Signal Processing: Image Communication **21** (2006), no. 9, 724 – 738, ISSN 0923-5965.

[LL05] Q. Li and Y. Luo, *Automatic camera calibration for images of soccer match*, World Academy of Science, Engineering and Technology **1** (2005), 170–173.

[LLB$^+$10] C. Lipski, C. Linz, K. Berger, A. Sellent, and M. Magnor, *Virtual Video Camera: Image-Based Viewpoint Navigation Through Space and Time*, Computer Graphics Forum **29** (2010), no. 8, 2555–2568, ISSN 1467-8659.

[Low04] D.G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision **60** (2004), 91–110, ISSN 0920-5691.

[LPC08] J.L. Landabaso, M. Pardás, and J.R. Casas, *Shape from inconsistent silhouette*, Computer Vision and Image Understanding **112** (2008), no. 2, 210–224, ISSN 1077-3142.

[NEC06] M. Nikolova, S. Esedoglu, and T.F. Chan, *Algorithms for finding global minimizers of image segmentation and denoising models*, SIAM Journal on Applied Mathematics **66** (2006), no. 5, 1632–1648, ISSN 0036-1399.

[OKK$^+$07] Y. Ohta, I. Kitahara, Y. Kameda, H. Ishikawa, and T. Koyama, *Live 3D video in soccer stadium*, International Journal of Computer Vision **75** (2007), no. 1, 173–187, ISSN 0920-5691.

[PBGC10] N. Papadakis, A. Baeza, P. Gargallo, and V. Caselles, *Polyconvexification of the multi-label optical flow problem*, Proc. IEEE International Conference on Image Processing (ICIP'10), 2010, pp. 765–768.

[PC10] N. Papadakis and V. Caselles, *Multi-label depth estimation for graph cuts stereo problems*, Journal of Mathematical Imaging and Vision **38** (2010), no. 1, 70–82, ISSN 0924-9907.

[SCD$^+$06] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, *A comparison and evaluation of multi-view stereo reconstruction algorithms*, Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 1, 2006, pp. 519–528, ISBN 0-7695-2597-0.

[SCG01] F. Szenberg, P. Carvalho, and M. Gattass, *Automatic camera calibration for image sequences of a football match*, Proc. Advances in Pattern Recognition (ICAPR'01), vol. 2013, 2001, pp. 303–312.

[SD99] S.M. Seitz and C.R. Dyer, *Photorealistic scene reconstruction by voxel coloring*, International Journal of Computer Vision **35** (1999), no. 2, 151–173.

[SG99] C. Stauffer and W.E.L. Grimson, *Adaptive background mixture models for real-time tracking*, Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '99), vol. 2, 1999, pp. 252–258.

[SGHS98] J. Shade, S. Gortler, L.-W. He, and R. Szeliski, *Layered depth images*, Proc. ACM Transactions on Graphics (SIGGRAPH'98), 1998, pp. 231–242, ISBN 0-89791-999-8.

[SH07] J. Starck and A. Hilton, *Surface capture for performance-based animation*, IEEE Computer Graphics and Applications **27** (2007), no. 3, 21–31, ISSN 0272-1716.

[SVZ00] D. Snow, P. Viola, and R. Zabih, *Exact voxel occupancy with graph cuts*, Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00), vol. 1, 2000, pp. 345–352, ISBN 0-7695-0662-3.

[SZS03] J. Sun, N.N. Zheng, and H.Y. Shum, *Stereo matching using belief propagation*, IEEE Transactions on Pattern Analysis and Machine Intelligence **25** (2003), no. 7, 787–800, ISSN 0162-8828.

[TF03] M.F. Tappen and W.T. Freeman, *Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters*, Proc. IEEE International Conference on Computer Vision (ICCV'03), vol. 2, 2003, pp. 900–906, ISBN 0-7695-1950-4.

[TMHF00] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, *Bundle Adjustment – A Modern Synthesis*, Vision Algorithms: Theory and Practice, Lecture Notes in Computer Science, vol. 1883, 2000, pp. 298–372, ISBN 978-3-540-67973-8.

[VBR+05] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, *Three-Dimensional Scene Flow*, IEEE Trans. Pattern Anal. Mach. Intell. **27** (2005), no. 3, 475–480, ISSN 0162-8828.

[YZ07] X. Ying and H. Zha, *Camera calibration from a circle and a coplanar point at infinity with applications to sports scenes analyses*, Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07), 2007, pp. 220–225, ISBN 978-1-4244-0912-9.

[ZKU+04] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, *High-quality video view interpolation using a layered representation*, Proc. ACM Transactions on Graphics (SIGGRAPH'04), vol. 23, issue 3, 2004, pp. 600–608.

[ZPB07] C. Zach, T. Pock, and H. Bischof, *A globally optimal algorithm for robust TV-L1 range image integration*, Proc. IEEE International Conference on Computer Vision (ICCV'07), 2007, pp. 1–8, ISBN 978-1-4244-1631-8.

Citation

N. Papadakis, A. Baeza, I. Rius, X. Armangué, A. Bugeau, O. D'Hondt, P. Gargallo, V. Caselles and S. Sagàs, *Virtual camera synthesis for soccer game replays* , Journal of Virtual Reality and Broadcasting, 9(2012), no. 5, November 2012, urn:nbn:de:0009-6-35132, ISSN 1860-2037.