Diss. ETH No. 20290

Video-Based Rendering Techniques

A dissertation submitted to
ETH Zurich

for the Degree of
Doctor of Sciences

presented by
Marcel Germann
Dipl. Informatik-Ing., ETH Zurich, Switzerland
born 28. March 1980
citizen of Switzerland

accepted on the recommendation of
Prof. Dr. Markus Gross, examiner
Prof. Dr. Marc Pollefeys, co-examiner

2012

In memory of my mother

Abstract

The goal of video based rendering is to render images or videos of a scene from novel viewpoints, based on video footage from one or more real cameras. This has a high potential especially for outdoor sports events, where usually several cameras are available, but adding other technology into the scene is expensive or not allowed. However, in such uncontrolled setups, the input suffers from several drawbacks. The cameras are usually sparsely placed, causing wide baselines between them. Positioned far from the scene, the cameras are also difficult to calibrate, i.e., to compute their positions, viewing directions and internal parameters based on the images. Therefore, the set of usable cameras is reduced to those with wide-angle shots, causing the coverage of subjects in the scene to be at low resolution. In this thesis we present two different approaches to render novel views in such difficult outdoor setups.

The first approach is based on a body pose estimation used to construct *articulated billboards*, a novel representation of the human body. First, a coarse pose guess is established according to comparisons of silhouettes with a database. From the k best 2D pose estimations of the individual cameras, the optimal combination is chosen according to errors in the 3D triangulation, which results in a 3D pose estimation. After a consistency test to remove left/right flips of arms or legs between frames, the body poses are optimized in a spatio-temporal energy minimization. This includes terms for smoothness, silhouette fitting as well as data-driven terms to favor plausible poses. The articulated billboards are placed according to the results of this pose estimation and consist of a billboard fan per body part. In a novel view-dependent blending and rendering technique they can be shown from arbitrary viewpoints.

The second approach introduces an adaptive reconstruction method and a view-dependent geometry morph. A separate 2.5D representation is obtained in every camera image by a coarse-to-fine reconstruction. It uses sparse feature match correspondences as well as back-projection errors to find optimal depth values for the vertices of a 2.5D triangulation and to adaptively subdivide triangles only where it is required. A refinement step according to back-projections and neighbor look-ups improves the found vertex depths. It results in a 2.5D reconstruction per camera, which are merged into a final 3D representation. Novel viewpoints

are rendered not only with a view-dependent blending but also with a view-dependent geometry. For this, a morph of the geometry is achieved by a force field computed from non-epipolar feature matches. The reconstruction is robust to several errors occurring particularly in outdoor setups and the view-dependent rendering corrects for calibration errors, for which no 3D reconstruction would fit to all camera images.

For both approaches we present results based on conventional TV camera footage of several soccer scenes. The quality of the images and videos is comparable to those of the input footage and the results show the potential of both approaches. We conclude this thesis with a comparison of the two approaches as well as a collection of ideas for future work.

Zusammenfassung

Das Ziel von Video-basiertem Rendering ist es von einer Szenerie, die mit einer oder mehreren Kameras aufgenommen wurde, Bilder oder Videos von einem neuen Blickwinkel zu zeigen, wo keine Kamera plaziert war. Dies hat ein hohes Potential, speziell für Aussenaufnahmen von Sportereignissen, die normalerweise von mehreren Kameras aufgenommen werden, wo es aber teuer oder nicht erlaubt ist, weitere Technologie der Szene hinzuzufügen. Die Eingangsdaten solcher Aussenaufnahmen weisen jedoch verschiedenen Mängel auf. Normalerweise sind nur wenige Kameras vorhanden, was bedeutet, dass die Distanzen zwischen ihnen gross sind. Da diese auch weit vom Geschehen entfernt sind, sind sie ausserdem schwierig zu kalibrieren, bzw. es ist schwierig anhand der Bilder zu bestimmen, wo sie positioniert sind, wohin sie schauen und welche internen Parameter sie haben. Dies reduziert die Menge der brauchbaren Kameras auf solche mit Weitwinkel-Ansichten, was wiederum bedeutet, dass die Personen in der Szene mit niedriger Auflösung abgebildet sind. In dieser Arbeit präsentieren wir zwei verschiedene Ansätze, um in solch schwierigen Aussenaufnahmen neue Ansichten zu rendern.

Der erste Ansatz basiert auf einer Körperposenschätzung die für die Konstruktion von *Articulated Billboards*, einer neuartigen Repräsentation des menschlichen Körpers, verwendet wird. Zuerst wird anhand von Silhouettenvergleichen mit einer Datenbank eine grobe Schätzung der Pose berechnet. Von den k besten 2D Posenschätzungen der einzelnen Kameras wird anhand von 3D Triangulierungsfehlern die optimale Kombination ausgewählt, woraus eine 3D Posenschätzung entsteht. Nach einem Konsistenztest zur Entfernung von Rechts/Links-Verwechslungen der Arme und Beine wird die Pose in einer spatio-temporalen Energieminimierung optimiert. Diese beinhaltet Terme für Glätte, Terme für das Passen auf Silhouetten, sowie datengetriebene Terme um plausible Posen zu bevorzugen. Die Articulated Billboards werden dann anhand der Resultate der Posenschätzung plaziert und bestehen aus einem Billboard-Fächer pro Körperteil. In einer neuen blickwinkelabhängigen Rendertechnik können diese von beliebigen Ansichten gerendert werden.

Der zweite Ansatz führt eine adaptive Rekonstruktionsmethode und ein blickwinkelabhängigen Geometriemorph ein. In jeder Kamera wird eine separate 2.5D Repräsentation mittels eines grob-zu-fein Verfahrens errechnet. Es verwendet dünngesiedelte Merkmalspaarungen sowie Rückprojektionsfehler um optimale

Tiefenwerte für die Ecken einer 2.5D Triangulierung zu finden und um adaptiv nur solche Dreiecke zu unterteilen wo es notwendig ist. Ein Verfeinerungsschritt anhand von Rückprojektionsfehlern und Nachbarschaftsabfragen verbessert die gefundenen Ecktiefen. Dies resultiert in einer 2.5D-Rekonstruktion pro Kamera, welche dann in eine finale 3D Repräsentation vereint werden. Neue Ansichten werden nicht nur mit einem blickwinkelabhängigen Blenden sondern auch mit einer blickwinkelabhängigen Geometrie gerendert. Hierzu wird ein Morph der Geometrie mittels eines Kraftfeldes erreicht, das durch nicht-epipolare Merkmalspaarungen berechnet wird. Die Rekonstruktion ist robust gegen verschiedene Fehler die speziell bei Aussenaufnahmen auftreten und das blickwinkelabhängige Rendering korrigiert Kalibrationsfehler, bei denen keine 3D Rekonstruktion in alle Kamerabilder passen würde.

Für beide Ansätze präsentieren wir Resultate die auf Aufnahmen von normalen TV-Kameras von verschiedenen Fussballspielen basieren. Die Qualität der Bilder und Videos ist vergleichbar mit denjenigen der Eingangsdaten und die Resultate zeigen das Potential beider Ansätze. Zum Schluss dieser Arbeit vergleichen wir die zwei verschiedenen Ansätze und erläutern Ideen füer zukünftige Arbeiten.

Acknowledgments

First of all, I would like to sincerely thank my advisor Prof. Markus Gross. His interest in and energy for open research problems was very encouraging and motivating for the work in this thesis. His open mind and constructive support for new ideas is extremely fruitful for a creative development in research. I would also like to thank Prof. Marc Pollefeys for his valuable comments, ideas and corrections to this thesis.

Many thanks goes to my collaborators (in alphabetical order) Dr. Alexander Hornung, Dr. Richard Keiser, Dr. Tiberiu Popa, Dr. Stephan Würmlin and Dr. Remo Ziegler. It was a pleasure working together with them and I enjoyed the collaboration with people from LiberoVision and people from ETH. Their support, advice, guidance and motivation was a fundamental part of my thesis.

It was also a pleasure to work together with Remo Frey and Marcel Müller. I would like to thank them for their hard work on the projects we did together.

Furthermore, I would like to thank the former and current members of CGL, DRZ, AGG, CVG and IGL. Throughout the work on this thesis, I had many many fruitful discussions with them and could benefit from being surrounded by experts in various fields. Unfortunately, a list of names would be too long to place here. But not only that they contributed to the work, they are also responsible for the wonderful work environment I could enjoy. Many of them became and will remain good friends. Thank you guys!

Also I would like to thank LiberoVision as a company and give my special thanks to all the people working at LiberoVision. Their help and support as well as the time I could spend there made me feel to be part of a great team.

Very important in supporting and motivating me in my work and throughout my life are my family and my friends. I would like to thank them very much for always being there for me.

This thesis was founded by the CTI grants 8429.1 and 9636.1. The image data is courtesy of Teleclub and LiberoVision.

Contents

CHAPTER 1

Notation

Vectors and matrices are in bold letters. Parameters and functions are in lowercase italics and sets are in uppercase letters. If the elements of a set are listed, they are in curly brackets $\{\}$. If a vector or a matrix is written with elements, then brackets $()$ are used.

1.1 Glossary of Symbols

Throughout the thesis, the following variables have a fixed meaning.

C	The set of all real cameras in the scene
c	A placeholder for an arbitrary camera
c_i	A placeholder for camera i
$\boldsymbol{c}$	The center of projection of an arbitrary camera
$\boldsymbol{c}_i$	The center of projection of camera i
$\boldsymbol{I}_{c_i}$	The image of camera i
$\boldsymbol{P}_i$	The camera matrix of camera i
$\boldsymbol{s}_{c_i}$	The 2D camera shift vector for camera i and a given subject
$\boldsymbol{u}, \boldsymbol{v}$	Points in 2D. Coordinates usually given as $\begin{pmatrix} u \\ v \end{pmatrix}$
$\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{q}$	Points in 3D. Coordinates usually given as $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$
J	The set of body joints
j	A placeholder for an arbitrary joint
j_i	A placeholder for joint number i
$\boldsymbol{j}$	The 3D coordinates of an arbitrary joint
$\boldsymbol{j}_i$	The 3D coordinates of joint number i
$\boldsymbol{j}_{i,c_k}$	The 2D coordinates of joint number i in camera c_k

1.2 Glossary of Abbreviations

The following abbreviations are used in the thesis.

DLT	Direct Linear Transformation
EDT	Euclidean Distance Transform
GMM	Gaussian Mixture Models

CHAPTER 2

Introduction

In recent years, the quality of video and TV cameras has improved not only in terms of color and resolution. Recently, also the possibility of 3D recording, transmission and display has started its entry into the professional and the consumer market. 3D television is acclaimed to be the next revolution in TV broadcasting after the change from black and white to color devices. In 2010 the broadcaster ESPN transmitted for the first time a sports event live on a 3D TV network. Specialized 3D displays usually work with polarized layers on the display in combination with polarized glasses to show a different view for each eye. Other displays use time multiplexing in combination with shutter glasses.

Compared to standard 2D TV, a 3D TV gives the viewer a stronger perception of the 3D geometry of a scene, because it adds disparity to the image, i.e. shows to each eye the same object from slightly different positions. However, the viewpoint remains fixed as it was chosen by the broadcaster or producer. And even for them, once a scene is recorded, the viewing angle can not be changed any more.

The research field of *video-based rendering* (sometimes also referred as free viewpoint video) tackles this limitation. The goal is to render a scene, that was recorded by one or more cameras, from novel viewpoints where no real camera was placed. This so called virtual camera can then be placed by the broadcaster or the viewer *after* the scene was recorded. A large variety

of applications directly gain from this. Examples are telepresence, games, movie production, 2D to 3D conversion, forensics, interactive TV broadcasts, and sports analysis. Video-based rendering is an area of active research in the computer graphics community. There exists many different approaches, ranging from simple interpolations of two images to complete reconstructions of the 3D geometry of the scene.

A crucial element of video-based rendering is usually the human body. The reason for this is that a viewers eye is very trained in seeing and recognizing human bodies. If there are errors in the rendering of objects in the background, our perception usually neglects or even corrects them. But if there is a non human-like motion or unnatural distortion in the rendering of a human body, this is much more disturbing for the viewers eyes. This problem is referred to as the *uncanny valley* [Mori, 1970]. The uncanny valley is a region in the range between complete non-human characters on one side and a real human on the other side. In this region a viewer perceives the animation (or reconstruction) as very unpleasing, because it is close to a real image or animation of a human but contains slight errors, which are perceived as very disturbing and unnatural.

Due to this bias of the viewer on human characters, many related works focus in the reconstruction and view synthesis especially or only on human bodies. Recent results show that in studio setups, novel viewpoint renderings of the human body are possible with a realistic output [de Aguiar et al., 2008; Vlasic et al., 2008]. Other methods showed that for reconstructing and rendering a human body, the knowledge of the current body pose is very helpful [Ballan and Cortelazzo, 2008; Carranza et al., 2003; de Aguiar et al., 2008]. It reduces the volume that is possible to be occupied by this human and thus the amount of unknown variables. A pose estimation can be used to enforce plausible poses and thus more human-like renderings.

In this work we focus on conventional sports broadcasts footage [Hilton et al., 2011; LiberoVision, 2012]. The goal is to extend the creative freedom of an editor or director by providing the possibility to place a virtual camera in the stadium without having to change or add anything in the already existing sparse physical camera setup. This allows to have the perfect perspective and therefore perfect shot at any given time. As noted in Guillemaut et al. [2009] and Hilton et al. [2011] this setup is very challenging due to several factors.

- *Sparce camera placement:* There are typically only few moving cameras available that cover the interesting part of the scene and can be calibrated. In soccer they are positioned only on one side of the stadium.

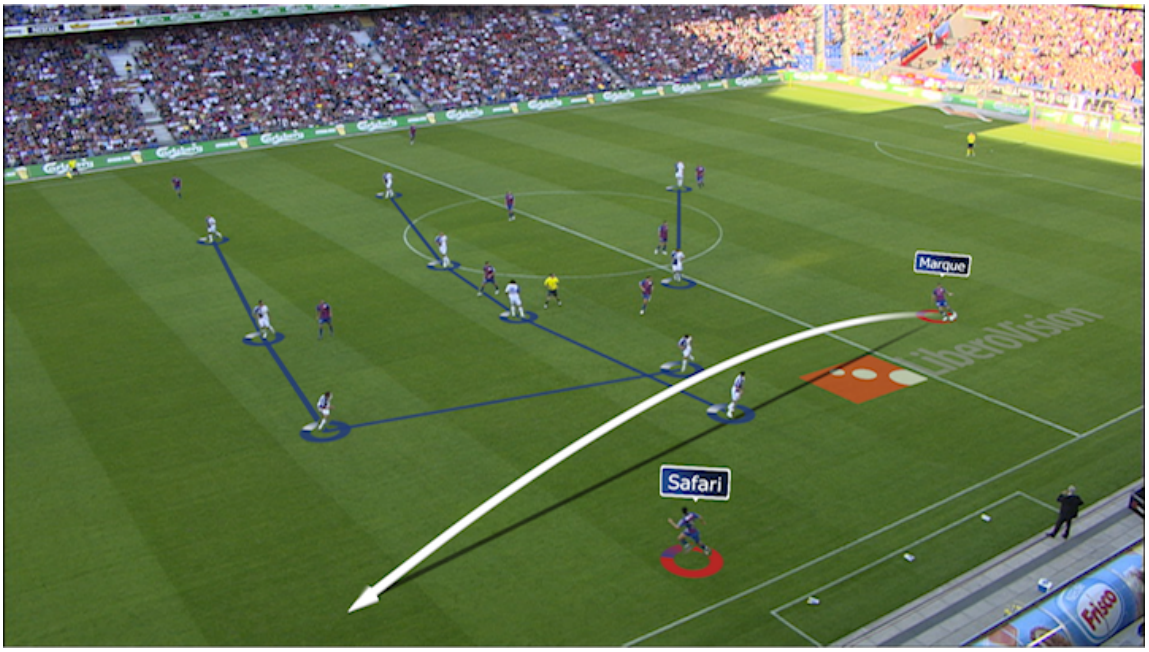


Figure 2.1: *Screen shot of a soccer game analyzed by LiberoVision technology.*

- *Low resolution:* Although the cameras provide high resolution images, they are usually set to be wide-angle for editorial reasons. Therefore, an individual player covers only a height between 50 and 200 pixels [Hilton et al., 2011].
- *Motion blur:* In sports broadcasts the player motion is usually fast and thus often results in motion blur.
- *Weak calibration:* Methods for per frame automatic calibration in such setups [Thomas, 2006] suffer from errors and typically contain no radiometric calibration.

All these factors have to be circumvented in order to create a convincing novel-view synthesis in such a challenging setup.

LiberoVision [LiberoVision, 2012] is a spin-off company from ETH Zurich that develops applications for video-based rendering of sports events. Their products allow a TV broadcaster to analyze a game with additional views from a virtual camera, artificial markings or drawings as well as changes in the scene like moving players in a freeze-frame. Figure 2.1 shows an example screen shot. In a collaboration with LiberoVision, we developed novel algorithms for video-based rendering to improve their system, reducing manual interaction and allowing the rendering of dynamic scenes.

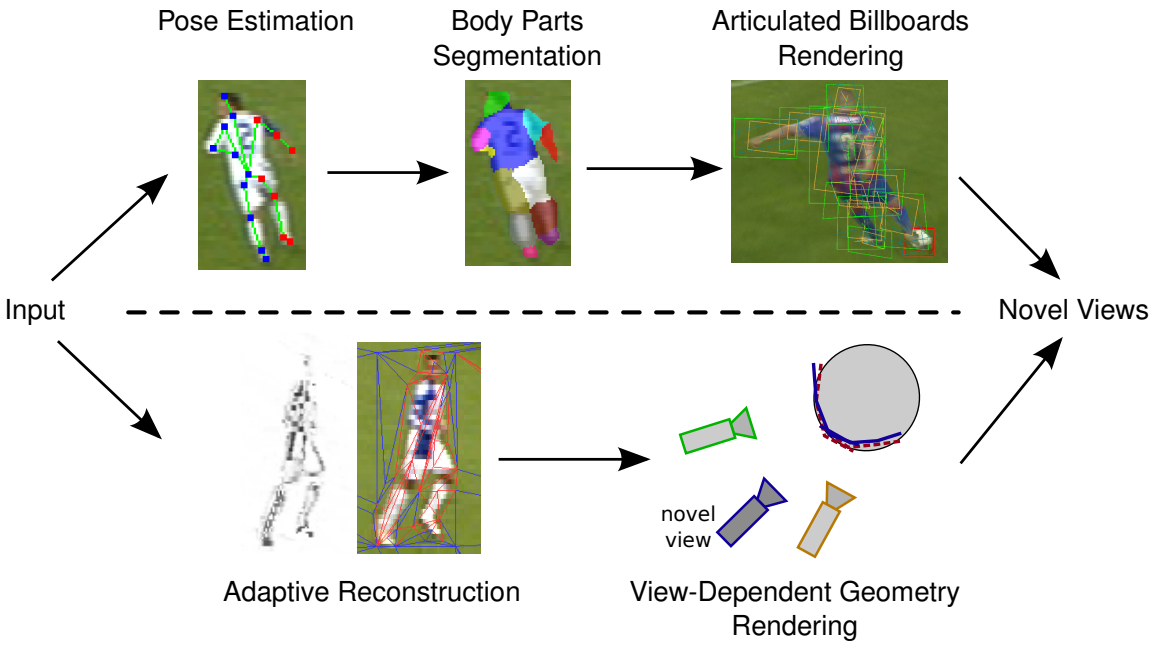


Figure 2.2: *Overview of the two solutions presented in this thesis.*

2.1 Overview

In this thesis we present several new methods for video-based rendering in challenging outdoor setups. This includes a body pose estimation, an improvement of feature match detection, an adaptive geometry reconstruction, a body parts segmentation method, two different geometric representations, a view-dependent geometry morph and view-dependent rendering methods. These algorithms can be bundled into two separate solutions for a full pipeline for video-based rendering, shown in figure 2.2.

The first solution is based on a new representation of the human body, called *articulated billboards*. It is illustrated in figure 2.2 in the upper part. Previous methods for novel-view synthesis of the human body use either simple representations like a billboard per subject, which suffers from ghosting artifacts due to the non-planarity of the human body, or they use more sophisticated methods like the visual hull, where a detailed reconstruction is achieved but it cuts off entire arms or legs when calibration errors occur. In a trade-off between these two extremes, articulated billboards allow renderings of novel views even in low resolution setups with inaccurate calibrations. They consist of a skeleton where each body part is represented as a fan of billboards and blended together with the other parts. On one hand, this copes with self-occlusions and preserves the 3D body pose, but on the other hand, it is still robust to calibration errors.

To construct the articulated billboards, we developed a body pose estimation for challenging outdoor setups. It consists of two stages and uses only coarse silhouettes from as few as two or three cameras as an input as well as inaccurate camera calibrations. The first stage is a data-driven pose estimation that results in a pose guess. In a sliding window approach, a silhouette comparison to a database of annotated sequences is done. Based on this, a 2D skeleton can be transferred onto the camera images. Using the camera calibration the 3D pose can be verified and selected as the best combination of 2D skeletons. In the same step also a camera calibration correction is computed per subject, that copes with errors due to, e.g., distortion of TV camera images. In a second stage, the resulting 3D pose guess is refined. First, the consistency between the frames is assured by applying left-right flips of arms and legs. Second, a space-time pose optimization is performed, which includes data-driven, silhouette-based, anthropometric and smoothness terms.

The 3D body pose is used to segment the 2D silhouette area into different body parts by a grow-and-shrink approach: a template model is fitted to the 3D skeleton pose and blown up to result in a grown version of the model. This is projected to the 2D images. Also a shrunk version is projected into the camera images. The segmentation is then guided by the areas that receive the same body part label from both, the grown and the shrunk projection. From these pixels a color model is built for every body part and used to segment the rest of the silhouette area. The advantage of this segmentation is that it adapts to the current appearance of the subject.

From the 3D pose the articulated billboards model is placed into the virtual space and billboards are attached for every part and camera. The billboards receive the sprites given by the corresponding camera and body part from the segmentation. Novel views can be rendered by a view-dependent blending of the articulated billboards.

The second solution for a full pipeline for video-based rendering is based on an adaptive geometry reconstruction and a view-dependent geometry morph. It is illustrated in figure 2.2 in the lower part. In the adaptive reconstruction, for every camera, a 2.5D triangulation is computed. This is done in several steps. First, the camera image is triangulated with large triangles. These triangles receive a depth value at every vertex, initialized by the distance to the ground plane. These depth values are optimized to fit feature matches, which are based on Daisy features [Tola et al., 2008] but enhanced by allowing near-epipolar matches. In this optimization, the triangles are not connected to each other. Therefore, even two triangles that share a vertex in 2D can obtain a different depth value at this vertex. The depth optimization is iterated with a subdivision of only those triangles whose color projections from other

cameras do not fit into the currently processed camera. This allows to refine only where it is needed and thus reduces the computation time. In a following step, the depths are refined according to back-projection errors evaluated at depths taken from neighboring triangles and random perturbations. Finally, the reconstructions from all cameras are merged into a single 3D model. The merge results in a complete reconstruction even of parts that are occluded in one camera.

Because of calibration errors, usually there is no position for a triangle vertex that is optimal for all camera views. Therefore, we propose a view-dependent geometry morph for the rendering of the resulting triangles. The geometry is morphed according to a transformation of the space. This transformation function is obtained from the non-epipolar feature matches. At every feature point, the amount and direction of the shift from the epipolar line gives the transformation of the space at this point. The resulting renderings achieve a full interpolation of the original camera views while still allowing arbitrary viewpoints with realistic renderings.

2.2 Principal Contributions

This thesis makes the following contributions:

- **Body Pose Estimation:** We propose an algorithm for human body pose estimation that consists of two major steps. First, a data-driven rough pose estimation is applied based on silhouette comparisons in a sliding window. Second, after a consistency check between frames, this pose is refined in a space-time optimization according to data-driven, smoothing and silhouette-based terms. This combination is able to robustly estimate body poses in outdoor setups with as few as two cameras.
- **Body Parts Segmentation:** A template-based segmentation of images into human body parts is presented. The segmentation uses a 3D pose estimation to deform a template mesh and project it into the 2D camera images. In a grow-and-shrink approach a thinner and a larger version of the mesh are used for this to detect reliable areas in the image. These areas are then used to learn a color model and to classify nearby unreliable pixels. This method adapts to different subjects and lighting conditions.
- **Articulated Billboards Representation:** We introduce a new representation for the human body that is based on a skeleton and billboard

fans attached at every body part. This representation is simple in reconstruction, storing and rendering.

- **Articulated Billboards Rendering:** We propose a real-time rendering method for articulated billboards that smoothly interpolates between original camera views. The billboards are blended for each body part and accumulated to the entire body. Due to the planar geometry of billboards, this rendering is robust against small calibration errors but still preserves the impression of the body pose.
- **Feature Match Filtering:** We present an extension to the Daisy feature matches that filters reliable correspondences. It allows near-epipolar feature correspondences but validates them by assuring symmetric matches.
- **Adaptive Reconstruction:** We propose a reconstruction method for outdoor setups with sparse camera placements and calibration errors. It starts with a simple planar approximation of the scene and adaptively refines this only where it is needed. The refinement is iterated with a depth optimization. An additional refinement is achieved by evaluating back-projection errors of neighbor depths and random perturbation. This reconstruction is done for every camera to obtain a 2.5D triangulation for each, that are finally merged into a 3D reconstruction. Our method works in challenging setups and copes with calibration errors.
- **View-Dependent Geometry:** Our view-dependent geometry morph deforms a reconstruction according to a force field. This force field is deduced from reliable feature matches. The resulting video-based renderings preserve interpolation of original camera images and smooth transitions between them or to arbitrary viewpoints in the vicinity of the original cameras.

2.3 Thesis Outline

The thesis is organized as follows:

- **Chapter 3** discusses previous work that is related to this thesis.
- **Chapter 4** gives a background to the topics discussed in this thesis. It introduces important terms and techniques used in video-based rendering.
- **Chapter 5** presents our human body pose estimation in detail.

- **Chapter 6** introduces the articulated billboards model and discusses methods for its construction, optimization and rendering.
- **Chapter 7** presents our adaptive geometry reconstruction and the view-dependent geometry morph.
- **Chapter 8** concludes this thesis and discusses possible future work. It also gives a comparison of the two different solutions for a video-based rendering pipeline presented in this thesis.

2.4 Publications

In the context of this thesis, the following peer-reviewed publications have been accepted.

M. GERMANN, T. POPA, R. KEISER, R. ZIEGLER, and M. GROSS. Novel-View Synthesis of Outdoor Sport Events Using an Adaptive View-Dependent Geometry. Accepted for *Proceedings of Eurographics (Cagliari, Italy, May, 2012), Computer Graphics Forum, vol. 31, no. 2.*

This paper presents an adaptive reconstruction method for human bodies in challenging outdoor setups as well as a view-dependent geometry and blending for realistic rendering of novel viewpoints.

M. GERMANN, T. POPA, R. ZIEGLER, R. KEISER, and M. GROSS. Space-time Body Pose Estimation in Uncontrolled Environments. In *Proceedings of 3DIMPVT (Hangzhou, China, May, 2011)*

This paper proposes a human body pose estimation that is robust against calibration errors and low resolutions. It works with as few as two cameras despite wide baselines.

M. GERMANN, A. HORNUNG, R. KEISER, R. ZIEGLER, S. WÜRMLIN, and M. GROSS. Articulated Billboards for Video-based Rendering. In *Proceedings of Eurographics (Norrköping, Sweden, May, 2010), Computer Graphics Forum, vol. 29, no. 2, pp. 585-594.*

This paper introduces articulated billboards, a representation and rendering method for human bodies in uncontrolled environments.

During the course of this thesis, the following peer-reviewed technical papers have been accepted which are not directly related to the presented work.

I. K. PARK, M. GERMANN, M. D. BREITENSTEIN, and H. PFISTER. Fast and Automatic Object Pose Estimation for Range Images on the GPU. In *Machine Vision and Applications, 2010, vol. 21, no.2, chapt. 749, pp. 749-766.*

This paper proposes a parallel data-driven object pose estimation applicable for depth images.

G. GUENNEBAUD, M. GERMANN, and M. GROSS. Dynamic Sampling and Rendering of Algebraic Point Set Surfaces. In *Proceedings of Eurographics (Crete, Greece, April, 2008), Computer Graphics Forum, vol. 27, no.2, pp. 653-662.*

This paper introduces an explicit and more generic solution to algebraic point set surfaces as well as a fast sampling and rendering algorithm based on forward warping and integrated APSS.

CHAPTER 3

Related Work

This thesis covers different fields in computer graphics and computer vision, including body pose estimation, feature match filtering, segmentation, scene reconstruction, representation, morphing and rendering methods. As a consequence, there also exists a huge variety of works that are related to this thesis. In order to summarize these works in a reasonable order and extent, we structured this chapter into two main parts that cover the most relevant fields. The first part (section 3.1) is about body pose estimation and covers the work that is related to chapter 5. The second part (section 3.2) focuses on the synthesis of novel views, which is related to the chapters 6 and 7 of this thesis.

We do not, however, summarize related work on feature match improvement and segmentation. Both topics are minor elements of our own work and also very specific to the application we used these methods for. A good overview on feature matching can be found in the survey by Tuytelaars and Mikolajczyk [2008]. For related work on segmentation we recommend the survey by Freixenet et al. [2002].

3.1 Body Pose Estimation

Given one or more images of a human subject, body pose estimation aims to find the positions of the articulated parts of the body in 2D or 3D. In our setup, multiple views of the same subject as well as camera calibrations are available. Therefore, a switch from 2D to 3D and vice-versa is possible (section 4.5), allowing to use also 2D pose estimations to be lifted into a 3D body pose.

A hierarchical or a linear ordering of the vast extent of work in the field of body pose estimation would be difficult and confusing. For clarity, we decided to present the related work in pose estimation in "matrix" form comparing the methods against different criteria. Instead of discussing the related work hierarchically, we simply go through all the criteria, describe them and directly give the most prominent or most related examples for each criterion.

We focus on more recent and related work. For a more detailed overview of the work in body pose estimation until 2006, we recommend the surveys by Moeslund and Granum [2001] and Moeslund et al. [2006].

3.1.1 Overview

Our overview matrix mentioned above is shown in figure 3.1. It lists the works in body pose estimation that is most relevant to ours, sorted according to the publication date. The reason for the chronological order is to show a bit the trends over the years.

We derived eight criteria to distinguish between the different approaches. In the following subsections we go over these criteria and discuss the corresponding related work. In subsection 3.1.10 we focus the discussion on those methods that are specifically designed for or successfully applied to sports broadcasts.

3.1.2 Detection vs. Tracking

In video sequences body pose estimation can be applied in every frame independently. However, the estimation of a body pose can gain from an already estimated pose of the previous (and/or following) frame. The use of these previous estimations is called tracking, whereas the estimation without is called detection or single frame pose estimation. Tracking can be done on just the position (e.g. a bounding box) of the subject or more detailed on the individual joints. Approaches that temporally smooth the estimated poses in

Method	Tracking or only estimation	Active	Marker	Depth	Multi-Camera	Template-Based	Parts-Based	Silhouette-Based
Hogg [1983]	tracking	no	no	no	no	yes	no	no
Felzenszwalb and Huttenlocher [2000]	estimation	no	no	no	no	yes	yes	no
Sullivan and Carlsson [2002]	estimation	no	no	no	no	no	no	no
Efros et al. [2003]	estimation	no	no	no	no	no	no	no
Ramanan and Forsyth [2003]	estimation	no	no	no	no	yes	yes	no
Carranza et al. [2003]	tracking	no	no	no	yes	yes	no	yes
Grauman et al. [2003]	estimation	no	no	no	yes	no	no	yes
Theobalt et al. [2004]	estimation	open	open	3D volume	open	no	no	volume
Sigal et al. [2004]	tracking	no	no	no	yes	yes	yes	no
Mori [2005]	estimation	no	no	no	no	yes	no	no
Ramanan et al. [2005]	estimation	no	no	no	no	yes	yes	no
Jaeggli et al. [2005]	tracking	yes	no	sparse	no	yes	no	no
Agarwal and Triggs [2006]	tracking	no	no	no	no	no	no	yes
Park and Hodgins [2006]	estimation	yes	yes	no	yes	yes	yes	no
Mori and Malik [2006]	estimation	no	no	no	no	no	no	no
Ferrari et al. [2008]	estimation	no	no	no	no	no	yes	no
Ballan and Cortelazzo [2008]	tracking	no	no	no	yes	yes	no	yes
de Aguiar et al. [2008]	tracking	no	no	no	yes	yes	no	yes
Urtasun and Darrell [2008]	estimation	no	no	no	no	no	no	yes
Gupta et al. [2008]	estimation	no	no	no	yes	yes	yes	no
Andriluka et al. [2009]	estimation	no	no	no	no	no	yes	no
Fossati et al. [2009]	tracking	no	no	no	no	yes	only hands/feet	yes
Andriluka et al. [2010]	tracklets	no	no	no	no	no	yes	no
Shotton et al. [2011]	estimation	yes	no	yes	no	no	yes	no
Yang and Ramanan [2011]	estimation	no	no	no	no	yes	yes	no
Park and Ramanan [2011]	open	no	no	no	no	yes	yes	no
Ours ([Germann et al., 2011] and Chapter 5)	estimation	no	no	no	yes	yes	no	yes

Figure 3.1: *Overview of the related work in body pose estimation in chronological order. "open" means that both ways are possible.*

a post-process (e.g., with a Kalmann filter) as well as approaches that use a temporal window to estimate the pose are not considered as tracking in this thesis.

An early example of tracking was presented by Hogg [1983]. They use a template model to compare edges to the image. The result of the previous frame is used to constraint the movements between the previous and the current frame by constraining the angular changes in their body model between these frames. This is one of the most used temporal constraints to do tracking.

Another example of an approach that uses tracking was presented by Ballan and Cortelazzo [2008]. At the beginning, a detailed mesh of the subject is acquired and a skeleton is fitted into it. This mesh is then deformed according to the current pose guess of the skeleton. To evaluate the guess, they use correspondences from the 3D mesh to 2D positions. Using optical flow, the 2D correspondences are transformed from frame t-1 to frame t. The distance from the 2D positions of the current pose guess to these transformed positions from the last frame directly results in an error metric. A second error measure is a silhouette matching similar to ICP but in 2D between the projected model silhouette and the real silhouette. The method is applied to a studio setup with four cameras.

The main challenge of algorithms based on tracking is that they are inherently jeopardized to *drift*. If a pose estimation in a frame fails then this directly negatively affects the estimation in the subsequent frame, usually causing it to fail too. This can result in continuous bursts of wrong estimations without the ability to recover. Therefore, we use in our approach for the initial pose guess a sliding window approach that gains from information of neighbor frames but does not use previous estimations.

3.1.3 Active vs. Passive Methods

While passive methods only use one or more camera images to estimate the body pose, active methods influence the environment to get additional information about the subject. This active change of the environment can be the projection of different light patterns onto the surface of the subject. The knowledge of which pattern was projected at which time instant is then used to triangulate a points position according to the calibrations of camera and projector [Scharstein and Szeliski, 2003]. The result is a depth map that can be used for methods described in section 3.1.5. Jaeggli et al. [2005] project lines onto the human body to be able to determine depth information, and

especially deal with occlusions. These projections are made at points-of-interest according to a direct feedback from the pose estimation.

Another active method uses infrared light to estimate the depth according to the time-of-flight or according to structured patterns. Recently, the Kinect [Kinect, 2010] by Microsoft has entered the consumer market. It contains camera, infrared projector and sensor as well as microphones in one single and affordable device. Shotton et al. [2011] showed that this can be used to do body pose estimation even in arbitrary home setups. They use the depth information for a classifier at every pixel and finally compute spatial modes of the inferred per-pixel distributions to get a pose estimation.

Active methods usually return more accurate pose estimations than passive methods. However, they require an interaction with the scene that is not always possible. In outdoor setups the range for the interaction is limited due to technical issues. One example for time-of-flight cameras is the wave length that causes a trade-off between accuracy and depth range. Another example is also that the maximal depth is limited due to the energy loss of the signal.

3.1.4 Marker-Based Methods

Instead of projecting light patterns onto the subject, colored markers can be stitched directly onto the body. The advantage of them is, that we know where on the body the marker is, while with light projections the position on the surface has to be determined first. Marker-based methods can be seen as active methods, since they also change/influence the scene by placing markers.

A common device to reconstruct the 3D positions of markers is the Vicon system [Vicon, 2012]. While most marker-based methods use about 40-60 markers, Park and Hodgins [2006] extend this to 350 markers to get a very detailed pose estimation and animation of the skin. An overview of marker-based methods for movie production can be found in the work of Menache [1999].

Since in our target setup, outdoor sport games, placing of artificial markers is usually not possible, we only consider methods without markers in the remainder of this thesis.

3.1.5 Depth Data vs. Color Only

In order to reduce the disturbing visible light interactions in the scene, depth scanners can be used that work with non-visible light. Most notable examples are the Swiss Ranger [Oggier et al., 2005] and Microsofts Kinect [Kinect, 2010]. Both devices use infrared light to project known patterns on the scene for computing a depth map. Recently, methods were developed [Shotton et al., 2011], that achieve astonishing results. However, the range of such infrared scanners focuses on indoor setups and does not scale to larger scenes.

The method presented by Theobalt et al. [2004] takes as an input not only 2.5D depth data but a full 3D volume. This can be acquired by either using multi-view silhouettes or multi-view stereo. The skeleton is then fitted into this geometry.

As already mentioned, in our setup depth data is usually not possible to acquire. Therefore, we focus the following sections mostly on works that do not rely on depth data.

3.1.6 Multi-Camera Methods

Estimating the body pose in one single image without depth information is difficult due to inherent ambiguities. These ambiguities are reduced as soon as more than one image is available, taken from cameras at different viewpoints. Parts occluded or difficult to distinguish from each other in one camera might be covered better on another camera. For this reason, pose estimation in multi-camera setups have shown to be more accurate than with only one camera.

Some algorithms were proposed, that deform a previously acquired template mesh of the subject to fit the current silhouettes in a multi-view setup [Carranza et al., 2003; Ballan and Cortelazzo, 2008; de Aguiar et al., 2008]. These methods provide very good results, but impose restrictions on the setup. They require a carefully built studio setup, many well calibrated cameras with high resolutions and very good spatial coverage.

The work by Gupta et al. [2008] is based on 2D part detections (see section 3.1.8). They improve the 2D part detections by using a multi-camera setup and verifying these detections in 3D, resulting in reliable handling of occlusions and self-occlusions.

Monocular video sequences are easier to acquire, but pose estimation in single camera setups suffer from an inherent depth ambiguity. Therefore, if there

are more than one camera available, it is recommendable to also use them. In outdoor sports setups, there are usually several cameras. Even though many of them are difficult to calibrate or do not focus on the interesting part of the scene, usually at least two of them can be used. Our method is able to profit from the multi-camera setup to solve for ambiguities and occlusion problems, but nevertheless only 2 cameras are already sufficient.

3.1.7 Template-Based Methods

Template-based methods use a virtual representation of the human body that is drawn or rendered to be compared to the input. Estimating the 3D pose can be achieved by minimizing this difference between the rendering and the input.

An example for a simplified template model was presented by Felzenszwalb and Huttenlocher [2000]. Pictorial structures are designed for recognition tasks but also return an estimate of the 2D body pose when applied to humans. The model used in this case is a pictorial structure, that is a collection of parts (body parts in our case) which are connected in spring-like manner. By matching these pictorial structures to monocular camera images, the subject is detected and its pose estimated.

For multi-camera studio setups, Carranza et al. [2003] use a generic template model which is fitted into the silhouettes. The model is adjusted in length and size of the limbs such that it optimally matches the evidence. Similarly, de Aguiar et al. [2008] fit a model, but use an acquired scan of the subject instead of a template model.

An interesting segmentation-based approach for outdoor setups was presented by Mori [2005]. They use only a simple 2D template model similar to the cardboard persons [Felzenszwalb and Huttenlocher, 2000; Ramanan and Forsyth, 2003] but with occlusion order. Instead of working on pixels, they employ super-pixels [Ren and Malik, 2003], which are an over-segmentation. The border of a half-limb usually lies on a line along several super-pixels. Together with kinematic and symmetry constraints, this is directly used to evaluate pose guesses and find an optimal pose estimation.

Our method for pose estimation uses a template mesh, but only for segmentation and for finding left/right flips of arms or legs. This makes it robust against calibration errors and differences between the mesh and the actual subject, which usually occur in outdoor sports setups.

3.1.8 Parts-Based Methods

Instead of evaluating the match of an entire pose guess, part-based methods focus on the detection of segments of the body. Part-based methods reduce the huge space of possible body poses into a few smaller spaces of possible body part poses. Some of these works directly use a body template model or a skeleton model to match the body parts onto detected body parts in the input image.

Sigal et al. [2004] developed a bottom-up approach that uses detections of body parts together with a loosely-connected body model. A belief propagation framework includes connections between body parts of the same frame but also between frames. Nevertheless, the detection of the body parts is very noisy and often fails to distinguish left-right ambiguities, the belief propagation finds a stable pose estimation.

Ramanan and Forsyth [2003] and Ramanan et al. [2005] use appearance models for the individual body parts. Ramanan and Forsyth [2003] cluster the appearance of body parts over all frames to use it as a model in difficult frames, e.g. when the person is not moving. Ramanan et al. [2005] improved this by searching for easy-to-detect poses, i.e., walking poses, where the body parts models are learned. These models are then directly used to detect body parts in other frames with more difficult poses.

Andriluka et al. [2009] model a prior for the possibility of the entire pose. The likelihood, however, is a product of per body part likelihoods that are computed independent of the other body parts. This can be taken as an approximation if there are not too many self occlusions. It reduces the search space and allows for a dense evaluation of all part positions and rotations. The descriptor used for the body parts captures the distribution of locally normalized gradient orientations in a log-polar histogram. Andriluka et al. [2010] extend this to use the tracking of a 2D bounding box and the viewpoint as an input to the 3D pose estimation. This improves the result especially in ambiguous poses or occlusions. Ferrari et al. [2008] extend the work by Ramanan et al. [2005] by reducing the search space of possible positions and rotations of a body part to a detected foreground area. Additionally, they use a spatio-temporal inference that penalizes for large movements of the body parts between consecutive frames.

3.1.9 Silhouette-Based Methods

As stated by Agarwal and Triggs [2006], using silhouettes for body pose estimation has three main advantages:

- Silhouettes are easy to retrieve, i.e. by background subtraction
- They do not change if the appearance of the surface changes
- They already contain much information about the 3D pose

The last point can be seen that for a human, seeing only silhouettes from several cameras is already enough to determine the body pose, with the exception of left-right ambiguities.

These advantages lead to various approaches for pose estimation that are based on silhouettes.

The "shape+structure" model by Grauman et al. [2003] only uses silhouettes as input and no model matching in 3D is required. The model has to be learned from synthetic renderings from the same camera positions as the input and thus assumes their calibrations to be known in advance. Their experiments and evaluations are restricted to walking poses.

Agarwal and Triggs [2006] generate silhouette descriptors and use a Bayesian nonlinear regression to retrieve a model that is simple but able to represent various pose types, including types that were not in the training set. Even with only monocular images, they are able to demonstrate the effectiveness of using only silhouettes to estimate the body pose.

A local probabilistic regression method is presented by Urtasun and Darrell [2008]. Their approach consists of an online learning of appearance-to-pose mappings using local Gaussian Processes. This local approach it is able to handle multimodal outputs and does not just average them, which would result in completely wrong mappings.

3.1.10 Application on Sports Broadcasts

Most of the above works focus on studio setups or setups where the subjects are covered well or in many cameras. For sports broadcasts, this is usually not the case and there are only few methods tailored to the specific challenges of this type of data.

For closeup views, a deformable shape matching can be used as in Sullivan and Carlsson [2002] and Mori and Malik [2006]. They use a coarse head and body detection as a starting point. Then the body shape, represented as points

on feature lines, is matched to shapes in a database by point correspondences. Sullivan and Carlsson [2002] also employ constraints on the color and on the spatial arrangement of the points to find the optimal match and thus the optimal pose in the database. Sullivan and Carlsson [2002] use this to detect a certain movement in a closeup sequence of a tennis player and Mori and Malik [2006] apply the shape matching to walking sequences as well as to speed skating.

A promising approach was presented by Efros et al. [2003]. Even though their method targets on action recognition, they show that it is possible to use this to also estimate a rough pose of legs, torso and head. The input data is matched to similar sequences in a database by comparing the optical flow of the human figure as a descriptor of the current pose and movement. The results show that the action of a subject covered by only 30 pixels height can be robustly detected. The already mentioned approach by Ramanan et al. [2005] was also applied to videos from sports events. But compared to Efros et al. [2003] they work on closeups where the subject covers most of the image and thus has a higher resolution.

Fossati et al. [2009] present a method that in a first step tracks hands and feet. The results are directly used in a Gaussian Process mapping to the estimation of the full body pose, which reduces the size of the required learning database. The body pose is refined by a image-based objective function. This approach can be applied to monocular sequences even with low resolution. It is able to learn from one sport type to be applied to similar ones. They apply this method to golf swings, skiing and skating, where the space of poses is more limited than in soccer scenes.

In their recent work, Yang and Ramanan [2011] extended the idea of pictorial structures [Felzenszwalb and Huttenlocher, 2000] to detect parts at particular orientations. This uses the fact that, e.g., in images there are usually many vertical and even more horizontal gradients in the background. Therefore, it is better to search for diagonal gradients to detect limbs. They also apply this successfully to pictures from outdoors sports. These images are mainly at a higher resolution than in our setup. However, they achieve astonishing results, even though they only use monocular views.

Very recently, Park and Ramanan [2011] introduced n-best algorithms to the computer vision community by presenting an algorithm to compute several pose guesses for an input image. The general idea behind this is, that this can be used to later on select the best out of these pose guesses according to temporal information or any other higher order knowledge. Applied to sports scenes, it is able to return the most possible poses, e.g. in occlusion ambiguities. Their method is based on the work of Yang and Ramanan [2011]

and applied to similar test scenes, where the subjects are covered at a higher resolution than in our setup.

3.2 Novel-View Synthesis

Novel-View Synthesis aims to render a real scene from a virtual camera at a position where no real camera was recording. Beginning with the pioneering work of Kanade et. al [Kanade et al., 1995], a lot of research on novel-view synthesis has been done in computer graphics. To give an overview of the work that is related to our methods presented in chapter 6 and chapter 7, we order the works according to how much 3D geometry they employ. This starts with algorithms that do not use a 3D proxy at all but interpolate the views directly in 2D. It ends with methods that reconstruct a pixel accurate full 3D geometry of a scene to render novel viewpoints. The following subsections will survey the related work in this order. Section 3.2.7 discusses works on the correction of rendering errors due to inaccurate calibrations or wrong geometry. The last part (section 3.2.8) will focus the related work on outdoor sports.

3.2.1 2D Methods

One idea to generate novel views of a scene is to interpolate existing camera images directly in 2D, without any 3D geometry at all. Morphing [Seitz and Dyer, 1996] is a popular method to create such interpolations. While the original approaches used a lot of human interaction to find correspondences, Yamazaki et al. [2005] use automatic feature detection to achieve a 2D morphing function for interpolation. Because of the restriction to bijective mappings between the images, morphing can not solve correctly at occlusions and thus results in shrinking or growing of image parts at such regions. Connor and Reid [2003] use layers instead of morphing. This can cope with occlusions since every layer gets an order number assigned which allows for depth sorting. Both methods, however, are only able to interpolate between images which is a camera flight along only a single line. Arbitrary views, i.e. from camera positions that are not on this line, are not possible.

Mahajan et al. [2009] extend this in several ways. On one hand, they search for each pixel an optimal path from its position to the position in a second cameras view in the gradient domain, by searching for an optimal transition pixel. This also prevents from generating holes in the interpolation and implicitely solves for occlusions. On the other hand, they also allow interpolations in the

area between more than two cameras and not just along a line between two cameras. However, they achieve this, i.e. for four cameras, by simply first interpolating for two camera pairs separately and then interpolating between the resulting two images.

Another extension to 2D image interpolation was made by Lipski et al. [2009]. They assume the cameras to be roughly placed on a sphere and allow interpolations on the surface of the sphere. Similar to the method of Mahajan et al. [2009], this still does not allow to fly into the scene or outside the range of the cameras, but only achieves interpolations in the area of the sphere between these cameras. To achieve this, they use a setup consisting of cameras with less than 10° baseline.

Light field rendering, introduced by Levoy and Hanrahan [1996] represents the scene as the collection of all rays between two planes, each of them discretized by a grid. Since these rays define the full plenoptic function, i.e. all paths of light trough the scene, the scene can be rendered from any point on the plane. The acquisition of the light field is achieved by a camera array that matches the grid on one of these planes. The views between the cameras are then interpolated in the rendering.

All these 2D interpolation approaches work well for dense camera setups, which are not available for sports broadcast setups. If only few cameras are available, it is necessary to imply a geometric proxy with 3D information about the scene that improves the interpolation [Buehler et al., 2001; Siu and Lau, 2004].

3.2.2 Lumigraph and Unstructured Lightfield/Lumigraph

The lumigraph, developed at the same time as the light field and presented by Gortler et al. [1996] includes a depth correction of rays to compensate for objects that are not in the focal plane. For this, a rough geometry of the scene is required.

Heigl et al. [1999] extend the light field / lumigraph approach to also use depth maps of the scene as well as to use unstructured camera setups, i.e., to not limit the cameras to be placed in a structured array. They use an adaptively refined geometry mesh according to depth from multiview stereo. This geometry is computed in real-time per view. Even though this method still uses dense camera setups, it was the first approach of an unstructured lumigraph / light field rendering. They also introduced the idea of a view-dependent geometry and not just a view dependent blending.

As a similar extension to the lumigraph, the unstructured lumigraph presented by Buehler et al. [2001] describes the full range from using no geometry at all to using the full geometry information about the scene. In this generalization the first end corresponds to the light field, where no geometry is used. The second end corresponds to view-dependent texture mapping [Debevec et al., 1996], where a 3D model of the scene is present and the camera images are projected and blended on top of this geometry.

Based on the unstructured lumigraph, Matusik and Pfister [2004] presented an entire system for 3D TV acquisition, transmission and display. It consists of an array of cameras to acquire the lightfield/lumigraph and an array of projectors together with lenticular screens capable of front or rear projection.

3.2.3 Billboarding

For challenging outdoor setups, a very simple approximation of the geometry is sometimes sufficient because it is more robust to calibration errors. One of the simplest are billboards, which were initially used for fast rendering of synthetic data [Aubel et al., 1999; Décoret et al., 2003; Behrendt et al., 2005; Andújar et al., 2007]. Whereas these methods are able to efficiently subdivide the geometry of a model into impostors, they only focus on artificial data, i.e. rely on given 3D models.

Billboarding can also be used for acquired data of real objects. Their advantage over more sophisticated models is that even when calibration errors occur, billboards simply shift the entire subject in the rendering. This introduces more ghosting (duplication of body parts) but prevents from losing parts of the body. The interpolation of original camera images will be preserved and it will not introduce any cracks into the subject. In the following, we will focus on billboarding for non-synthetic data.

Hayashi and Saito [2006] and Inamoto and Saito [2007] presented methods based on billboarding. As they have shown, the construction and rendering of billboards is very fast and produces novel-view results that are of similar quality to the input images, especially when viewing from positions close to the original cameras. However, due to the planar representation of billboards, there are many ghosting artifacts because the human body can usually not be approximated as a plane.

To overcome these ghosting artifacts, the representation of billboards was extended in the work of Waschbüsch et al. [2007] to 3D video billboard clouds. On top of each billboard, they add a topology, a displacement map. With this, a much better approximation of the surface is possible and thus the ghosting

is nearly removed at all. To acquire these depth maps they employ structured light projectors. Unfortunately, this is not possible in outdoor sports games.

Another way of reducing ghosting artifacts was presented by Ballan et al. [2010]. They also use billboards for the main subject and compose this with a background that was reconstructed as a mesh based on point correspondences. Their method allows flights from one camera into another. The ghosting is reduced by finding the optimal transition point to blend from one cameras billboard into the one of another camera. This optimal point is chosen to be where the artifacts are as small as possible and not noticed by the viewer.

Microfacets [Yamazaki et al., 2002; Goldlücke and Magnor, 2003] are tiny billboards that turn view-dependently such that they are always perpendicular to the viewing direction. With this they can approach complex object surfaces much better than rigid models. An example for this is fur or other very thin structures that was acquired with high resolution textures.

Le et al. [2005] proposed a method to extract billboards from optical flow. They use generated input images from synthetic models with high quality to obtain a dense point cloud from the optical flow. In a RANSAC approach planes are fitted to this point cloud. This works well in controlled (synthetic) setups, but relies on a good optical flow estimation.

Our articulated billboards (chapter 6) extend the billboard representation by employing a pose estimation to represent articulated body parts in separate billboard fans. This does not rely on depth data, good optical flow or high resolutions but still reduces the ghosting to a minimum.

3.2.4 Visual Hull

The concept of the visual hull was introduced by Laurentini [1994]. We give a more detailed description of it in section 4.6.1 and, therefore, here just quickly mention the general idea. For an object viewed by several cameras, the visual hull is the set of all points in 3D that projects in all cameras to the area inside the silhouette. The resulting volume is a convex approximation to the objects geometry and can be used to render the scene from an arbitrary viewpoint with the video camera images as textures [Laurentini, 1994; Matusik et al., 2000; Li et al., 2004; Grau et al., 2007; Petit et al., 2010; Gross et al., 2003]

The exact view-dependent visual hull [Miller and Hilton, 2006] renders only a view-dependent subsample of the visual hull similar to primary rays in raytracing. With this, the geometry is still static and does not change view-dependently, but the method accelerates the rendering process.

Visual hull methods are suitable for novel view synthesis of single objects. In crowded scenes, unless a large number of cameras is employed, extraneous so-called phantom geometry is generated [Franco and Boyer, 2009]. A more severe problem for uncontrolled outdoor setups is that, especially when using distant cameras, small calibration errors can cause large errors. In a soccer scene with low resolution coverage of the players, entire arms or legs are cut off because of this.

Guillemaut et al. [Guillemaut and Hilton, 2011; Guillemaut et al., 2009; Hilton et al., 2011] address many of the above mentioned challenges of visual hull approaches for free-viewpoint video in sports broadcasting. By jointly optimizing scene segmentation and player reconstruction they achieve a better reconstruction of the visual hull. The conservative visual hull initially enlarges the visual hull to reduce the cutting of body parts due to calibration errors [Kilner et al., 2007]. These approaches are leading to a more accurate geometry than the visual hull, but still require a fairly large number of cameras (6-12).

Inamoto et al. [Inamoto and Saito, 2002] also use silhouettes and match dense correspondences on the silhouette borders using epipolar lines. When working with weak calibrations, these correspondences will suffer from the same drawbacks as the visual hull. To still get reliable correspondences they rely on manual interaction to select them, especially when the silhouettes overlap.

3.2.5 Stereo

Stereo methods reconstruct a 2.5D image (range image) by assigning to each pixel in a camera also a depth value, which is usually done by directly finding for each pixel the corresponding pixel in an other camera [Marr and Poggio, 1979]. This can be extended to use more than two cameras to increase the range of possible novel viewpoints. An example for this is the work by Kanade et al. [1997], where over 50 cameras were used for a stereo-based reconstruction.

The stereo reconstruction by Bradley et al. [2008] targets on garment for which it generates a dense geometry. The results are very convincing, but dense reconstruction is difficult to do in wide-baseline camera setups.

An alternative approach suitable for setups where dense feature matching is difficult to accomplish, is patch based reconstruction [Zitnick et al., 2004; Fraundorfer et al., 2006; Stich et al., 2008; Sinha et al., 2009; Gallup et al.,

2010]. Patch based reconstruction methods only require a sparse set of correspondences, since between two cameras they only match patches instead of matching every pixel. However, these methods are usually limited to objects with a strong planarity assumption [Fraundorfer et al., 2006; Sinha et al., 2009; Gallup et al., 2010], and thus not suitable for reconstructing players. Zitnick et al. [2004] use a layered depth representation. Starting with a 2D segmentation into small patches, they assume first that each patch has a single disparity. Then they enforce constraints such that neighboring patches with similar colors should receive similar disparity. Stich et al. [2008] do a bottom-up approach where super-pixels are merged if they are on the same plane. This requires accurate camera calibrations where the matches between two cameras are correctly on their corresponding epipolar line.

Our adaptive view-dependent geometry (chapter 7) only uses reliable feature matches but in combination with back-projection errors to adaptively subdivide and refine a 2.5D triangulation where it does not match the evidence. This does not use a planarity assumption. Also, the merged 3D reconstruction is rendered using a view-dependent geometric morph that corrects for calibration errors, that occur in low resolution outdoor setups.

3.2.6 Template-Based Methods

When rendering novel views of persons such as players in a sports game, the knowledge about the nature of the object, that it is an articulated character, can be used as a prior information. Similar to pose estimation approaches discussed in section 3.1.7, a template model of the human body can be fitted into the evidence. However, this fitted template model can be used also for the rendering.

Hornung et al. [2007] showed the advantage of template-based methods even on monocular images. Their approach does not target on novel-view synthesis, but if a skeleton fit onto only one single image of a person or character is given, their method allows to change the pose of the subject. The template is used to guide the segmentation and correctly solve for occlusions and perspective.

Carranza et al. [2003] not only deform a generic template model but even adjust body parts and muscle sizes to fit the silhouettes of several input cameras. The rendering at depth discontinuities is improved by assuring that for vertices the texture information of only those cameras is used where the vertex is assured to be visible. A vertex is classified as visible in a camera if and only if it is visible in both the original and slightly displaced view.

The methods by Vlasic et al. [2008] and by de Aguiar et al. [2008], presented at the same SIGGRAPH conference, both use a previously acquired template mesh and deform this to fit the actual input frame. This template mesh is not a generic body model but a detailed scan of the current subject which is usually acquired at the beginning when the user of the system is asked to stand in a predefined pose. This reduces the degrees of freedom (e.g. body parts lengths) in the per-frame fitting process.

These template-based methods target on studio setups with accurately calibrated cameras where the subject is well covered in the camera views.

3.2.7 Error Correction

In outdoor setups, where camera calibrations are usually not correctly estimated, it is impossible to reconstruct the correct geometry. Since the rendering is usually done by projective texturing with view-dependent blending weights [Buehler et al., 2001], the errors not only result in wrong shapes and thus shifts of depth discontinuities, but also in shifted projections when texturing the geometric proxy. As we have seen already in the method by Carranza et al. [2003], especially at depth discontinuities, the rendering has to be done carefully to prevent from wrongly rendering body parts with the texture from another part. Several other methods were developed that cope with rendering errors caused by inaccurate calibrations or geometries.

A popular method to address this problem are floating textures [Eisemann et al., 2008]. The geometry is rendered and shaded independently from the first camera and from the second camera. Then, in image space, the two images are locally aligned to eliminate the ghosting and blurring artifacts. For the alignment they use the optical flow to get a correct warping, which requires a geometric proxy that is already close to the correct shape and also dense feature point matches.

The work by Taneja et al. [2010] includes a volumetric reconstruction of outdoor scenes. They detect and remove ghost volumes. These ghost volumes are wrongly detected voxels at ambiguous positions that are occluded in some cameras. By back-projection into the camera views, such ghosts can be located.

Ambient point clouds, introduced by Goesele et al. [2010], detect uncertain geometry in the reconstruction. These uncertain parts are then not rendered as rigid geometry but as a blurred area over possible positions of this geometry part. This does not correct the geometry but results in a statistically better approximation and reduces the perceived error due to smooth transitions.

3.2.8 Application on Sports Broadcasts

Some of the related work is also targeted directly on the synthesis of novel viewpoints in outdoor sports, similar to our work. We would like to have a special focus here on those.

The already mentioned work by Kanade et al. [1995] pioneered the novel-view synthesis in sports games. Their results consisting of scenes of football games clearly set a milestone and showed the possibilities and usefulness of renderings from virtual cameras. In contrast to our work, they use many cameras additionally placed in the scene.

Inamoto and Saito [2002] also work with material from a soccer game and process the foreground (players) and the background separately. Their approach only interpolates between two views. Therefore, it is not possible to view the scene from arbitrary view points but just along the line between the original views. For the pitch they use homographies and interpolate by blending. To interpolate the views of the players, they find correspondences on the silhouette borders by intersecting it with epipolar lines. Hayashi and Saito [2006] also separate between foreground and background. The players are rendered with billboarding.

Most related to our work and also focusing on outdoor sports is the research by Guillemaut et al. [Guillemaut and Hilton, 2011; Guillemaut et al., 2009; Hilton et al., 2011], which were already mentioned in section 3.2.4. Their methods use about 6 to 12 cameras in outdoor soccer or rugby matches and achieve highly realistic novel view renderings even of sequences. We will later in this thesis directly compare the renderings of our method to their results.

The products of LiberoVision [2012] generate novel views of different sports that continuously fit into original camera shots. With a simple interface and a user-aided process, these scenes can be transformed quickly from the original camera inputs to a 3D representation that allows also to mark, move or remove players, as well as to add artificial drawings. Since this system still relies on manual interaction, it focuses on freeze-frames rather than longer sequences. Our work aims to improve this system by reducing the user interactions and make it possible to render dynamic scenes.

CHAPTER 4

Prerequisites

This chapter describes the entire pipeline of video-based rendering, starting from the input camera images to the final renderings of novel viewpoints. It does not contain own technical contribution, but gives the background on methods used for video-based rendering that is needed for the following chapters. Most notably, this chapter introduces important terms and variables that will be used later on.

The general pipeline of video-based rendering is described in section 4.1. The setup in video-based rendering for sports events and its challenges are explained in section 4.2. Section 4.5 explains common methods for camera calibration, section 4.4 for background subtraction and segmentation, and section 4.3 for feature matching, which are used for generating a virtual representation (section 4.6) and for rendering novel views (section 4.7). In section 4.8 possible methods for evaluation of the results are described. In the end, in section 4.9, the LiberoVision system is described, as one possible implementation and direct application of the entire pipeline.

4.1 Pipeline Overview

Since there are many different ways to achieve the goal of video-based rendering, it is difficult to give a detailed general pipeline. However, most of the

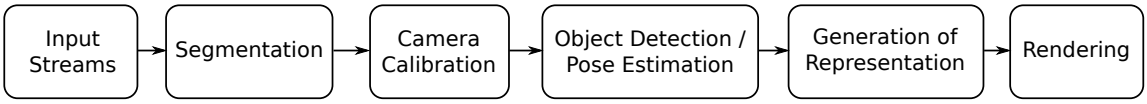


Figure 4.1: *General pipeline of video-based rendering.*

methods as well as both of our own approaches fit into the generic pipeline shown in figure 4.1. Some methods just bypass or combine parts of it. Therefore, this pipeline can be taken as a generic schema which we use to describe the important elements of video-based rendering.

The first item symbolizes the input, which are synchronized video streams, usually only 2 or 3 in our setup. This can be seen as an image per camera and discretized time instant, called frame.

In the second step, the segmentation (section 4.4), different parts of the image are distinguished to separate, e.g., background and foreground or between different players. This assigns to every pixel in the image a meaningful label that tells where this pixel belongs to in the real world.

The camera calibration (section 4.5) is the process of estimating the position, direction and internal parameters of the video cameras.

Following the calibration, often important objects are detected and/or detailed poses of them acquired. Especially when using simple object representations, this information can be used directly in the next step to generate a virtual model of the object. A method for pose estimation of the human body is part of this thesis and described in chapter 5.

In the second last step, a virtual representation, a model, of the scene is constructed (section 4.6). This is a geometric proxy with an approximation of the real worlds geometry as well as color and sometimes also more information about the objects surface.

The rendering (section 4.7) then takes this virtual model and draws it according to the users desired view such that she/he percepts the resulting image as similar as possible to a real world view from the desired viewpoint.

4.2 Sports Broadcast Setups

Similar to other hard- and software, also setups and techniques for video recording and television broadcasting developed a lot since the initial start in the thirties. Two changes are important for video-based rendering. First, most broadcast systems now record the synchronized video streams directly

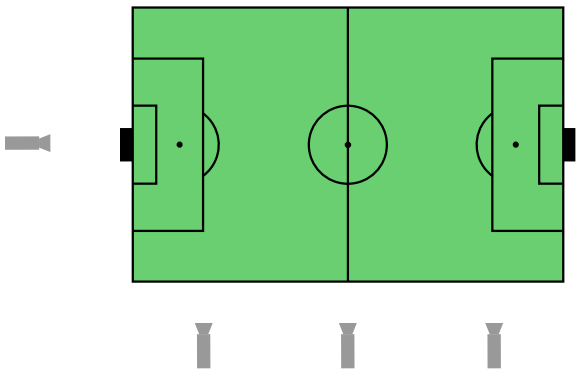


Figure 4.2: *Example for typical camera placement in a soccer stadium. The cameras usually stay at their positions but pan, tilt and zoom are controlled by cameramen.*

in a digital format. This makes it easy to access arbitrary frames directly from the computer using standardized digital protocols. Second, the resolutions recently switched from standard definition (SD) with 720×576 pixels to high definition (HD) with 1920×1080 pixels resolution[1].

In sports broadcasts there are usually several fix placed cameras, which are controlled in rotation and zoom by cameramen. For soccer games, this is usually a setup similar to the one shown in figure 4.2. In addition to this, recently, so called spider cameras (e.g., spidercam^{TM} [Spidercam, 2012] or skycam^{TM} [Skycam, 2012]) were introduced. These remotely controlled cameras mounted on cable-suspended carrier are able to almost freely move over the scene above the players. In popular games, they are installed and used to show additional top views or closeups that follow the scene activity.

In the for this work used video footage, usually only two to three camera images were usable per frame. Some cameras have closeup views that are not able to calibrate and some were focusing on the audience instead of the players.

4.2.1 Challenges

Video-based rendering in such outdoor setups has to cope with a variety of difficulties, that are not present in studio setups. This thesis addresses the development of video-based rendering algorithms to either deal directly with these difficulties or to deal with the errors resulting from methods (e.g.

[1] The resolution values are for the PAL system, mainly used in Europe, Asia, Australia, Africa. The NTSC and SECAM systems have similar values.

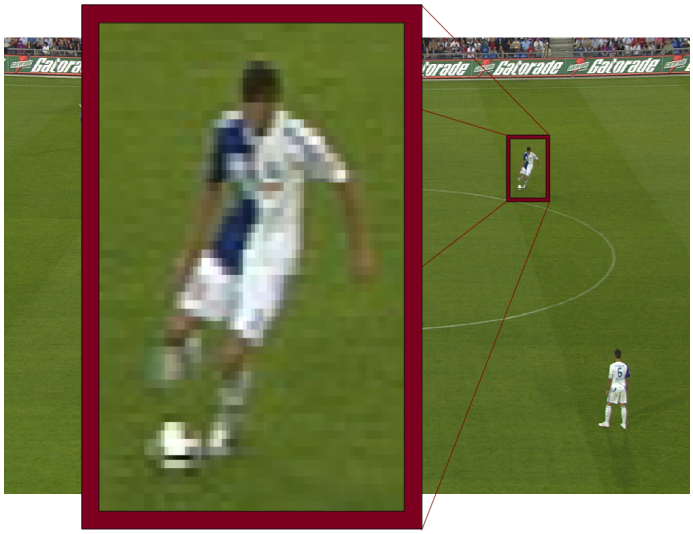


Figure 4.3: *Low resolution of input camera images.*

calibration methods) due to these difficulties. In this section we give an overview over the main challenges.

Low Resolution

To be able to calibrate a camera (see section 4.5), usually a wide area has to be covered. Therefore, closeups are mostly not possible to calibrate at all. Another problem with closeups is that they cover only a small portion of the scene and thus in the reconstruction only add information in this small portion.

On the other hand, cameras with wide area views, even with full HD resolution, have the drawback that the resolution is distributed over a large space. Therefore, the resolution on a small portion, i.e. a player, is low. This results in resolutions of about 50 to 200 pixels for the height of a player [Hilton et al., 2011]. An example is shown in figure 4.3.

Low resolution directly influences the quality of the reconstruction and the rendering of novel views. For the reconstruction, the number of feature points is smaller as well as there are aliasing errors due to the fact that pixels are a simple average over their covered area. For the rendering, despite small improvements through super-resolution methods, the output is limited to the quality of the input.

Motion Blur

Movements of objects relative to a camera or any changes on the camera parameters that change the cameras projection function cause motion blur. The reason for this is that the exposure time (the time where the shutter is open) per video frame is non-zero. The resulting motion blur can be described by a convolution with a point spread function:

$$f = g * p + n \tag{4.1}$$

with g the image without blur, p the point spread function and n the noise. The point spread function depends on the exposure time and on the 2D movement of the projection of a 3D scene point. The movement can have different sources. Examples are if the subject moves or the camera, or if the camera changes the zoom.

In sports broadcasts motion blur mostly occurs due to the fast movements in sports and due to the panning and zooming controlled by cameramen, which causes p to be a non-linear and difficult to estimate function. As opposed to studio setups, p normally can not be measured because we do not have physical access to the cameras nor the scene. Additionally, most camera server systems directly use a compression codec to reduce the size of the stored data. This introduces noise n, also added to the result.

Wide Baselines

As shown in figure 4.2, the cameras in sports broadcasts are usually placed sparsely in the scene, in a wide distance between them. When focusing on an object in the scene, this causes the viewing directions onto this object to be very different between two cameras, and the camera images of the object to differ much. As a result, there will be many parts of the objects surface covered by only one camera. In these parts no feature correspondences between cameras can be computed. Also, the parts covered on more than one camera will have heavy transformations between the camera views and, therefore, the number and the quality of feature matches will be low.

Uncontrolled Lighting

For feature matching and segmentation algorithms, but also for novel-view rendering methods, the lighting in the source images plays an important role. Sports events are usually held in the evening where the sun is low or artificial stadium lights are used. This causes shadows and different

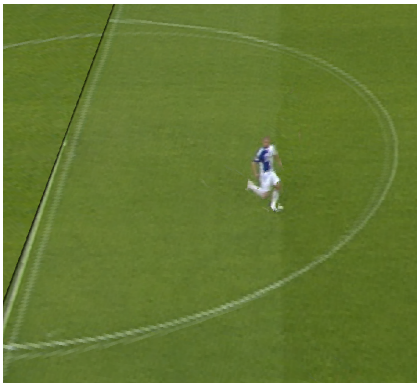


Figure 4.4: *Example with calibration errors visible on the ground as ghosting of lines.*

brightness levels in the used cameras. Also, each camera has its individual white balance. Multi-view segmentation based on color information as well as correspondence matches and background subtraction methods suffer severely from these differences in lighting.

Camera Distortion

Unlike the cameras used for studio setups, TV cameras usually have a radial lens distortion which is difficult to estimate. This is also due to the fact that it is mostly not possible at all to place calibration patterns in the scene. Such distortions directly effect the quality of the calibration. The rendering shown in figure 4.4 gives an example of a calibration that contains errors due to the image distortion of at least one of the cameras. In the figure, the pitch is rendered as a plane with projective textures. The duplication of the lines show the errors in the calibration.

4.3 Feature Detection and Matching

To estimate the camera calibration or the depth of a camera pixel, pixel correspondences are useful. In a multi-camera setup, a pixel correspondence is the knowledge that a given pixel $\boldsymbol{u}_{c_1}$ in camera 1 covers the same point in 3D as pixel $\boldsymbol{u}_{c_2}$ in camera 2.

In order to find pixel correspondences, feature point matching algorithms were developed to compare a pixel and its surrounding area in one camera with a possible match in another camera. These algorithms define a *feature vector* for a given pixel in a camera image, that depends on the pixel itself

and its surrounding pixels. Finally, the task of matching pixels is reduced to comparing these feature vectors.

One of the most popular methods to define feature vectors is the scale-invariant feature transform by Lowe [Lowe, 1999]. The resulting feature vectors are invariant to scaling and rotation, and partially invariant to illumination changes as well as affine or 3D projection. Another robust feature descriptor is the DAISY feature vector [Tola et al., 2008]. We will describe here only the latter one in detail, since DAISY features performed better in tests we did for outdoor sports setups. This is mainly because they are designed for and work well with wide baselines.

4.3.1 DAISY

Let $\boldsymbol{I}$ be the input image given as gray-scale image. To compute the DAISY feature vector, first, histograms are build for every pixel. The histograms describe the distribution of gradients in the area around the pixel by quantizing the angular directions into h bins. Thus, for the entire image, for each quantized direction o_i with $i \in \{1,..,h\}$ an orientation map is computed as follows:

$$\boldsymbol{G}_i = \max\left(\frac{\delta \boldsymbol{I}}{\delta o_i}, 0\right) \tag{4.2}$$

where o_i is the orientation of the derivative. For each pixel, this directly gives the histogram with H bins. From this, the convolved orientation maps are computed as

$$\boldsymbol{G}_i^{\Sigma} = G_{\Sigma} * \boldsymbol{G}_i \tag{4.3}$$

with G_{Σ} a Gaussian kernel used to control the size of the region. They are computed for different Σ.

For a given pixel location (u,v), the DAISY feature vector is defined as

$$\begin{aligned}
\mathcal{D}(u,v) = \quad & [\tilde{\boldsymbol{h}}_{\Sigma_1}(u,v), \\
& \tilde{\boldsymbol{h}}_{\Sigma_1}(\boldsymbol{l}_1(u,v,R_1)),...,\tilde{\boldsymbol{h}}_{\Sigma_1}(\boldsymbol{l}_t(u,v,R_1)), \\
& \tilde{\boldsymbol{h}}_{\Sigma_1}(\boldsymbol{l}_1(u,v,R_2)),...,\tilde{\boldsymbol{h}}_{\Sigma_1}(\boldsymbol{l}_t(u,v,R_2)), \\
& ... \qquad (4.4) \\
& \tilde{\boldsymbol{h}}_{\Sigma_1}(\boldsymbol{l}_1(u,v,R_q)),...,\tilde{\boldsymbol{h}}_{\Sigma_1}(\boldsymbol{l}_t(u,v,R_q))]^T \qquad (4.5)
\end{aligned}$$

where $\tilde{\boldsymbol{h}}$ are the normalized vectors

$$\boldsymbol{h}_{\Sigma}(u,v) = [\boldsymbol{G}_1^{\Sigma}(u,v),...,\boldsymbol{G}_h^{\Sigma}(u,v)]^T \tag{4.6}$$

t is the number of histograms at a single layer (i.e., at the same distance from the center (u,v)) and q the the number of layers around (u,v). In each layer $j \in \{1,...,q\}$ the distance R_j to (u,v) is constant.

As with other descriptors, the DAISY feature vector can be compared to DAISY feature vectors from pixels in other cameras to find correspondences. The comparing is usually done by computing the sum of squared differences of the vectors, i.e. the squared L_2 norm of the difference of the two vectors.

Rotation and Scale Invariance

From the definition above, the DAISY features are not rotationally invariant. However, if the rotation between two images is known, i.e., if calibrations are given, the DAISY features can be easily rotated to match each others orientation. This is done by reordering the columns of the feature vector to shift circularly the h bins of the histograms as well as the t histograms in each layer. To match the rotation, the orientation of the first bin (and histogram) can be selected to match the epipolar line of the pixel in the other image.

Scale invariance is also not addressed by the DAISY descriptor. However, if the scale changes are not too big, then the feature comparison and thus the matching still works well, as the authors showed in their paper [Tola et al., 2010] and also can be seen in our results (see chapter 7).

4.3.2 Detecting Edges: Canny Edge Detection

Canny edge detection, developed by John F. Canny [Canny, 1986], is a simple and fast method to mark edge pixels in an image. These edges can then be used later on for the calibration or other tasks, i.e., the localization of depth discontinuities, that rely on the localization of sharp changes of brightness in an image.

The algorithm first applies a Gaussian smoothing to the image by a convolution with a Gauss kernel. The purpose of this step is to reduce the noise and thus the number of false positives in the edge detection.

After the smoothing, the derivative in horizontal ($\boldsymbol{G}_x$) and vertical ($\boldsymbol{G}_y$) direction of an image $\boldsymbol{I}$ are estimated. A common method to compute this is the Sobel filter, which is the following convolution:

$$\boldsymbol{G}_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \star \boldsymbol{I}, \quad \boldsymbol{G}_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \star \boldsymbol{I} \tag{4.7}$$

where $\star$ is the convolution operator. From this, the angle of the direction of the gradient at a pixel position (u, v) can be estimated as

$$\Theta = \arctan\left(\frac{\boldsymbol{G}_y(u,v)}{\boldsymbol{G}_x(u,v)}\right). \tag{4.8}$$

At a high gradient, usually also neighbor pixels have a relatively high gradient. To select only the maximum out of these, i.e., where the highest change in intensity is, the algorithm tests the gradient values at nearby pixels along the direction of the gradient. The current pixel is only set as a possible edge if it has locally along this line the highest value.

Out of these possible edge pixels, one could simply threshold according to their gradient value and mark them as an edge if the value is high enough. However, usually there is no optimal value that fits for all parts of an image. Therefore, two thresholds are used in a hysteresis approach, where we walk in orthogonal direction to the gradient direction, i.e. along a possible edge, using the direction information from equation (4.8). While marking pixels as edge, we only switch to marking as non-edges if the value is below the lower of the two thresholds. While marking pixels as non-edge, we only switch to marking as edge if the value is above than the higher of the two thresholds. This allows further noise reduction.

Finally, the result is a binary map that assigns to each pixel if it is at an edge or not.

4.3.3 Detecting Lines: Hough Transform

Especially in sports games, important features are the lines on the field. As we will see in section 4.5.2 and in section 4.4, these lines can be used to compute the camera calibration. Detecting lines in an image is therefore an important part of the entire pipeline and we will thus describe here the Hough transform [Hough, 1962] as an example for line detection.

The Hough transform uses an edge detection (e.g., the Canny edge detection, section 4.3.2) as an input. The goal is to turn the result of the edge detection, which is usually noisy and has false positives as well as false negatives in it, into a list of lines denoting collinear points. This is achieved by transforming the 2D result of the edge detection into the 2D Hough space. The Hough space is the discretized space of possible lines with the distance r of the line to the origin as one dimension and the angle θ between the x-axis and the vector along this shortest distance as the other dimension. Each detected edge point in the original image can only contribute to a given subset of all possible lines

and thus only contribute for a given subset of points (bins) in the Hough space. Therefore, the transformation can be done by simply collecting votes for each point of the Hough space. Finally, the lines in the original image can be assumed to be those points of the Hough space that have a high value (many votes).

4.4 Segmentation

A segmentation of an image is a subdivision of the set of all image pixels into subsets, i.e., segments. The pixels of a segment preferably correspond to the same object or color resulting in a meaningful subdivision of the image usually into connected segments. More formally, a segmentation is an injective mapping function $M(x,y)$ that labels each pixel $\boldsymbol{I}(x,y)$ of an image $\boldsymbol{I}$ with one out of the set of n labels $L = \{l_1,...,l_n\}$.

The task of a segmentation algorithm is to find a meaningful M that helps to understand or process an image. In our setup this is mainly to distinguish different players, the pitch or other objects in the scene. Of course, this can be done manually, but especially when dealing with video sequences, this is too time consuming and thus not applicable. Therefore, we focus here on segmentation algorithms that are automatic or semi-automatic. In general, the approach is to compute for each pixel $\boldsymbol{I}(x,y)$ and label l_k, with $k \in L$, a probability $P(M(x,y) = l_k)$ based on the image evidence. Then, the mapping $M(x,y)$ can be defined as always selecting the label with the highest probability. Several methods exist to compute $P(M(x,y) = l_k)$ [Forsyth and Ponce, 2002; Inamoto and Saito, 2007; Guillemaut and Hilton, 2011].

4.4.1 Color Models

Color models allow to compute $P(M(x,y) = l_k)$ for every k according to the color of $\boldsymbol{I}(x,y)$. To achieve this, a color model for every l_k is built. A color model is an approximation to the distribution of the pixel colors in the corresponding pixel type. This directly makes use of the fact that, e.g., green pixels are usually pixels that belong to the pitch and thus to the background.

A color model can be represented by the weighted sum of several Gaussian densities as a *Gaussian Mixture Model* (*GMM*). A GMM with n Gaussians is defined as the probability

$$p(\boldsymbol{x}) = \sum_{q=1}^{n} w_q N(\boldsymbol{x}|\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q) \tag{4.9}$$

where

$$N(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\sigma}) = (2\pi)^{-\frac{D}{2}}|\boldsymbol{\sigma}|^{-\frac{1}{2}}e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{4.10}$$

is the probability density function of a multivariate Gaussian, with D the number of dimensions, i.e., color channels. $p(\boldsymbol{x})$ with $\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix}$ is an approximation for $P(M(x,y) = l_k)$. w_q are the mixture weights for the Gaussians and μ_q and σ_q their means and standard deviations.

To learn a color model, a given labeling $\hat{M}$ is required. The task of estimating a color model is then to find the optimal values for all $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ such that $p(\boldsymbol{x})$ is as close as possible to $P(\hat{M}(x,y) = l_k)$. This is usually done by using the expectation-maximization algorithm [Dempster et al., 1977].

Using this, we can, e.g., build a color model for the background and one for the foreground. The assignment of a pixel is then the pixel type that has a higher probability.

To reduce the errors due to noise, a smoothing step can be applied to M. The idea is to get rid of single pixels or very small or thin groups of pixels that belong to a different segment than the pixels surrounding them. A common method used for this is morphological closing [Serra, 1983].

4.5 Calibration

Even though there exist methods to synthesize novel views purely in 2D, e.g. by image morphing [Beier and Neely, 1992] or by moving gradients [Mahajan et al., 2009], most algorithms to render novel viewpoints use informations on the cameras about their position, rotation, focal length, aspect ratio, and lens distortion. These factors are determined as the calibration of a camera and directly give us a function to project a 3D point into the camera image or project a 2D point of the camera image into 3D space.

4.5.1 Camera Model

A photo camera or a video camera can be approximated by the pinhole camera model, shown in figure 4.5. It defines the camera as a center of projection $\boldsymbol{c}$ and the image as a virtual plane Π, called the *image plane* or *focal plane*. The red line is an example for a light ray that comes from a real object (the cube) and hits $\boldsymbol{c}$. The intersection with Π gives us a projection onto this virtual plane. The same projection but mirrored can be seen on the plane shown to the left of $\boldsymbol{c}$. If this plane would be inside a box that has only a small

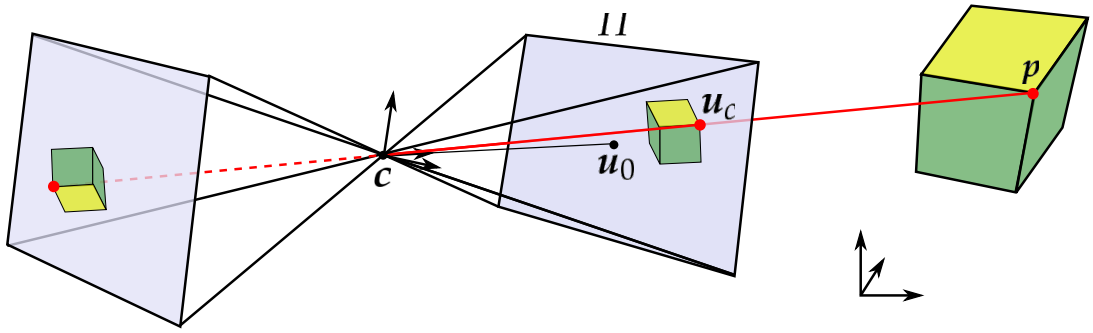


Figure 4.5: *The pinhole camera model. The three vectors at* $\boldsymbol{c}$ *represent the camera space coordinate system, where the z-axis is on the principal axis. The three vectors represent the world coordinate system.*

hole at $\boldsymbol{c}$ then this projection could be seen in real. This is the reason for the name pinhole camera.

The camera space has its origin at $\boldsymbol{c}$ and its z direction orthogonal to Π. The transformation of point $\boldsymbol{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ from world coordinates into camera coordinates $\boldsymbol{u}_c = \begin{pmatrix} u \\ v \end{pmatrix}$ is then given by

$$\underbrace{\begin{pmatrix} u \cdot d \\ v \cdot d \\ d \end{pmatrix}}_{\boldsymbol{u}_c} = \underbrace{\begin{pmatrix} \frac{f}{a_x} & 0 & u_0 & 0 \\ 0 & \frac{f}{a_y} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\boldsymbol{K}} \underbrace{\begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\boldsymbol{M}} \underbrace{\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}}_{\boldsymbol{p}} \tag{4.11}$$

where $\boldsymbol{K}$ the 3×4 intrinsic camera matrix and $\boldsymbol{M}$ is the 4×4 extrinsic camera matrix. The point coordinates are in homogeneous coordinates, where we simply set the last column of the 3D coordinates to 1.

The intrinsic matrix $\boldsymbol{K}$ describes the internal camera properties. f is the focal length, a_x and a_y are the pixel width and height, respectively. $\boldsymbol{u}_0 = (u_0, v_0, 1)^T$ is the principal point, usually the center of the image, which is the intersection of the z axis of the image space with Π. The line that contains $\boldsymbol{u}_0$ and $\boldsymbol{c}$ is called the *principal axis*.

The extrinsic matrix $\boldsymbol{M}$ determines the global position and rotation of the camera. It consists of a 3×3 rotation matrix R in the upper left corner and a translation vector $\boldsymbol{t} = (t_1, t_2, t_3)^T = -\boldsymbol{c}$.

In the remainder of this thesis, we refer to a projection from world space to camera space as the transformation from the coordinates (x, y, z) to (u, v) and

the depth d. The inverse operation, given a depth and the 2D coordinates, can be done by

$$\boldsymbol{x} = (\boldsymbol{KM})^{-1} \boldsymbol{u}_c. \tag{4.12}$$

This also gives us the ray r (red line in figure 4.5) with the origin $\boldsymbol{r}_0 = \boldsymbol{c}$ and the direction $\boldsymbol{r}_d = \boldsymbol{x} - \boldsymbol{c}$, to compute intersections with objects. Further in this thesis, we will denote $P = KM$ and $P^{-1} = KM^{-1}$.

The projection of the ray r into another camera image results in a line, called the *epipolar line*.

The calibration can also be used for *triangulation* (also referred as *lifting*). In triangulation, 2D pixel correspondences are used to find the 3D position of the corresponding surface part. Assuming the 2D positions $\boldsymbol{u}_{c_1}$, $\boldsymbol{u}_{c_2}$, ..., $\boldsymbol{u}_{c_k}$ in cameras c_1, c_2, ..., c_k belong to the same surface position in 3D, but this 3D position is unknown. It can be found by intersecting the rays r_1, ..., r_k projected from $\boldsymbol{u}_{c_1}$, ..., $\boldsymbol{u}_{c_k}$ in 3D. If the camera corrections are not exact, then these rays might not intersect all at one point. Then the closest point to all rays can be taken as an approximation.

A drawback of this camera model is that it does not cover non-linear distortions. However, extensions exist that are also able to describe and estimate, e.g., non-linear lens distortion [El-Melegy and Farag, 2003].

4.5.2 Calibration Estimation

Given one or more camera images, there exist several methods to estimate the camera calibrations, i.e. the 3×4 *camera matrix* $\boldsymbol{P} = \boldsymbol{KM}$ for each camera. These algorithms are also called *resectioning*. We focus here on resectioning methods that are suitable for sports setups.

Using Given Point Correspondences

To estimate the calibration, we can use point correspondences from world space to camera space. These points can be given as input from a user, i.e., by clicking in the camera image onto positions for which the corresponding 3D points are known or also given by the user. In sports games, this can be markings on the field. This gives us $\boldsymbol{p}^{(i)}$ and $\boldsymbol{u}_c^{(i)}$ for a number n of correspondences $i \in \{0, ..., n-1\}$.

A common method to use correspondences for computing $\boldsymbol{P}$ is the *direct linear transformation (DLT)* algorithm [Hartley and Zisserman, 2003]. Each

correspondence i gives us the following equations:

$$\boldsymbol{u}^{(i)} = \boldsymbol{P}\boldsymbol{x}^{(i)}. \tag{4.13}$$

This can be re-written as

$$\boldsymbol{u}^{(i)} \times \boldsymbol{P}\boldsymbol{x}^{(i)} = \mathbf{0}, \tag{4.14}$$

which is

$$\begin{pmatrix} \mathbf{0}^T & -d_i\boldsymbol{x}_i^T & v_i d_i \boldsymbol{x}_i^T \\ d_i\boldsymbol{x}_i^T & \mathbf{0}^T & -u_i d_i \boldsymbol{x}_i^T \\ -v_i d_i \boldsymbol{x}_i^T & u_i d_i \boldsymbol{x}_i^T & \mathbf{0}^T \end{pmatrix} \underbrace{\begin{pmatrix} \boldsymbol{P}^{(1)} \\ \boldsymbol{P}^{(2)} \\ \boldsymbol{P}^{(3)} \end{pmatrix}}_{\boldsymbol{s}} = \mathbf{0}, \tag{4.15}$$

with $\left(\boldsymbol{P}^{(k)}\right)^T$ the k-th row of $\boldsymbol{P}$. We define the 12-vector $\boldsymbol{s}$ as a the stacked transposed rows of $\boldsymbol{P}$.

Because the equations in equation (4.15) are linearly dependent, we can omit the third equation, resulting in

$$\begin{pmatrix} \mathbf{0}^T & -d_i\boldsymbol{x}_i^T & v_i d_i \boldsymbol{x}_i^T \\ d_i\boldsymbol{x}_i^T & \mathbf{0}^T & -u_i d_i \boldsymbol{x}_i^T \end{pmatrix} \boldsymbol{s} = \mathbf{0}, \tag{4.16}$$

If n point correspondences are given, then the equations can be stacked to a system of $2n \times 12$ matrix $\boldsymbol{A}$ and the resulting camera matrix $\boldsymbol{P}$ can be computed by solving

$$\boldsymbol{A}\boldsymbol{s} = \mathbf{0} \tag{4.17}$$

At minimum, $5\frac{1}{2}$ correspondences are used to determine a solution. However, usually the data is not exact due to noise and also the pixel accuracy. Therefore, it is useful to incorporate the information of more point correspondences than the minimum. This leads to an overdetermined solution that can be solved in an iterative approach, by minimizing $||\boldsymbol{A}\boldsymbol{s}||$ subject to the normalization constraint $||\boldsymbol{s}|| = 1$.

Using Lines in the Scene

As mentioned before, in sports setups such point correspondences are usually not given as an input. However, if the position of the field markings in world space are known and if a user inputs their position in the camera images (e.g., by clicking on them), then the calibration can be computed according to the above procedure. Unfortunately, this will involve user interaction and thus

be time consuming. Also, in some sport games (e.g. soccer) not all distances between field markings are defined by the rules. Therefore, the positions can be different for different stadiums.

Field markings are mostly lines and circles in sports games. Even if their position in 3D is not known, they can serve to compute camera calibrations. A real-time method that does this is the one by Thomas [2006] and Thomas [2007]. They make use of the prior knowledge that in sports events, the cameras can pan and zoom but their position remains constant because they are mounted on fixed positions in the stadium. Using this, the calibration estimation can be split into three parts, where the cameras position is estimated only once in the first part of the algorithm as a pre-process. This reduces the amount of computation in the remaining parts of the algorithm.

4.6 Reconstruction

Assuming we have input images from calibrated cameras, the next goal is to create a virtual representation of the scene. This will then be used to render the scene from arbitrary viewpoints. The type of underlying geometric proxy is closely related to the method how it was retrieved. The variety of reconstruction methods is huge and thus we only focus here on approaches that are important for this thesis. Also, methods that are purely image-based and thus do not use a camera calibration [Beier and Neely, 1992; Mahajan et al., 2009] are omitted here. As described in section 4.7, there exists also methods to render novel views without the use of a geometric proxy at all. However, many methods build on a - simple or elaborated - geometric proxy that approximates the 3D shape of the scene.

4.6.1 Visual Hull

The visual hull, introduced by Laurentini [Laurentini, 1994] is a method to reconstruct a geometric proxy based on silhouettes. Assuming we have given the silhouette of, i.e., a human body, in several camera views. Usually they are computed by using a segmentation, most notably background subtraction. Such a silhouette together with the corresponding camera calibration divides the world space into two parts. One part is the set of all 3D points that fall on points inside the silhouette when projecting into the corresponding camera image. The other part are the points which are outside the silhouette. From the definition of a silhouette we know, that the object - and thus also the geometric proxy for the object - lies inside the silhouette. Therefore, the latter

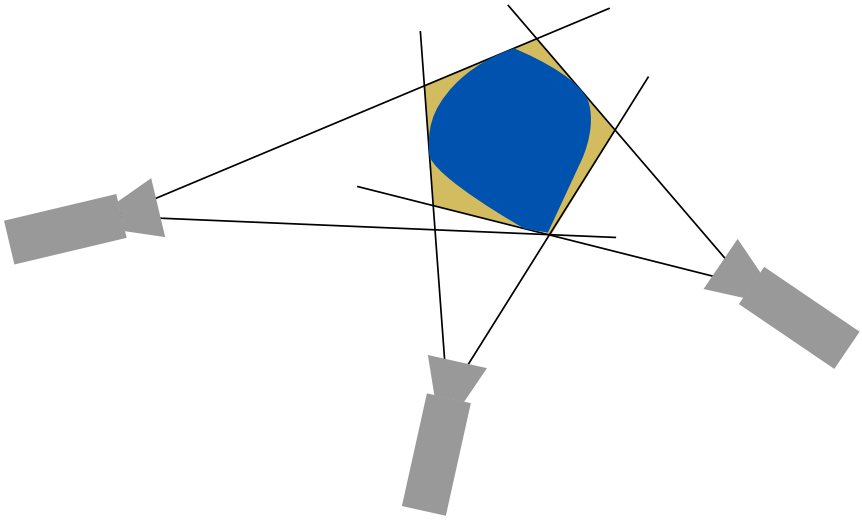


Figure 4.6: *The visual hull. Given the blue object and the silhouettes it produces on the three cameras (indicated by black lines), the visual hull is defined as the maximum volume that projects entirely into the silhouettes in all cameras. This results in a larger volume than the actual object, marked as a brown polygon.*

part of the scene, the points that do not project into the silhouette, can be carved out. Having in the beginning a geometric proxy that contains the entire world space, this step reduces the proxy to the cone of the silhouette of this camera. Repeating this step for all cameras that cover the object, the representation gets finer and finer. The resulting visual hull is the geometric proxy that occupies the maximal space that lies within all silhouette cones. Figure 4.6 illustrates the visual hull.

A disadvantage of the visual hull is that it is not able to reconstruct concavities. Additionally, it requires many cameras, and especially cameras on all sides, to result in an accurate reconstruction.

4.6.2 Stereo

Stereo reconstruction algorithms try to find for every pixel in the camera a depth value that determines how far away the first object is. Together with the camera calibration, the resulting *depth image* or *range image* gives a reconstruction of the scene that covers all points visible from this camera and thus is called a 2.5D reconstruction. Adding more cameras from different viewpoints results in a full 3D reconstruction.

To estimate the depth of a point in a camera, a second camera is used. Assum-

ing the calibrations of camera c_1 and c_2 are given and we want to estimate the depth of a pixel $\boldsymbol{u}_{c_1}$ in c_1. Projecting $\boldsymbol{u}_{c_1}$ into world space gives us the epipolar line $\boldsymbol{v} + t\boldsymbol{d}$ in c_2. If the calibrations are correct, then the corresponding pixel $\boldsymbol{u}_{c_2}$ in c_2 must be on this epipolar line. This *epipolar constraint* reduces the task of finding $\boldsymbol{u}_{c_2}$ to finding the parameter t of the epipolar line, and thus to a line search. The search for an optimal t is what a stereo algorithm does. If it is found, the 3D position of the corresponding pixel can be computed by triangulation of these found corresponding 2D points. The depth in camera c_1 is then simply the distance from $\boldsymbol{c}_1$ to this 3D point.

To find these pixel correspondences, i.e., to find t, there exists several methods. As a first step, the search space can be reduced by a minimum and a maximum expected depth according to knowledge about the scene. From this the search on the remaining positions can be done by using feature matches. For pixels without feature matches the depth can be interpolated. Another method is to use a search window around the pixel and compare this. However, this does not consider distortions and differences in lighting due to the different viewpoints.

For other methods on stereo, we recommend the survey by Scharstein et al. [Scharstein and Szeliski, 2002] and for a list and comparison of more recent approaches, the Middlebury website on stereo [MiddleburyS, 2012] and multi-view stereo [MiddleburyMVS, 2012].

4.6.3 Billboarding

Billboarding approximates the objects by simple planes - one per camera. Compared to dense stereo (depth per pixel), this is easier to compute because a plane has only 4 parameters to determine. Also the rendering is easier and thus faster when having only a few simple geometric proxies.

Assuming we have given several views of an object from calibrated cameras. For each camera that covers this object a plane is placed in 3D. The plane is set orthogonal to the camera's view direction. This leaves open only one parameter to define, the depth of the plane, i.e., the distance to the camera. If we can define an estimation of the center of mass of this object, then this can be used. A simple approach would be to triangulate in 3D the center of mass of the 2D silhouettes. The silhouettes are also used as a mask on the plane, describing which points on the plane can be omitted in the rendering because they are outside the silhouette. This cut out billboards or masked billboards are sometimes referred to as *sprites*.

Waschbüsch et al. [Waschbüsch et al., 2007] improved the representation of

billboards by adding per-pixel depth values to the surface of the planes, thus adding a displacement orthogonal to the plane surface. The depth can be acquired using stereo methods.

4.7 Novel-View Synthesis

Having many different methods to construct a 3D representation of a scene, there also exists many methods to render them. We will describe here the general methods of light field rendering (section 4.7.1) as well as view dependent texture mapping (section 4.7.2), which both are combined as unstructured lumigraph rendering (section 4.7.3). Finally, we will also describe billboard rendering in more detail (section 4.7.4), since a part of this thesis builds on top of this.

4.7.1 Light Field

The light field [Levoy and Hanrahan, 1996] (and very similar the lumigraph [Gortler et al., 1996]) does require a geometric proxy at all. It represents the scene as the light rays between two planes. If a parametrization on these planes is given, then a ray can be represented by the two points where it hits the two planes, (u,v) in plane 1 and (s,t) in plane 2. The advantage of this is that no geometric proxy is used at all and, therefore, no geometry has to be computed at all. A light field can be acquired by an array of cameras in the first plane. Each pixel of each camera corresponds to one light ray $L(u,v,s,t)$.

The rendering of a view in a light field is done by computing for every target pixel the corresponding ray $L(u,v,s,t)$ in the light field. From this ray, the color value can be used directly. If there was no ray acquired that exactly matches the target pixel, the color value of surrounding rays can be interpolated.

4.7.2 View-Dependent Texture Mapping

Debevec et al [Debevec et al., 1996] introduced view-dependent texture mapping as an image-based rendering method. It uses a coarse geometric proxy and the camera images to render novel views. Their main target was on architecture, but the principle of view-dependent texture mapping can be used for any geometric proxy.

Assuming that the surfaces of an object exhibits only Lambertian reflection, then the color values of the model can be computed by simply projecting the camera images onto the surface. This can be thought of as replacing the cameras with projectors and is thus also referred as *projective texturing*. At points where there are more than one source cameras to project from, the color values need to be averaged or weighted. Debevec et al. recommend to use weights that are for each camera inversely proportional to the magnitude of the angle between the rendering view and this camera. Additionally, at borders between points of different numbers of source camera coverage, these weights are smoothed such that no sudden changes are visible.

However, if the model is not correct, then this method still causes errors. Debevec et al. suggested a method based on stereo to locally adjust the surface to reduce the error. More recently, Eisemann et al. introduced *floating textures* [Eisemann et al., 2008] as another solution to reduce the error.

4.7.3 Unstructured Ligh Field / Lumigraph

The main restriction of the light field and lumigraph approach is the assumption that the cameras are on a plane and also in a structured order. An extension to also allow an unstructured light field or lumigraph was presented by Heigl et al. [1999]. Their approach uses dense but arbitrary camera positions, e.g., a hand-held camera, as input to compute novel views. In a first step, calibrations as well as depth maps are computed for every camera image using close other camera views. These depth maps are then used to reconstruct in realtime a view-dependent geometry for a novel view. The geometry adapts to the depth maps of the cameras, whereas in the original light field the geometry is just the focal plane. Nevertheless this method breaks the restriction of a structured camera setup and reduces the ghosting artifacts, it still requires a dense camera placement.

The unstructured lumigraph [Buehler et al., 2001] generalizes light field / lumigraph rendering and view-dependent texture mapping into one framework. Similar to the method by Heigl et al. [1999], their system is not bound to a structured camera placement. According to the angles to the camera as well as by incorporating occlusions, they weight the light rays to blend. Additionally, also a resolution penalty is taken into account, that makes sure better resolution sources are preferred.

Unstructured lumigraph rendering smoothly changes from light field / lumigraph rendering, if many cameras are available, to view-dependent texture mapping, if only a few cameras are available and a geometric proxy is present.

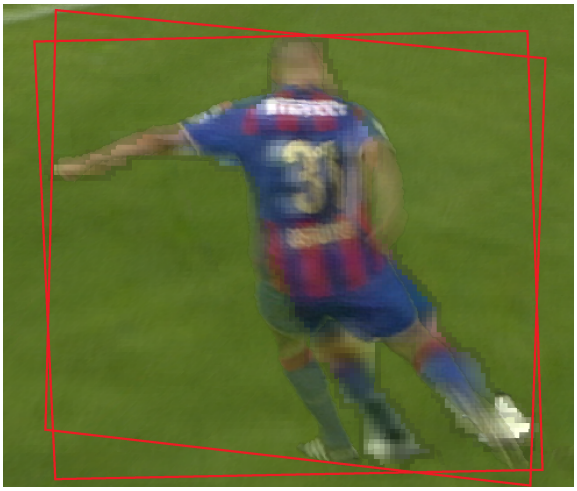


Figure 4.7: *A novel viewpoint rendered with billboards. Because of the non-planarity of the body, ghosting artifacts appear. An example is the duplication of the left leg.*

However, the unstructured lumigraph also suffers from the problems that these two methods have. Either a good geometry is required or enough images have to be available which are well calibrated. In outdoor sports scenes neither of them is usually available.

4.7.4 Billboard Blending

Billboards are rendered similar to standard view-dependent texture mapping. The only difference is that the billboards belonging to the same object are blended together as if they would be on the same surface. For this blending, the same weights can be used as in the unstructured lumigraph. To get the border of the geometry, i.e., a mask that defines which parts of the texture is part of the geometry of a billboard, silhouette masks are used. These masked billboards are sometimes also referred to as *sprites*.

As with any geometry approximation, also billboards introduce artifacts when their geometry differs from the real geometry. They are called *ghosting* artifacts and are object parts that are visible more than once in the rendered novel view. Since billboards are simple planes, ghostings appear already as soon as the real object is not planar or mostly planar. Figure 4.7 shows an example where the legs of a person are not on the same plane that approximates the torso and the arms. Therefore, the legs are shown twice in the resulting rendering.

The advantage of billboards is that they are very robust to calibration errors.

Especially in outdoor sports setups, where the low resolution and the calibrations can lead to entire arms or legs being cut of when using more elaborated methods like the visual hull, billboards still cover the entire body and the resulting increased ghosting is less disturbing.

4.8 Evaluation Methods

Evaluation methods are important to judge the quality and accuracy of the result of an algorithm. In video-based rendering, especially based on TV footage, an evaluation is difficult. This is due to several reasons. The most important problem is that there is no ground truth 3D data available to which we could compare the resulting representation or rendering. Another problem is also the difficulty to locate an error. Even if there would be a method for comparing the output, the question of the source of the error is usually difficult to answer if working with methods that mix geometry and blending. Pixel errors can be caused by wrong calibrations, by wrong segmentation, by wrong blending or by other sources.

Nevertheless, over the years several methods to evaluate the results were developed. The most important ones are listed in the remainder of this section.

Since the goal is to render images or videos that will be watched by humans and should be plausible for the human visual system, a straightforward approach is to use this also for evaluation. A direct evaluation of an error value is very difficult since it would have to approximate or simulate human perception. However, in a user study relative comparisons can be tested. This allows to evaluate if a new algorithm was ranked significantly higher than a given state of the art algorithm. User studies are not an accurate measurement of the correctness of a method. Even if a result is theoretically more correct than another one, this could still be ranked lower by users because of, e.g., non-smooth artifacts that are minor physical changes but more disturbing for the human eye than others. This can be seen as a drawback but in certain setups, where the actual goal is to please the human eye, this might be even more helpful than other evaluation methods.

Another evaluation method are leave-one-out comparisons. Assuming n usable cameras, only $n-1$ of them are used to reconstruct and render a scene. The evaluation is done by comparing the left out camera image with a rendering from exactly this view. The comparison could be done automatically by evaluating pixel difference. However, this assumes that the calibrations are exact, which is usually not the case. Nevertheless, in showing to a user

such renderings, one might see directly the quality of the result and thus this is an often used method to show and compare results.

Some of the rendering methods to generate novel views described above use view-dependent texture mapping. Having more than one color projections for one point in 3D directly allows to compare these colors to test the quality of the geometry and the calibrations. This can be evaluated in a novel view per pixel or faster when rendering from the view of one of the camera images. Thus, the scene is projected back into a given camera c_0 using the color values from all available cameras. In this camera view, the color values from the different sources are compared and the errors (e.g., sum of squared differences) can be evaluated. In the following parts of the thesis, this error will be referred as *back-projection error*.

Instead of working with real world input data, the camera footage could come from renderings of a synthetic model. In this case the calibrations are known, or even their error could be simulated and would be known too. However, the main difficulty would be to simulate the noise, the camera distortions, the lighting changes, the white balance errors, the motion blur and many other effects realistically to be accepted as equal to real world input.

4.9 LiberoVision

The company LiberoVision AG [LiberoVision, 2012] was founded in 2006 as a spin-off from the Computer Graphics Laboratory at ETH Zurich. They currently provide three products, all of them aiming on realistic replays of sports games based on only standard TV cameras:

- **LiberoVision Highlight** allows to re-render a sports scene from novel viewpoints as well as adding drawings virtually onto the pitch. Players can be highlighted, moved around or even removed. This tool helps the sports experts at TV stations to show or directly analyze a game for the watchers. The computation is semi-automatic and takes about 5 minutes to process a frame to analyze.
- **LiberoVision Playbook** is a faster version of LiberoVision Highlight that only takes about 30-60 seconds to process a frame. This can be used for direct replays whereas LiberoVision Highlight can be used for high quality renderings in the game breaks or after the game.
- **LiberoVision Offside** targets on offside or near offside situations and allows to show the offside line as well as to render the scene from the

linesman's view. It has a slimmer design than LiberoVision Highlight and thus has a processing time of only a few seconds.

All of these products do not require any additional technology in the sports scene. Therefore, they can be applied to footage from satellite streams or directly at the stadium. Currently, LiberoVision has customers around the globe, including broadcasters like ESPN, NBC, ZDF and many others.

This thesis was done in a CTI collaboration project between ETH Zurich and LiberoVision. This project targets on improving the current technology and developing new methods that can be used in future versions or products. The following three chapters present the three main parts of the project.

CHAPTER 5

Body Pose Estimation

In the pipeline overview in chapter 4 it was already mentioned, that if the pose of a human body is known, this can be used to improve the virtual representation and thus the rendering result of novel-view synthesis. An example for the benefit of knowing the body pose are articulated billboards, which we will see in chapter 6. Manually annotating human body poses is cumbersome and practically not applicable, especially for video sequences. Therefore, semi-automatic or automatic body pose estimation methods are needed.

As mentioned in chapter 3, many methods already exist. However, most of them are designed for studio setups with a controlled environment [Carranza et al., 2003; Balan et al., 2007; de Aguiar et al., 2008] or use higher resolutions [Mori and Malik, 2002; Mori et al., 2004; Ramanan et al., 2005; Felzenszwalb and Huttenlocher, 2005; Agarwal and Triggs, 2006]. They are not applicable to our target setup. However, Efros et al. [Efros et al., 2003] showed that even in very low resolution outdoor setups, data-driven comparisons can be made to detect actions and rough positions of torso, arms and legs.

In this chapter we present a method for full body pose estimation. It targets on and works in uncontrolled outdoor setups with low resolutions, since it only relies on coarse silhouettes and coarse calibrations. It implies multiple cameras, but can work with already two of them.

Instead of learning a statistical model for the kinematics of the human skeleton, our algorithm directly uses a database of poses. This has two advantages. Firstly, such a data-driven method allows to easily add new pose sequences to adapt to new setups or previously unknown poses. Secondly, there is less statistical bias to more common poses, since the method simply searches for the closest pose in the database. Using a database with anthropometrically correct data will always favor anthropometrically correct poses and thus result in a plausible pose guess for the initial estimation.

The main contributions of our method are:

- A time consistent silhouette based database pose look-up providing an initial pose estimation
- A camera shift computation to correct for calibration errors
- Local and global consistency check to improve the initial pose estimation
- A space-time pose optimization based on novel constraints

The work described in this chapter was presented in [Germann et al., 2011], some parts already in [Germann et al., 2010].

We will first give the problem description of body pose estimation (section 5.1). Then, after an overview (section 5.2), the detailed steps of the algorithm are explained (sections 5.3, 5.3.5, and 5.4). Finally, we will show some results (section 5.6) followed by a discussion and an outlook for future improvements (section 5.7).

5.1 Problem Description

In our setup we assume given camera calibrations as well as the camera images. The calibrations can be done according to section 4.5. Since this method targets on outdoor setups, the calibrations are not assumed to be very accurate.

The problem of body pose estimation is then to find for every subject the positions $\boldsymbol{j}_i$ in 3D of the entire set J of skeleton points. This set can be defined as detailed as needed. For simplicity, we refer to all of these points as *joints*, even though they do not have to be at the position of a real joint. From 3D it is always possible to project the joint positions back into the 2D camera views and thus get the pose there too.

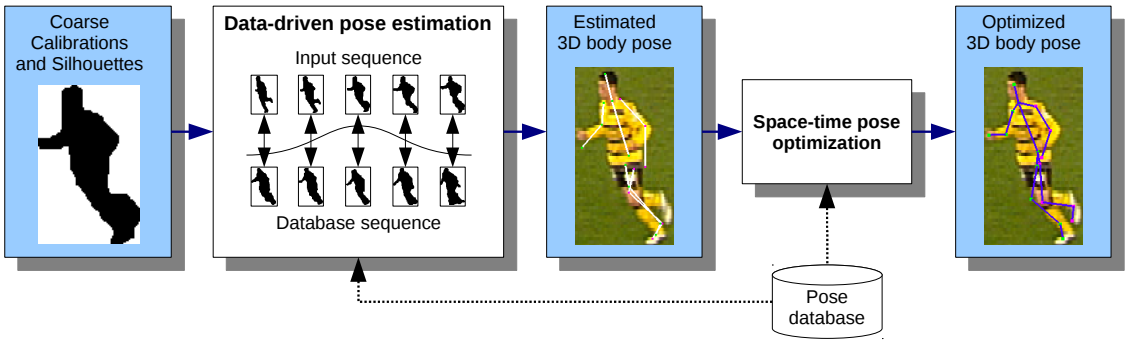


Figure 5.1: *Algorithm overview.*

5.2 Algorithm Overview

Our algorithm for body pose estimation consists of two steps, illustrated with the two white rectangles in figure 5.1.

In the first step, an initial pose guess is found for all frames. For this, the algorithm first extracts 2D poses for each individual camera view using a spatial-temporal silhouette matching technique. The optimal match of such 2D guesses is computed by triangulation, leading to a 3D pose guess.

This pose detection is inherently prone to ambiguities, namely left right flips of symmetrical parts. Although the skeleton matches the silhouettes quite well, the arms or legs of the player can still be flipped. Due to occlusions and low resolution, these ambiguities are sometimes very difficult to spot even for the human eye. Therefore, we employ an optical flow based technique to detect the cases where flips occur, and correct them to obtain a consistent sequence. It is important to note that optical flow is in such setups not reliable enough for tracking the entire motion of a players body parts over an entire sequence, but it can be used for local comparisons as shown by Efros et al. [Efros et al., 2003].

However, in general, no pose from the database will match the actual pose exactly. As a consequence, in the second step of the algorithm, this initial 3D pose is refined by an optimization procedure, which is based on spatio-temporal constraints. The resulting optimized 3D skeleton optimally matches the silhouettes from all views and features temporal consistency over consecutive frames.

The rest of this chapter is organized as follows: Section 5.3 describes the first step of the algorithm in detail and Section 5.4 the second step. After a few details about the implementation in section 5.5, section 5.6 shows results

Figure 5.2: *The used skeleton with* 17 *joints and* 16 *bones.*

from our method and discusses them. Finally, we conclude this chapter in Section 5.7.

5.3 Initial Pose Estimation

The initial pose estimation per subject is divided into two sub-steps. First, in a sliding window approach over all frames the m most similar short sequences of silhouettes are found for each camera view and frame. This results in m 2D pose guesses per camera and frame. Out of these pose guesses, the best ones are found by 3D triangulation and optimizing for a camera shift, which also results in a 3D pose guess. Second, the poses are checked for consistency, i.e., arms or legs are left/right flipped if required, based on an optical flow approach.

In this section, we first describe the skeleton representation and the setup of the database as well as what we assume as input. After this, the two sub-steps, the data-driven pose guess and the consistency check, are explained in detail.

5.3.1 Pose Representation

We use a skeleton with 17 joints connected by 16 bones to represent a 3D pose S, as shown in figure 5.2. To allow for size independent comparisons with other poses and also to reduce the number of variables to a minimum, the skeleton is stored in angle space. Every bone i is represented relative to its parent bone using two angles α_i and β_i as well as the length l_i of the bone. The root bone is defined by its orientation given by three angles α_0, β_0, γ_0 and by a global position $\boldsymbol{j}_0$ as well as its length l_0.

Whereas all angles as well as the root positions are stored per frame, the bone lengths l_i are global over all frames. The joint positions j_i in the 3D Euclidian space can easily be computed from this representation and vice-versa.

5.3.2 Pose Database Construction

Since several parts of our algorithm are data-driven, a database of 2D and corresponding 3D body poses is required. Even though we do not need all possible human motions to be in the database, at least the different kinds of motions should be roughly represented in the database. As an example, if the input data will contain only standing and walking, then we do not need datasets of jumping in the database, but at least some sequences of a person walking, turning, starting to walk, stopping and standing.

However, since we need not only the silhouettes but also the correct 2D and corresponding 3D body pose in the database, it is very time consuming to create a large database manually. Therefore, we use the CMU motion capture database [CMU, 2012] to create an initial set of poses. To also compute the silhouettes, a template mesh rigged with the same skeleton is deformed using linear blend skinning to match the 3D pose. From this, virtual snapshots are taken where the silhouette is extracted and the 2D body pose computed. With this method we created an initial database of around 20000 silhouettes and corresponding poses.

To adapt for new setups, this database has to be extended, since the CMU database has only a limited number of types of poses, mostly from running and walking sequences. As an example, we have from there only one sequence where a person is kicking a ball. Running the pose estimation on soccer scenes with this database, results in weaker estimations for poses that are very different from any of the database poses, but others, e.g., walking will be detected well. Therefore, we manually annotated such failure cases in several soccer scenes and added them to the database. This is done in a bootstrapping process, where each added body pose directly improves the estimation of the next one. With this method we increased the size of the database by 900 silhouettes. This is significantly fewer than the ones generated automatically, but enough to enlarge the span of example poses to obtain good results. It is important to note, that the added example poses were not taken from the same sequences as we used to fit the poses. In figure 5.3 some of these manually added poses can be seen. To increase the quality of the initial pose estimation, the database can be enlarged by adding more manually corrected poses.

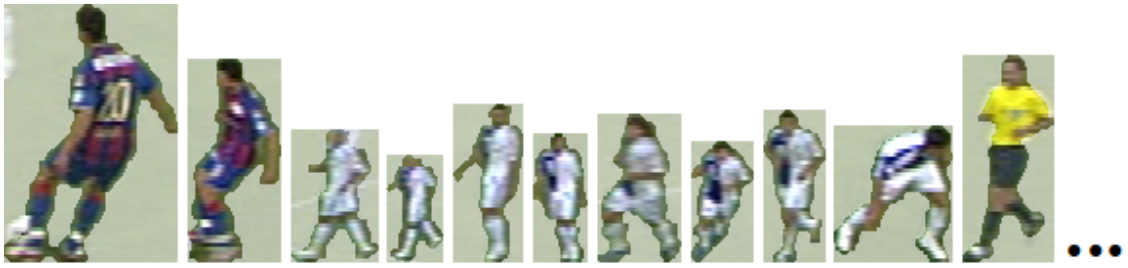


Figure 5.3: *Database poses of real scenes.*

5.3.3 Silhouette Extraction

Besides the camera calibrations, the space-time 2D pose estimation also requires coarse silhouette masks. They can be extracted by using background subtraction (see section 4.4). These silhouettes are usually not very accurate, due to the low resolutions and the outdoor camera setup. However, since our method compensates for the inaccurate camera calibration individually per player and also because the silhouettes are compared on a low resolution, this is still enough for good results in pose estimation.

5.3.4 Space-time 2D Pose Estimation

In a first step of the initial pose estimation, the most plausible 2D pose is found in each camera view and frame by comparing the silhouette to the silhouettes in the database. However, instead of looking at each of them separately, the estimation takes also into account the information from nearby frames and from other cameras.

For each frame, we compare a sequence of n subsequent silhouettes from the input to sequences in the database and compute the following error value.

$$E'_q(s) = \sum_{i \in \{-\frac{n}{2}, \dots, \frac{n}{2}\}} \theta(i) \frac{1}{|P|} \sum_{p \in P} |v_{t+i}(p) - v'_{q,s+i}(p)| \tag{5.1}$$

v_t is the silhouette of the currently processed frame at time t and $v'_{q,s}$ is a silhouette from the database from sequence number q at time s. Both of them contain the binary values of the silhouette at fixed raster positions (grid with height = 40 and width = 32) stacked into a 1D vector. The use of a raster on the cropped silhouette images allows to compare silhouettes of different sizes and also different aspect ratios. An example is shown in figure 5.4.

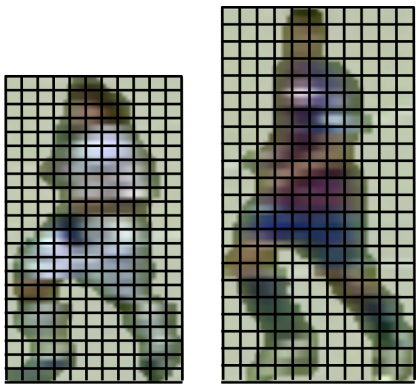


Figure 5.4: *The silhouette comparison is done on a raster fit to the cropped image. This allows comparisons between different sizes and aspect ratios.*

For normalization, P is the set of all raster positions where the corresponding pixel is in both images not possibly occluded, i.e., part of another players silhouette. The marking of such possible occlusions is done automatically in a pre-processing step, where silhouette pixels are labeled if they are in more than one players bounding box.

The weights $\theta(i)$ are based on a simple approximation to a Gaussian function. It is $\theta(0) = 1$ and $\theta(i) = \frac{1}{4^{|i|}}, \forall i \neq 0$. Each weight is normalized by a division through the sum of all weights.

There are comparison windows where some of the silhouettes are not available. Either they are missing in the input sequence or in the database sequence or in both. This, e.g., happens usually at the beginning or at the end of a sequence. In such cases the weight $\theta(i)$ of the corresponding window positions is set to 0. This is done before the normalization such that this positions are not taken into account at all.

To penalize too large differences of the aspect ratios of the two silhouettes to compare, an error term for aspect ratio error is computed:

$$R_q(s) = \sum_{i \in \{-\frac{n}{2}, .., \frac{n}{2}\}} \theta(i) a_q(s+i) \tag{5.2}$$

with

$$a_q(s) = \begin{cases} 1 - \frac{wh_{q,s}}{w_{q,s}h} & \text{if } wh_{q,s} < w_{q,s}h, \\ 1 - \frac{w_{q,s}h}{wh_{q,s}} & \text{otherwise} \end{cases} \tag{5.3}$$

with w, h, $w_{q,s}$ and $h_{q,s}$ the pixel width and height of the input and the reference silhouette, respectively. This is added to the comparison error to

result in the final error:

$$E_q(s) = E'_q(s) + \lambda_R R_q(s) \tag{5.4}$$

The constant λ_R was empirically determined to 0.1, which was used for all our experiments.

Comparing sequences ($n > 1$) instead of single images ($n = 1$), does not only add temporal coherence resulting in smoother motions, but also improves the pose estimation. Even image parts occluded over few frames can be fitted more robustly. Additionally, this approach helps to prevent matching a silhouette that is similar but originated from a completely different pose. This is depicted in figure 5.5.

The choice of an optimal window size n is made in a trade-off. For a small n, the computational cost is lower, due to less comparisons that have to be made. Additionally, the database can be smaller since it is easier to find a match for a silhouette than for a sequence where more variety in the database is required. On the other hand, for a large n, the quality of the results increases as mentioned above. We empirically determined $n = 5$ as optimal and used this for all of our experiments.

To reduce the computational costs, an early abort is performed if in a comparison window the silhouette in the center (the currently processed frame) has a too large difference when compared against the corresponding position in the database. In such cases it is not necessary to compare also the other window positions and the algorithm continues with the next window.

Using the final energy function from equation (5.4), we search for each camera view for the best $m = 2$ pose hypotheses from the database. In 2D silhouettes, symmetrical parts can usually not be correctly identified. Figure 5.7(a) shows an example with two different labeling options for the legs. Therefore, we allow left/right flips of arms and legs in this step. This increases the number of pose hypotheses from two to eight, due to the four possibilities to label arms and legs as left or right.

Triangulation

For each camera, out of these eight pose hypotheses, the optimal one is selected by using a multi-view evaluation that directly also computes an initial camera shift as follows.

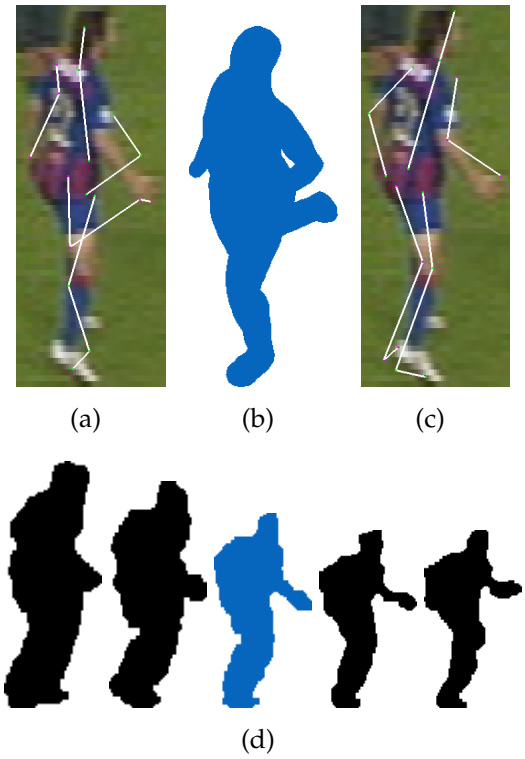


Figure 5.5: *(a) Estimated 2D pose by comparing just the current frame to the database. (b) The found database item for the single frame comparison. (c) Estimated 2D pose by comparing sequences of silhouettes. (d) The found database sequence with the corresponding pose in the middle.*

For each possible combination of pose hypotheses from all available cameras, a triangulation error

$$E_t = \frac{1}{|J||C|} \sum_{c \in C} \min_{\mathbf{s}_c} \sum_{\mathbf{j}_i \in J} dist(P_c^{-1}(\mathbf{j}_{i,c} + \mathbf{s}_c), \mathbf{j}_i) \tag{5.5}$$

is computed. This is a sum of distances of the rays from the cameras to the optimal joint positions as shown in figure 5.6. J is the set of joints in 3D that are closest to the corresponding projected rays from all cameras, in least squares sense:

$$\mathbf{j}_i = \arg\min_{\mathbf{j}_i} \sum_{c \in C} dist(P_c^{-1}(\mathbf{j}_{i,c} + \mathbf{s}_c), \mathbf{j}_i) \tag{5.6}$$

J is the set of joint positions in 3D of the 17 joints. $\mathbf{j}_{i,c}$ is the 2D joint position in camera c of joint number i. The camera shift $\mathbf{s}_c$ is a 2D vector set per camera and subject. It corrects for calibration errors per subject and camera. Since it

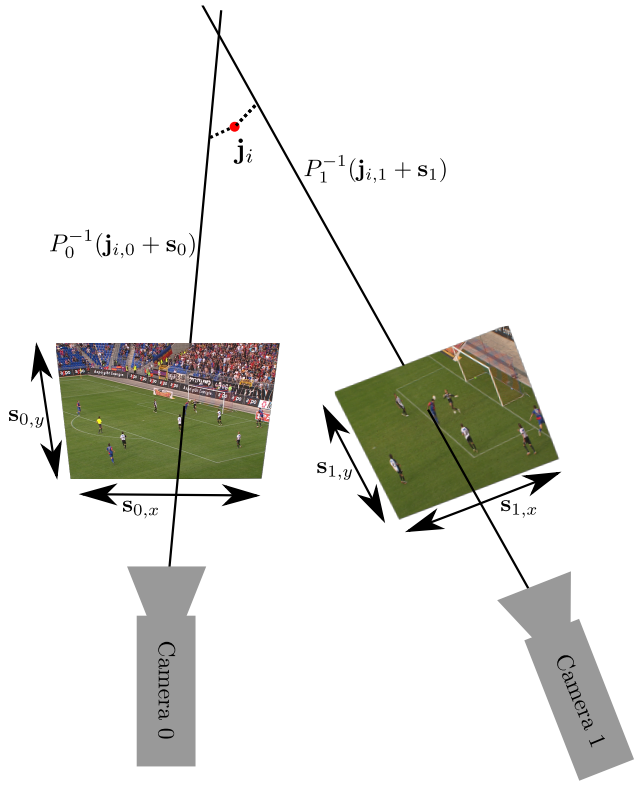


Figure 5.6: *Triangulation of a joint position. $\mathbf{j}_i$ is set to the point closest to the all rays from the corresponding 2D joint positions.*

is not set per joint, it does not adapt to per joint errors but to local calibration errors for the entire human body. The projection $P_c^{-1}()$ transforms such a 2D point from camera space of camera c into a ray in world space while $dist()$ describes the distance of a ray to a 3D point.

From equation (5.5) we directly get the optimal joint positions in 3D as well as an optimal camera shift per camera and subject.

5.3.5 Pose Consistency

The body pose guess relies on silhouette matching and, therefore, is prone to ambiguities. As explained in Section 5.3.4, given a silhouette and an initial 2D pose for it taken from the database, we can not decide if the labellings of left/right at arms and legs are correct.

This directly results in ambiguities in 3D, when working only with two cameras or when there are occlusions in all but two cameras. A detailed schematic example of this is depicted in figure 5.7. Figure 5.7(a) shows two

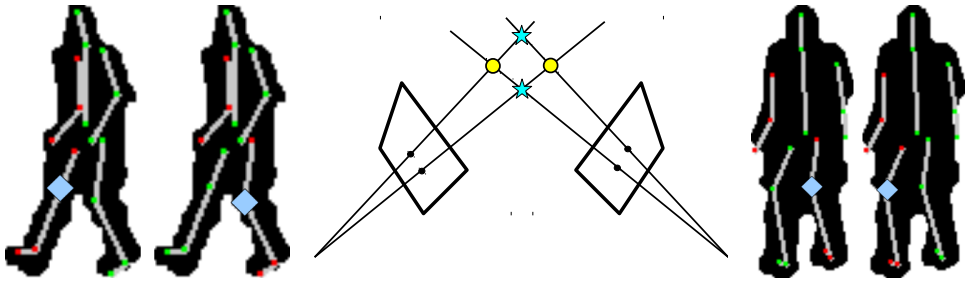


Figure 5.7: *Example for pose ambiguities. (a) Possible labellings in first (left) camera. (b) Schematic view from the top to illustrate the two possible positions of the knees. (c) Possible labellings in the second (right) camera.*

possible labelings of the subjects silhouette in the first camera. The possible position of the right knee is marked by a blue diamond. This view is from the left camera in the schema in figure 5.7(b). The same subject in the same frame but in the second camera shows the silhouette in figure 5.7(c), again with the two possible labellings of the legs. Therefore we have four possible positions in 3D for the right knee after lifting into 3D. They are shown in figure 5.7(b). If the right knee falls on one of the positions marked with a star, the left knee will fall on the other star. If the right knee falls on one of the positions marked with a circle, then the left knee will fall onto the other circle. Let the circles be the correct positions of the knees, then we can have two different failures: either the knees are just wrongly labeled in 3D but at the correct positions or the knees are at wrong positions (the stars).

From the silhouettes and the triangulation only, we cannot decide in such a situation which positions are correct, especially when only two cameras are available. A possible approach to disambiguate the flipped cases would consist of checking all possible combinations and keep the anatomically possible ones. However, it is possible that several configurations of flips yield anatomically correct poses.

To solve these ambiguities, we use the fact that the poses should be consistent over time but still be plausible poses. I.e., the left and the right leg should not flip from one frame to another, but over all frames, the poses should not be unnatural. Thus, we propose a two step approach: first, the local consistency between each pair of consecutive 2D frames is resolved, resulting in an entire sequence of temporally consistent 2D poses. We calle this *local consistency*. Second, the entire sequence is resolved globally, which we call *global consistency*.

Local Consistency

In this first step, we are not looking for the correct poses but try to make the poses consistent over time. The goal of this step is to make sure that the 2D poses recovered from a camera at the previous frame $k-1$ (figure 5.8(a)) and at the current frame k (figure 5.8(c) or 5.8(d)) are consistent, i.e., there are no flips of arms or legs between consecutive frames.

In other words, if a point on the human body at frame $k-1$ and its corresponding pixel $\boldsymbol{u}_{k-1}$ in a camera view belongs to the right leg, this point and its corresponding pixel $\boldsymbol{u}_k$ in this camera in frame k should belong to the same leg as well. To assure this assumption, we assign to each pixel in both color images $\boldsymbol{I}_{c_{k-1}}$ and $\boldsymbol{I}_{c_k}$ a corresponding bone, and we compute the optical flow [Bouguet, 2000] (figure 5.8(b)) between the frames.

The reason for this is, that the optical flow gives us the correspondence of the pixels in the two consecutive images. Thus, a pixel in frame $k-1$ and its corresponding pixel in frame k, computed using optical flow, should be assigned to the same bone. Otherwise there would be a flip as shown in figure 5.8(c).

Therefore, we compute this association for all combinations of possible labellings in frame k, compute the consistency of the pixels, and select the most consistent label configuration for the second frame. This is done for all frames k sequentially.

To make the approach more robust in respect to optical flow errors, we only consider pixels with good optical flow, i.e., when there is a high confidence value for this pixel in the computation of the optical flow. Also, only those pixels are used where the corresponding pixel labels in both frames are of the same bone type, i.e., either both arms or both legs. For instance, if a pixel $\boldsymbol{u}$ belongs to the left arm in frame $k-1$ and to the torso in frame k, it is most likely due to an inaccurate optical flow based on occlusion and we can thus omit this pixel. If the pixel belongs to a different bone of the same type it is a strong indication of a flip. We employ a voting strategy to select the optimal flip configuration. If the number of pixels that have their corresponding pixel on the same side of the body (the green pixels in figures 5.8(f) and 5.8(g)) is larger than the number of pixels with their corresponding pixel on the flipped side (red pixels), then the current labeling is kept. Otherwise, the labeling of left and right are switched.

In order to do the above described evaluation of possible flips, each pixel has to be assigned to its corresponding body part, i.e. to the corresponding bone. A naive assignment would be to select for every pixel the bone which

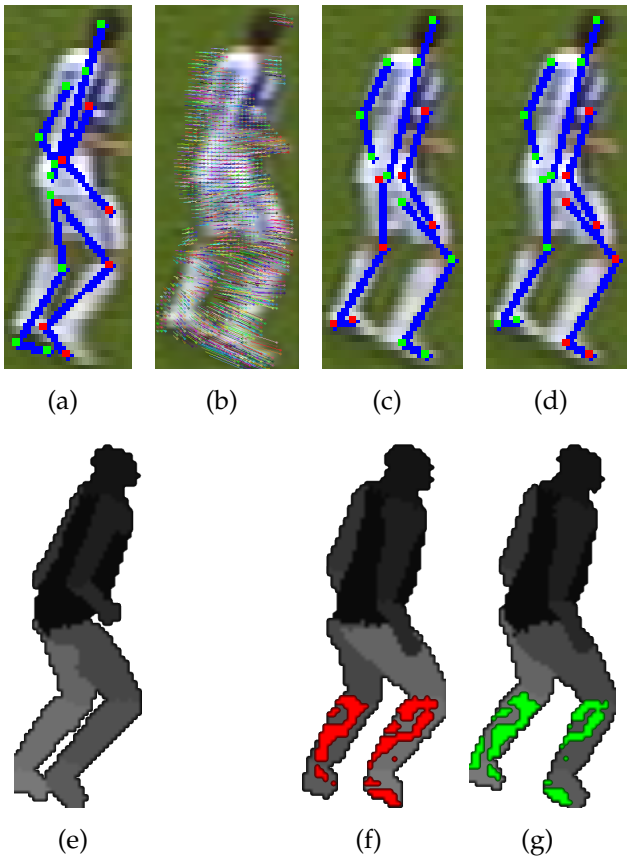


Figure 5.8: *Local consistency: (a) Previous frame. (b) Optical flow between previous and current frame. (c) Wrongly assigned legs in current frame. (d) Correctly assigned legs in current frame. (e) fitted mesh. (f) Fitted mesh in current frame with correct and wrong matches labelled as green and red, respectively. (g) Error of the flipped (correct) legs in the current frame.*

is closest in the 2D image. But this is incorrect, since it does not take into account occlusions. Also, an automatic segmentation of the body parts in 2D would not be appropriate for our setup due to the low resolutions and image artifacts. Therefore, we compute the correspondences in 3D using a 3D pose, that can be constructed using the calibrations and the 2D poses in all the cameras as described in section 5.3.4. This pose is computed for all possible flips in frame k (figure 5.8(f) and 5.8(g) show only the two possibilities for the legs) as well as for the pose in the previous frame $k-1$ (Figure 5.8(e)). Again, we use a template mesh deformed and rendered for all possible 3D poses using color coding for the bone labels. Thus, the pixel assignment is a simple lookup providing an accurate assignment that takes self occlusions into account. Figure 5.8(f) shows an initial wrong labeling with the found

correct (green) and wrong (red) pixel assignments from the optical flow. Figure 5.8(g) shows the correct labeling.

This resolves most of the flips of arms or legs. For failure cases, the user can select a frame and change the flipping for all subsequent frames of a sequence with a mouse-click. This is the only user interaction in our system and for a view of a player takes only about 1 click per 10 frames.

Global Consistency

After the local consistency step, all consecutive frames should not have flips between them which means that the entire sequence is consistent. As a consequence there is still the possibility that the entire sequence is flipped the wrong way. However, this is a simple problem as we only have a binary predicate to apply for the entire sequence for the legs and one for the arms. Together, these are four possible labeling combinations for the entire sequence. The final labeling is selected by choosing the combination that minimizes the following error term:

$$E_g = \lambda_t E_t + \lambda_{DB} E_{DB} \tag{5.7}$$

This is a weighted sum with constant parameters λ_t and λ_{DB}.

E_t is the already defined triangulation error. It implies that the resulting pose fits well into the given input.

E_{DB} ensures that the selected labeling/flipping results in *plausible poses*, i.e. realistic poses that are anatomically possible, along the sequence. Since we also have 3D poses in the database, we can use these poses to compute E_{DB}. We assume all poses in the database are plausible poses because they are taken from real data and are, therefore, anthropometrically correct. If a pose P for a flipping possibility is close to a pose in the database, then the chances are high that this pose is also plausible. Therefore, if we compute the difference of P to the closest (i.e., most similar) pose P_{DB} in the database, then this directly gives us an error value, that penalties for non-plausible poses:

$$E_{DB} = \min_{P_{DB}} \frac{1}{2|J|} \sum_{i=0}^{|J|} (\hat{\alpha}_i - \alpha_i)^2 + (\hat{\beta}_i - \beta_i)^2 \tag{5.8}$$

where α and β are the joint angles of P. $\hat{\alpha}$ and $\hat{\beta}$ are the ones of the database pose P_{DB}. Since E_{DB} only takes into account the angles, this error is independent of the body sizes and the global position of the pose. Also the orientation of the root bone is not taken into account.

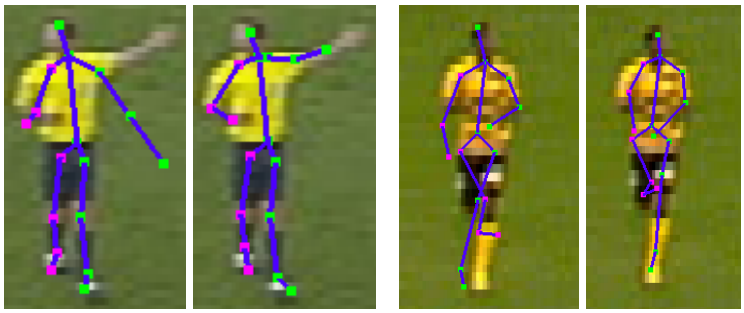


Figure 5.9: *Comparisons of the pose before (left) and after (right) the pose optimization.*

After selecting out of the four possible combinations the labeling that minimizes the sum of E_g of all frames, we have an initial pose estimation in all frames.

5.4 Pose Optimization

The initial pose guess transforms the poses from the database to the 2D views. However, besides the left/right flips no changes are made to the original 2D poses taken from the database. Therefore, the initial pose guess is limited to the poses of the database. But the database is only a subset of all possible poses, and thus, often does not contain the accurate solution. Therefore, this purely data-driven initial pose guess needs to be refined in an optimization step that adapts to the evidence and thus allows to fit also to poses that are not in the database.

We developed a new spatio-temporal pose optimization method to retrieve a more accurate estimation of the body pose as shown in figure 5.9. Our method is based on a energy function that consists of a combination of several spatial and temporal energy terms. The optimal solution is found using a minimization of this energy function.

The variables to be optimized are based on our representation of the skeleton S described in section 5.3.1. For each subject the variables are

- the root positions $\mathbf{p}_0$ per frame,
- the root orientations α_0, β_0 and γ_0 per frame,
- the relative joint angles α_i and β_i per frame,
- the bone lengths l_i which are global and

- the camera shifts $\mathbf{s}_c$ per frame and camera.

All the parameters besides the bone length are variable per frame. Fixing the bone lengths automatically introduces an anthropometric constraint, since bones should not shrink or grow over time. To initialize the bone lengths, the averages over all frames of the lengths given by the initial pose estimation are computed. Another nice property of the chosen skeleton representation is that it significantly reduces the number of variables because with this we have only one length per bone and only two angles per bone and frame. Again, the camera shifts are used to cope with calibration errors.

In section 5.4.1, the energy function is defined and in section 5.4.2 we explain the algorithm to minimize it.

5.4.1 Energy Function

The energy function is defined per subject for a given sequence of frames $T = \{t_0, ..., t_n\}$. It is the following weighted sum of error terms:

$$\begin{aligned} E(S) = \sum_{t \in T} (&\omega_s E_s + \omega_f E_f + \omega_{DB} E_{DB} \\ &+ \omega_{rot} E_{rot} + \omega_{rrot} E_{rrot} + \omega_p E_p) + \omega_l E_l \end{aligned} \tag{5.9}$$

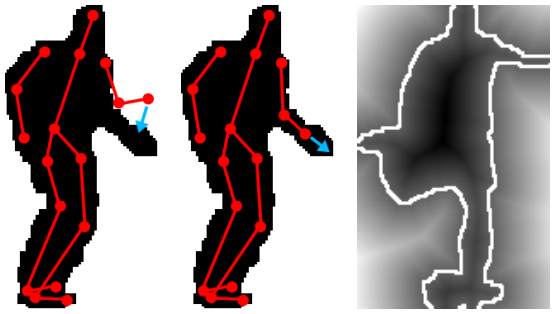
The following subsections describe the error terms in detail. For simplicity we omit the t in the equations.

Silhouette matching error term E_s

Using the calibration and the current camera shift, the current 3D joint positions can be projected into the 2D camera images. Their projection should be inside the silhouette in all camera views. If they are outside the silhouette, this means that we have joints or parts of bones that are not inside the visual hull. The silhouette matching error term E_s penalizes such joint positions. It is defined as

$$E_s = \frac{1}{|C||J_+|} \sum_{c \in C} \sum_{\boldsymbol{j} \in J_+} EDT_c(\boldsymbol{P}_c \boldsymbol{j} - \boldsymbol{s}_c), \tag{5.10}$$

where C is the set of all cameras that cover a silhouette of this subject. J_+ is the union of the set J of all joints and the points that are exactly in the middle of a bone. We could add more points on the bones to J_+ but in our experiments the joints and the points in the middle were sufficient. The normalized Euclidean distance transform EDT returns for every 2D point in the camera image the distance to the closest point inside the silhouette divided by the larger side



(a) (b) (c)

Figure 5.10: *Illustration of the two silhouette error terms. (a) The silhouette matching error term tries to move joints inside the silhouette. (b) The silhouette filling error term is responsible for the joints and bones to go into the extremities of the body. (c) Example of a EDT where the 0 iso-line is shown in white. The distances inside the silhouette are shown here too, but set to 0 in the algorithm.*

of the silhouettes bounding box. This normalization is important to make the error independent of the size of the subject in the camera image, which may vary according to the zoom. A visualization of such an EDT is shown in figure 5.10(c). $\boldsymbol{P}_c\boldsymbol{j}$ is the projection to transform the 3D joint $\boldsymbol{j}$ into camera space taking into account the camera shift.

The silhouette matching error pulls the joints inside the silhouette. An illustration of this is shown in figure 5.10(a).

Silhouette filling error term E_f

If only the silhouette matching error term E_s is used, joints outside the silhouette will be penalized. However, there is no restriction on them for where to be placed inside the silhouette and they usually shrink together inside the torso. Therefore, the silhouette filling error term E_f is added to prevent such a shrinking. Additionally, the silhouette filling error term makes sure that there are also joints in all the extremities. This is achieved by penalizing pixels of the 2D silhouettes that are far away from any bone. We define

$$E_f = \frac{1}{|C||R|} \sum_{c \in C} \min_{\mathbf{j} \in J} \sum_{\mathbf{r} \in R} dist(P_c^{-1}(\mathbf{r}_{i,c} + \mathbf{s}_c), \mathbf{j}), \tag{5.11}$$

where R is the set of all grid points from section 5.3.4 that are inside the silhouette.

The silhouette filling error term pulls the joints into the extremities. An illustration of this is shown in figure 5.10(b).

Distance to database pose error term E_{DB}

To ensure the final 3D pose to be a plausible pause which is kinematically correct (e.g., the knee joint bends the right way), we use the in section 5.3.5 defined distance E_{DB} to the closest pose in the database. This uses the advantage of our database that consists of correct 3D poses. It implicitly adds anthropometric constraints to our optimization and can be seen as the prior.

Smoothness error terms E_{rot}, E_{rrot} and E_p

When working with TV camera footage, which has usually frame rates higher than or equal to 25 fps, the amount of human motion between two consecutive frames is restricted. This can be used directly in the error function by adding three smoothness constraints. This enables us to introduce temporal coherence to the pose optimization and will also result in smoother final renderings of sequences.

The first smoothness error term E_{rot} penalizes large changes of the internal angles of the skeleton of consecutive frames:

$$E_{rot} = \frac{1}{2|J|} \sum_{i=0}^{|J|} (\alpha'_i - \alpha_i)^2 + (\beta'_i - \beta_i)^2, \tag{5.12}$$

where α' and β' are the corresponding angles of the same subject in the previous frame.

The second term E_p constrains large movements (translations) of the entire body relatively to the world coordinate system. We define

$$E_p = |\mathbf{j}_0 - \mathbf{j}'_0|, \tag{5.13}$$

where $\mathbf{j}'_0$ is the global position of the root joint in the previous frame.

Finally, a third term E_{rrot} also constrains the global rotation (relative to the world coordinate system) of the root bone:

$$E_{rrot} = \frac{1}{3} \left((\alpha'_0 - \alpha_0)^2 + (\beta'_0 - \beta_0)^2 + (\gamma'_0 - \gamma_0)^2 \right) \tag{5.14}$$

where α'_0, β'_0 and γ'_0 are the angles of the global orientation of the backbone in the previous frame.

Length error term E_l

So far, we do not have any constraints on the lengths of the bones. Also the distance to database error term only compares angles between bones but not their lengths. Therefore, we add as a final error term the length error term E_l to the target function. It is based on the fact that the initialization of the bone lengths is already a good approximation when handling a sequence of frames. This is because it is set to the average length of the estimated bones of all frames. Therefore, the length error term tries to keep the optimized pose close to these lengths:

$$E_l = \frac{1}{|J|} \sum_{i=0}^{|J|} (l_i - \hat{l}_i)^2 \tag{5.15}$$

where l_i is the current bone length and $\hat{l}_i$ is the initial bone length.

5.4.2 The Optimization Procedure

The energy function given in equation (5.9) is minimized to optimize the body poses. Assuming that the initial pose estimation already gives an initialization close to the optimal solution, we can employ a simple and local optimization strategy. It iteratively optimizes the variables one by one by performing line search along randomly picked directions [Schreiner et al., 2004]. For each variable we select 10 random directions for optimization and we perform 20 global iterations. This optimization method performed well and can cope with the non-smooth nature of our objective functions, i.e. the problem that we can not compute derivations analytically for some of the error terms.

Because our energy function can be formulated as a sum of squares, we also tested Levenberg-Marquardt [Mor, 1978] with numerical derivations to minimize the energy, but yielded a lower performance. However, for future work, it would be interesting to try to locally compute the derivations analytically also for the problematic error term, i.e., the distance to database error term. This could be done by computing the gradient to the distance to the closest pose - without allowing to switch the pose. This could then be used together with a gradient descent approach, by recomputing the derivation of this energy term always for the current closest pose.

5.5 Implementation

The database is stored into files. For each silhouette, there are two files. The first one is an image file containing the color image as well as the silhouette in the alpha channel. The second one contains the meta data, which is the body pose in 2D and 3D, the camera calibration and information about where it was taken from. To speed up the pose estimation, all files are read into memory when starting up the system.

For the initial pose estimation, the silhouette comparisons are currently done on the CPU. The reason for this is that the small size of the comparison grid will not gain much from the parallelization on the GPU due to the overhead of uploading and downloading data to the graphics card. If there is enough texture memory available, one could think of storing the reference poses on the GPU, e.g., as collections in large textures similar to Germann et al. [2007]. This would allow to compare one input silhouette to more than one database silhouette at the same time. Because we use over 20000 silhouettes with a raster of 32×40 each, this would result in textures of the size of roughly $25,600,000$ pixels. Since the silhouettes are binary, this could be compressed, e.g., 8 binary values per 8-bit color channel. This would reduce the space for the entire database (without body poses) to a single channel square texture with 1790 pixel side length.

Both parts, the initial pose estimation and the pose optimization, are run independently on every input subject in the scene. Therefore, we implemented both parts in threads that run in parallel for each subject.

For the parameters in equations (5.9) and (5.7) were used an automatic parameter tuning procedure to find the optimal values. To achieve this, we annotated manually the 2D poses in two scenes over several frames. This gives us ground truth values for evaluating the algorithm. The algorithm was executed on these scenes and the final pose estimation was compared to the manual annotations by evaluating the average distance of the joints in 3D. This average distance directly gives us an error value E_Q to measure the error of the pose estimation. To optimize a parameter ρ, which can be any of the parameters in equations (5.9) or (5.7), the algorithm is run for different values of ρ. We used a coarse-to fine search over plausible values to find an optimal $\hat{\rho}$ that minimizes E_Q. The procedure also iterates over the possible parameters, i.e., for every ρ in equations (5.9) and (5.7). This automatic parameter optimization was done only once and the found parameters were used for all scenes and all frames.

Table 5.1: *The parameter values that we used for all our results, computed using our automatic parameter tuning system*

Param.	ω_s	ω_f	ω_{db}	ω_{rot}	ω_{rrot}	ω_p	ω_l	λ_{DB}	λ_t
	9	15	0.05	0.1	5	1	1	0.15	0.3

Figure 5.11: *Estimated poses in a soccer game. The image shows a projection of the 3D skeletons into a source camera.*

5.6 Results

To evaluate our system we used original TV footage. It was acquired from TV broadcasts of soccer games from which we have access to all camera streams. From this we took five sequences with two or three cameras, yielding roughly 1800 poses to process.

5.6.1 Settings

For the optimization functions in equations (5.9) and (5.7) we used the parameters shown in table 5.1 for all our results which were found in the automatic parameter optimization described in section 5.5

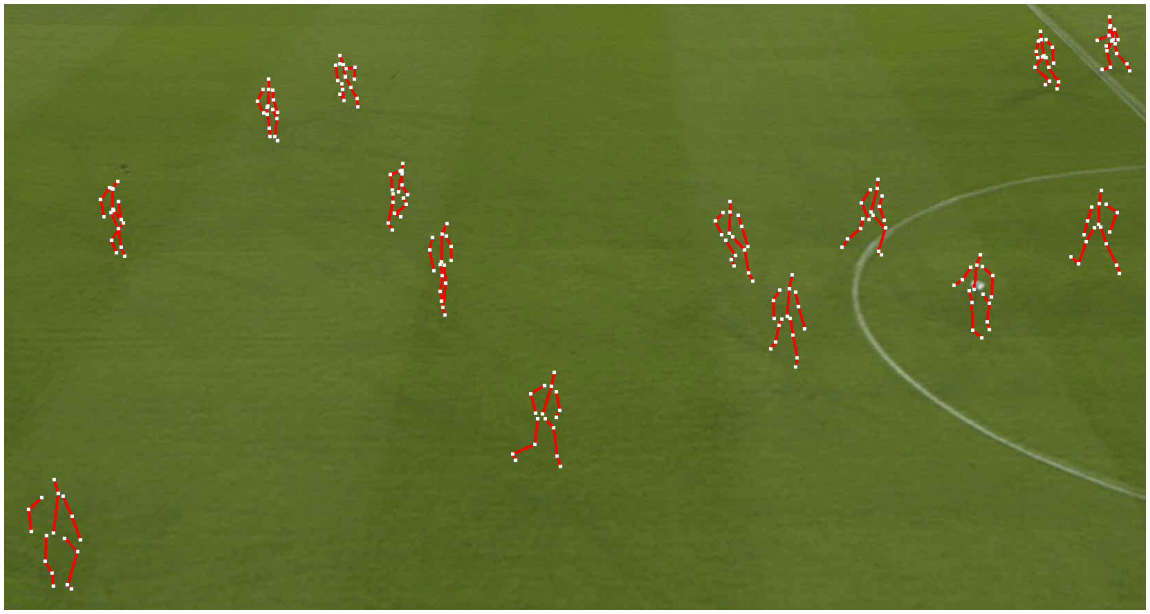


Figure 5.12: *3D rendering of the detected joint positions.*

5.6.2 Sequences

A subset of the results is shown in figures 5.11, 5.16 and 5.17.

Each column in figures 5.16 and 5.17 shows a set of consecutive poses and each item shows the image of the respective player in all the available cameras. Even with only two cameras and very low resolution images, our algorithm can retrieve good poses in most cases.

5.6.3 Application

A possible application for our pose estimation is to visualize the 3D pose of the skeleton. In figure 5.12 we show the 3D joint positions by rendering them from a novel viewpoint (i.e. a non camera view).

Another example for an application of our pose estimation, is to use articulated billboards (see chapter 6). In figure 5.13 we show some results of renderings from non-camera viewpoints, i.e. where no camera has been recording from. Even though the source footage covers the players only at low resolution, the final output still looks pleasing and is comparable to the input. Figure 5.14 shows close up shots from a novel viewpoint also based on low resolution input.

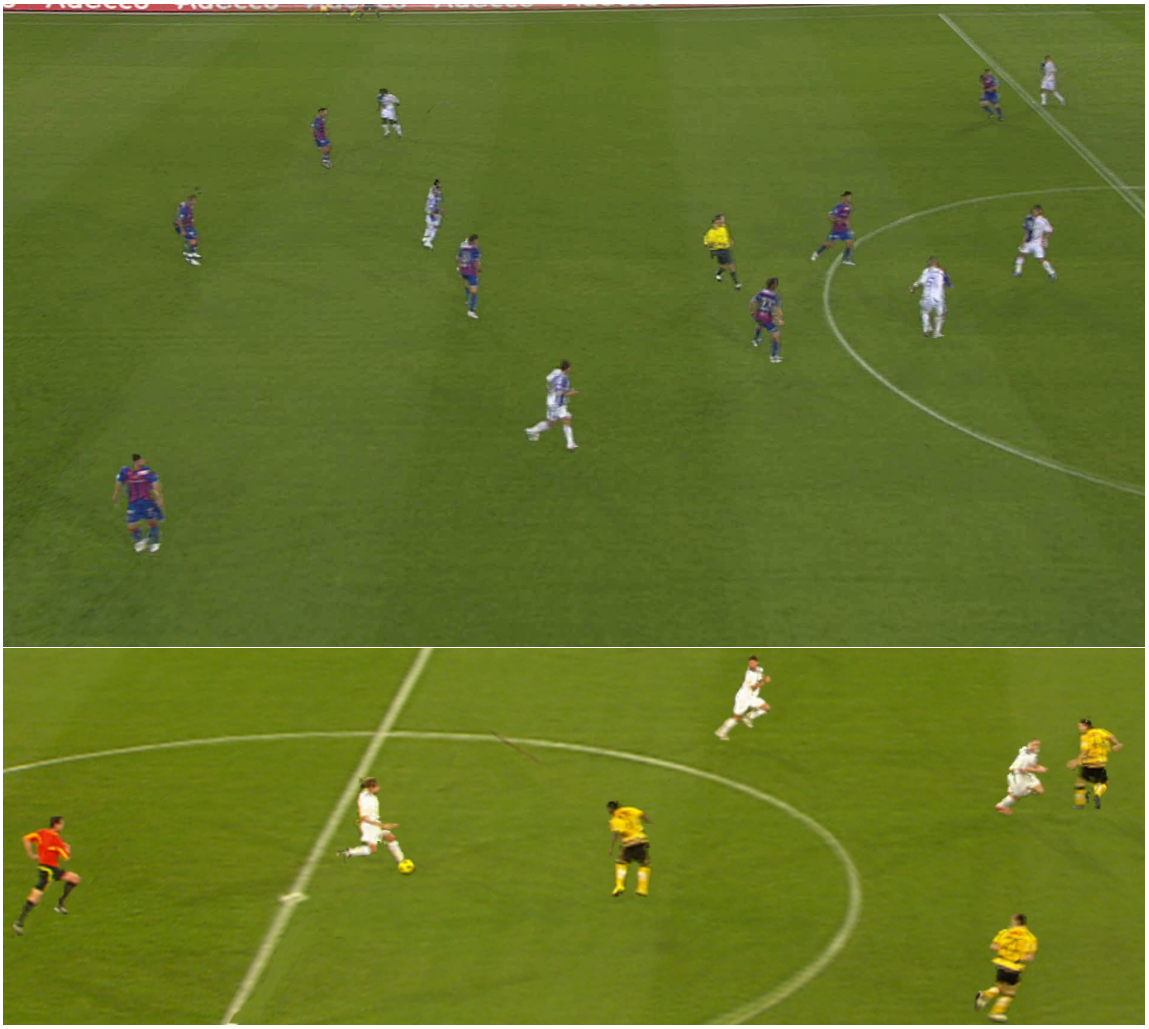


Figure 5.13: *Renderings from novel viewpoints of soccer scenes using our pose estimation and articulated billboard rendering.*

5.6.4 Timings

The single thread version of our pose estimation algorithm takes about 40 seconds per player per frame in a two camera setup and about 60 seconds for a three camera setup. The parallel version that runs a thread for every player as explained in section 5.5 can reduce this time. On an 8 core system this gave a speedup of roughly a factor of 8 and results in about 5 to 8 seconds per player and frame. The time used for the algorithm is splitted roughly into 34% for the initial pose estimation, 7% for the local consistency, 8% for the global consistency and 51% for the pose optimization.

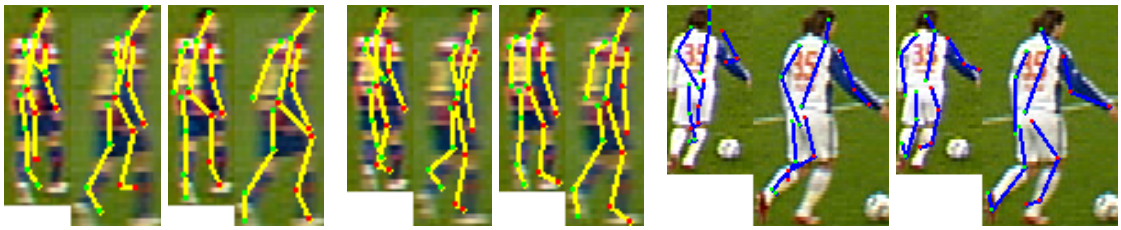
Figure 5.14: *Renderings from novel viewpoints: close up views.*

5.6.5 Comparison With/Without Pose Optimization

The improvement achieved in the pose optimization can be seen in Figure 5.9. The leftmost example is an example for the influence of the silhouette filling error term. For the arm of the player, both directions (rotating up or rotating down) would reduce the silhouette matching error term, but only rotating up would also reduce the silhouette filling error term. Therefore, this direction is chosen by the optimization. Figure 5.15 shows more examples each of them for two cameras before and after the optimization.

5.7 Discussion and Outlook

In this chapter we presented an algorithm for multi-view body pose estimation of sequences in uncontrolled environments. It focuses on source footage



(a) (b) (c)

Figure 5.15: *Examples for the improvement of the pose optimization. Figures (a), (b), and (c) are three examples, each of them showing the projections into both cameras (connected images) before and after the optimization.*

taken from TV broadcasts of soccer games. To compute accurate 3D poses, it relies on a rich database of poses and uses the temporal coherence of human motion as a strong prior.

The proposed data-driven pose estimation algorithm works in two main stages. For the first step, we introduced a novel spatio-temporal search to retrieve a good initial pose based on silhouette matching. A mesh based consistency check resolves for unwanted flips of the limbs, avoiding local minimas in the optimization step. In a second step, the initial pose estimation is improved using a novel optimization technique that combines spatial and temporal constraints to yield the final pose.

Even though the initial pose estimation makes use of the video images of previous frames, it does not depend on the result of the pose estimation of the previous frame. Thus, it does not suffer from drift which usually occurs in tracking based methods. As a result, our method is able to recover from bad pose guesses.

We showed results from real soccer scenes taken from TV cameras, where our algorithm is able to reasonably estimate the human body poses. As a direct application we showed that the pose estimation allows to render novel viewpoints using articulated billboards.

Our method leads to good results in many cases. However, there are situations where it can fail. One main reason for this is the fact that for most parts of the algorithm only coarse binary silhouettes are used. This is particularly the problem, when the arms are too close to the body as shown in figure 5.18(a), because several possible poses can have very similar binary silhouettes. If this is the case over many frames, then also the pose optimization is not able to recover the correct position. In such situations, only using silhouette information is not sufficient to disambiguate the poses. We would

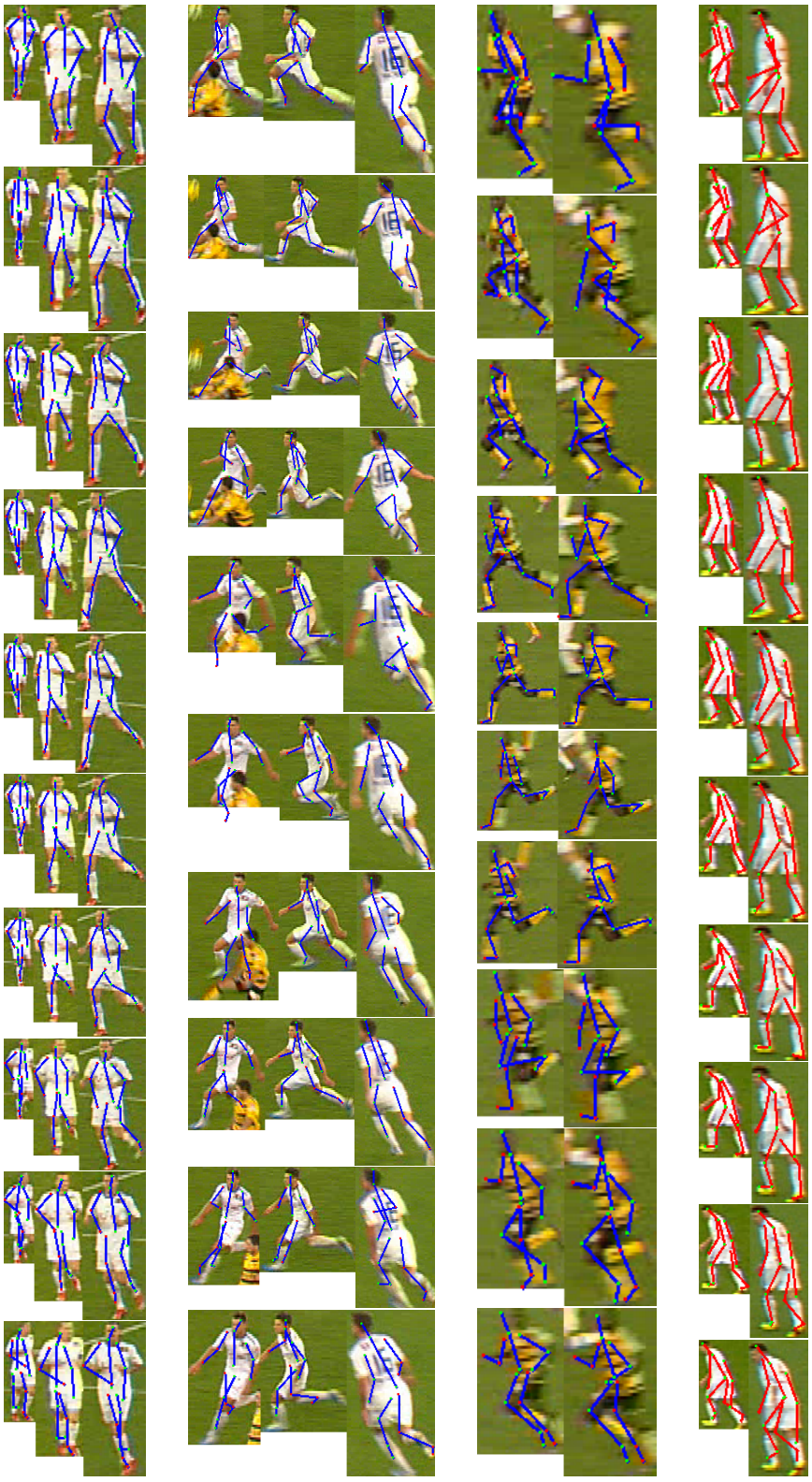


Figure 5.16: *Result sequences with all used camera views shown per frame.*

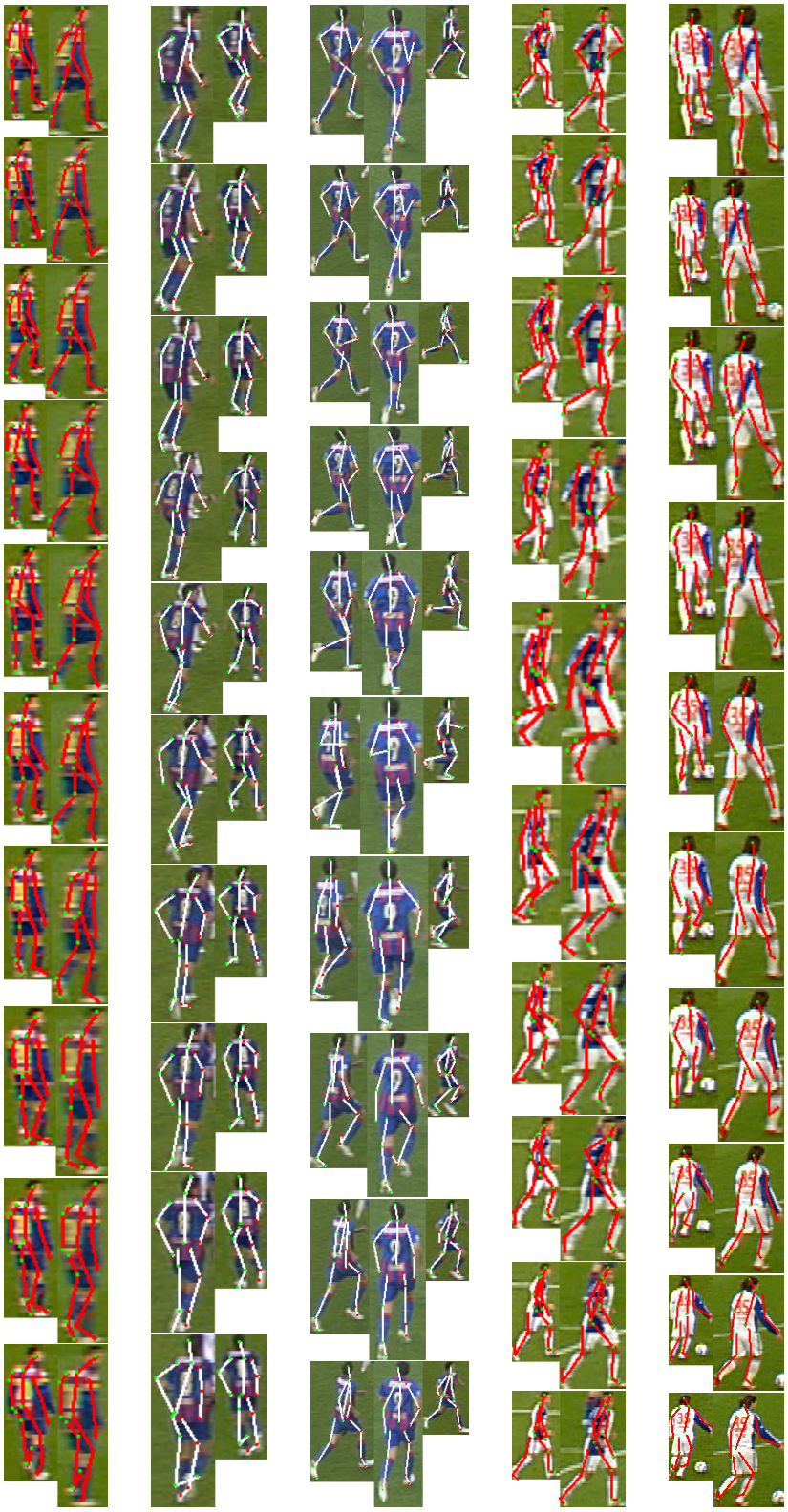
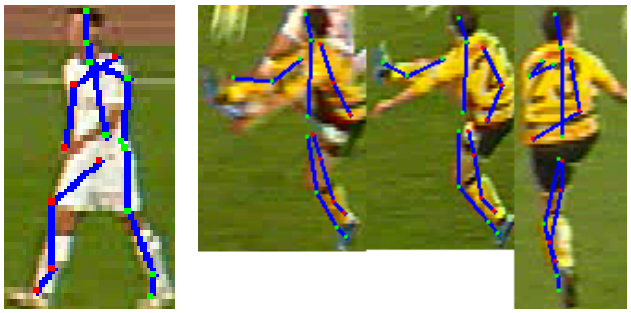


Figure 5.17: *Result sequences with all camera views shown per frame.*



(a) (b)

Figure 5.18: *Failure cases. (a) The arms are too close to the body and could not be positioned correctly. (b) A pose that is too far from the database and could not be estimated correctly.*

like to invest on incorporating optical flow into the optimization procedure, to be able to resolve such ambiguities.

Another limitation of our method is that the results greatly depend on the pose database. A good database should have a wide range of motions as well as a wide range of views such that the initial guess is close to the correct pose. Figure 5.18(b) shows an example where there was no similar pose in the database. The pose estimation failed such that the pose optimization was not able to recover the correct pose. In the future, we would like to find an automatic criterion to quantify a good match of a pose. Identifying good poses automatically will help to select the poses that should be added to the database and thus help to enlarge the space of possible poses.

In our algorithm, we use only implicit anthropometric constraints by using the distance to pose database error. In addition to this we would like to add specific anthropometric constraints on joint angles or bone lengths that, e.g., will not allow a knee to bend backwards or any other joint to do unnatural movements. This would reduce the search space and improve the results.

As another future work, we would like to investigate on methods for a faster search in the database, since in our current implementation the input is compared to all poses in the database. An algorithm that could be used for a speedup is parameter-sensitive hashing [Shakhnarovich et al., 2003] but applied on silhouettes instead of features.

CHAPTER 6

Articulated Billboards

In the last chapter, we have seen that the human body pose can be estimated even in uncontrolled outdoor setups with low resolution. In this chapter, we present *articulated billboards*, a representation and rendering method for the human body that makes direct use of the estimated body pose. It also targets on uncontrolled outdoor setups but despite of the low quality input is able to render realistic novel views.

There exists many methods for rendering novel views of the human body. However, as stated in section 3.2, most of them target on studio setups, high resolutions, dense camera setups and controlled lighting. The most promising approaches in challenging outdoor setups are the ones by Guillemaut et al. [Guillemaut and Hilton, 2011; Guillemaut et al., 2009; Hilton et al., 2011]. Their method allows for realistic renderings from arbitrary view points by a more accurate geometry than the standard visual hull. However, it still requires a fairly large number of cameras (6-12).

When working with only a few input cameras as in our target setup, billboarding ([Hayashi and Saito, 2006; Inamoto and Saito, 2007]) has shown to be more robust to calibration errors than other methods. The reason for this is that billboards are usually placed orthogonal to their corresponding cameras viewing direction. Thus, shifting the billboard or its camera will only cause to shift the entire projection of this human and camera. This increases the ghosting depending on the directions of the calibration errors of the cameras

to each other. However, the distortion is less disturbing than, e.g., with visual hulls, where entire body parts will be cut off and not shown at all if the calibration is inaccurate. On the other hand, billboarding does not render an articulated pose with correct self-occlusions and parallax. Either flat looking parts or disturbing ghosting effects will always be present if the human body is not on or nearly on a single plane.

The optimal representation and rendering method would be something between billboarding and methods with higher geometric detail. It should combine the robustness against calibration errors of billboarding with the articulation details of visual hulls or template based methods. This is exactly what the in this chapter presented articulated billboards do. The general idea is to split the human body into mostly rigid and simpler body parts that can be rendered separately and similar to billboarding. This results in a robust and articulated representation of the human body, suited for difficult setups.

The work described in this chapter was presented in [Germann et al., 2010].

6.1 Overview

Our method to construct and render articulated billboards can be divided into the following four main steps, visualized in figure 6.1.

In a first step, the 2D and 3D body poses of a subject are estimated. This is done by using the pose estimation algorithm from chapter 5. The found pose will form the basis for the underlying skeleton structure of the articulated billboards representation.

Given the 2D joint positions, in a second step a segmentation of the image into the different body parts is computed. For this step we imply a human template model in order to map image-pixels to billboards.

The third step of the algorithm integrates the pose and texture information from all individual views and generates the final articulated billboard model for rendering. This processing step includes a further optimization of the 3D joint positions and the camera shifts, which improves the texture overlap for each model segment, i.e., for the billboards in each body part. A final alpha-mask and texture optimization eliminates visible seams and discontinuities between adjacent billboards of different body parts.

The last step is the actual real-time rendering of novel views. We developed an algorithm for a fully GPU-based, view-dependent per-pixel blending scheme, which is optimized for rendering articulated billboard models efficiently while preserving the photo-realism of the original input video.

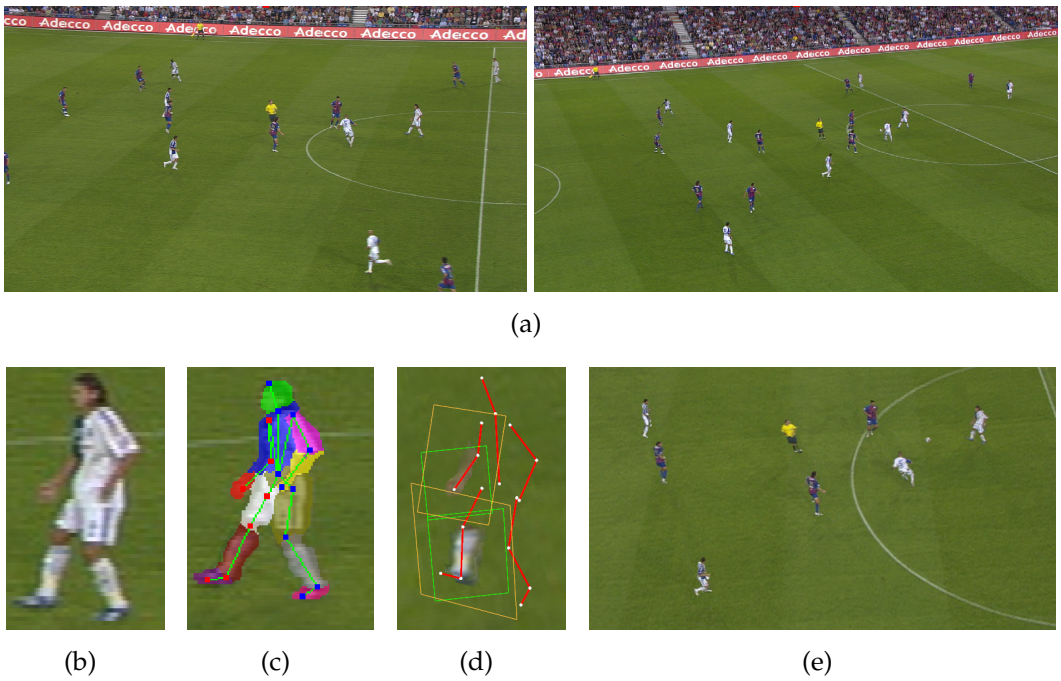


Figure 6.1: *Overview of our method. (a) Two wide-baseline input video frames of a soccer match. (b) Zoom on one of the players. (c) We first compute the subject's 2D pose in the input views and a segmentation into the different body parts. (d) A multi-view optimization then generates a 3D articulated billboard model. For clarity we show only a subset of the billboards in this example. (e) With the articulated billboard models photo-realistic views from a large range of novel viewpoints can be rendered.*

We will give first a more detailed definition of the articulated billboards representation in section 6.2. Following this, sections 6.3, 6.4, 6.5 and 6.6 explain the four steps in detail. Section 6.7 will give more detail about the implementation. Finally, we will show and discuss results in section 6.8 and conclude this chapter in section 6.9, where we also give ideas about future work.

6.2 Articulated Billboards Representation

We propose a representation based on *articulated billboards*. The basis of this model is a 3D human skeleton structure (see figure 6.2(a)). Every bone, represented by a 3D vector $\boldsymbol{b}_i$ and the position of its start-joint $\boldsymbol{j}_i$, corresponds to a major component of the body, e.g., the torso or the extremities. With each bone we associate a fan of billboards, which contains a billboard for every input image $\boldsymbol{I}_j$ of a subject (see figure 6.2(b)). More specifically, for each $\boldsymbol{I}_j$ the

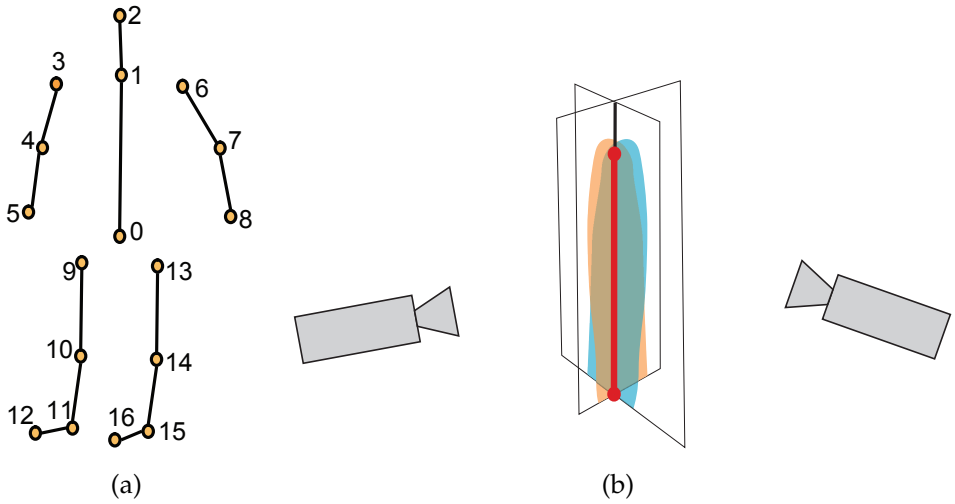


Figure 6.2: *(a) Skeleton structure used for our articulated billboard model. (b) Illustration of a single fan of two billboards and the corresponding source cameras.*

corresponding billboard plane is defined by the joint $\boldsymbol{j}_i$, the bone direction $\boldsymbol{b}_i$, and the vector $\boldsymbol{b}_i \times (\boldsymbol{c}_j - \boldsymbol{j}_i)$, where $\boldsymbol{c}_j$ is the camera position of $\boldsymbol{I}_j$. Hence, the billboards are aligned with the character bones and as orthogonal as possible to their associated input views.

6.3 Body Pose Estimation

The body pose estimation of chapter 5 can be used as it is. The skeleton we use for the articulated billboards is the same as in chapter 5. Thus, it only uses the images, rough calibrations and rough silhouettes as input. the directions of the bones are set to $\boldsymbol{b}_i = \boldsymbol{j}_{i+1} - \boldsymbol{j}$.

In our initial publication [Germann et al., 2010] we used a strongly simplified version of the pose estimation of chapter 5. Only the first part, the silhouette based 2D pose estimation was done and by using only the current frame without a sliding window. The simplified version did not solve for left-right ambiguities and also the pose optimization described in section 5.4 was not done. Therefore, for every view and subject, many joints had to be corrected manually, which is very time consuming. However, the only reason for this has been that the pose estimation was developed after the articulated billboards. Therefore, it was not available at that time. In contrast to that, we assume here the results of the pose estimation as input for the following sections in this chapter.

6.4 Template-Based Segmentation

To construct and render articulated billboards, a more detailed segmentation is needed than just the silhouettes (background subtraction). The reason for this is that we need to assign to every pixel in the source images not just the corresponding subject (or background) but also the corresponding body part. According to this assignment, the pixel will be assigned to the correct billboard later on.

Even with estimated 2D joint positions a robust segmentation of the image into the subject's body parts is still a difficult problem. One possible solution would be to use a database of silhouettes that are segmented into body parts instead of the above binary silhouette segmentation. However, this would make the creation of the database very complex and time-consuming. And still, we could not expect to always find sufficiently accurate matches.

We developed a method that fits a generic, pre-segmented 3D template model to the images and uses this to generate different confidence regions in the 2D images. We use the 3D poses from chapter 5 and fit a rigged 3D template model of a human body to it. The deforming can be done according to standard techniques for skeleton-based animation [Lewis et al., 2000]. From this, a virtual silhouette can be generated by projecting the model into the camera image. This approach has the considerable advantage that we get a good starting solution for the segmentation process and that we can easily resolve occlusions.

The fitted, pre-segmented template model does not perfectly segment the input frame $\boldsymbol{I}_k$. Also it might not correctly cover the entire silhouette. Therefore, a refinement of the segmentation is done in three steps. In a first step, a color model is learned per body segment based on automatically selected *confident* pixels of the pre-segmented body parts (see figure 6.3(a)). In a second step, the trained color model is used to label the *unconfident* pixels leading to a segmentation adjusted to the subjects body dimensions and silhouette (see figure 6.3(b)). In a third step, a morphological closing operation removes outliers as depicted in figure 6.3(c).

We will describe these steps in detail in the following subsections.

6.4.1 Selecting Confident and Unconfident Pixels

To determine the confident pixels, we project a slightly thinned and thickened version of the template model into the image and label the silhouette pixels accordingly. This is done by translating all vertices of the mesh in the direction

(a) (b) (c)

Figure 6.3: *Body segmentation. (a) Initial segmentation with safe pixels derived from the template model and unconfident boundary pixels. (b) Segmentation after labeling according to the trained color model. (c) Final segmentation after morphological removal of outliers.*

of the surface normal or the opposite direction, respectively. The model is projected for both versions, the grown and the shrinked one. These two projections yield two different segmentations in the camera images. Pixels which receive the same label in both projections are marked as confident pixels because they do not change their label within this range of possible errors in the size of the body parts. These pixels are directly labeled with the corresponding body segment. All remaining pixels within the silhouette are labeled as unconfident as shown in figure 6.3(a).

6.4.2 Segmenting Unconfident Pixels

To determine the most probable label for every pixel marked as unconfident, we use the confident pixels as a learning sample. Using only the confident pixels, a Gaussian mixture model (section 4.4.1) is learned for each body part label. By learning a color model on-the-fly, we provide a robust segmentation algorithm being able to handle segmentation in uncontrolled environments. Changing lighting conditions, subject specific appearance or view dependent appearance can thus be handled reliably.

This color model is then directly used to determine the label of each unconfident pixel. To reduce the amount of computation, only the two possible labels determined in section 6.4.1 do have to be considered and their GMMs tested.

(a) (b) (c)

Figure 6.4: *3D template fitting: (a) Corrected joint positions. (b) Initial fitting of the pre-segmented 3D shape template using the method of Hornung et al. (c) Our corrected fit which exactly matches the joint positions in (a).*

This leads to a segmentation of all pixels into body parts. To remove errors from noise, a morphological closing is applied at the end. An example of a final result can be seen in figure 6.3(c). Because of the use of a 3D model, this segmentation method automatically handles occluded body parts. It is robust, even for low quality input.

6.4.3 Using an Accurate 2D Pose

If an accurate 2D pose is available, i.e. manually corrected 2D joint positions, then this can be used to fit an optimal 3D model not per subject to all views but for each view separately. Because this manual correction had to be done anyways in the original publication [Germann et al., 2010], we used this also in the segmentation. We describe this here as an alternative to the direct use of a 3D body pose estimation.

Fitting a 3D model for each view separately according to the 2D pose means that the projection of the 3D pose *perfectly* aligns with the 2D joints. In a first step, only an *approximate* 3D pose is computed. This is done according to the method for 3D articulated models from a single image as presented by Hornung et al. [Hornung et al., 2007]. Given the 2D joint positions $\boldsymbol{j}_i$ for an image $\boldsymbol{I}_k$, their approach uses a database of 3D motion capture data to find a set of 3D joint positions $\boldsymbol{j}_i$ whose projection approximately matches the 2D input joints (see figure 6.4(b)). This can be used as a basis. We provide a simple but effective modification to their algorithm for computing the required *accurate* fit.

The approximate 3D match can be deformed, such as to align with the 2D joints according to the following algorithm. Through each 3D joint $\boldsymbol{j}_i$, we create a plane parallel to the image plane of $\boldsymbol{I}_k$. Then, we cast a ray from the camera center $\boldsymbol{c}_k$ through the corresponding target joint position $\boldsymbol{j}_i$ in $\boldsymbol{I}_k$ and compute its intersection with the plane. The 3D pose is then updated by moving each $\boldsymbol{j}_i$ to the respective intersection point and updating the 3D bone coordinate systems accordingly. The result is the required 3D pose which projects exactly onto the given 2D joints. Finally, the rigged model is deformed to fit the 3D skeleton (see figure 6.4(c)). Note that this algorithm generally does not preserve the limb lengths of the original 3D skeleton and therefore, enables an adaptation of the 3D template mesh to fit the subjects dimensions more accurately.

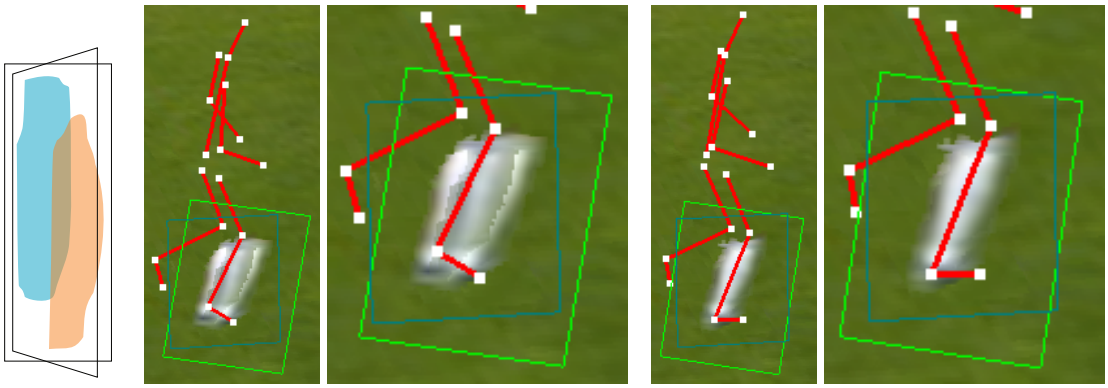
The rest of the segmentation algorithm is then proceeded as described in sections 6.4.1 and 6.4.2.

6.5 Result-Based Optimization

After the segmentation of the silhouette into body parts, everything that is required for the construction of articulated billboards is ready. The articulated billboards could be placed in 3D world coordinates according to the joint positions of Section 6.3 and could be rendered using the segmentation. However, if a 3D joint of the articulated billboard model is not optimally positioned, the texture resulting from the rendering of all billboards of a billboard fan will not align. Figures 6.5(a) and 6.5(b) show an example for this, where for visualization only the lower part of the leg is rendered. In this part a ghosting appears because of a misalignment.

It seems difficult to remove such ghosting since we usually do not even know the source of it. The ghosting could occur even if the 3D pose is perfectly correct but the camera calibrations are not shifted correctly or vice-versa. The final decision if the positioning of the joints in 3D is optimal and thus the ghosting is minimized can usually only be made when looking at renderings. In this section, we describe how the rendering results can be used to optimize the 3D joint positions by automatic evaluations using a quantitative measure of the alignment of the billboard textures.

In the following, we first define a scoring function for a position of a joint in one view and for one camera pair. This scoring function is then extended to several views and cameras. Using this scoring function and anthropometric constraints the 3D pose of the articulated billboard model is optimized.



(a) (b) (c)

Figure 6.5: *(a) Illustration of a misaligned billboard fan. (b) Billboard fan before joint optimization. (c) Result after optimization. Note the improved texture alignment.*

Finally, we will describe a seam correction which removes texture discontinuities between adjacent billboards.

6.5.1 Position Scoring

To score the quality of a joint position of an output view V, all billboards adjacent to this joint are evaluated. For each fan of billboards, the alignment of its billboards for a pair of input views ($\boldsymbol{I}_1$, $\boldsymbol{I}_2$) is scored by a pixel-wise comparison of the projected textures. For every output pixel $\boldsymbol{u}$ of V, the per-pixel score $s_{I_1,I_2}(\boldsymbol{u})$ is defined as

$$s_{I_1,I_2}(\boldsymbol{u}) = \begin{cases} 1 - \epsilon(V_{I_1}(\boldsymbol{u}), V_{I_2}(\boldsymbol{u})), & \boldsymbol{u} \text{ active in } \boldsymbol{I}_1 \text{ and } \boldsymbol{I}_2 \\ 0, & \text{otherwise} \end{cases}, \tag{6.1}$$

where $V_{I_j}(\boldsymbol{u})$ is the color contribution of a billboard associated with view $\boldsymbol{I}_j$ to pixel $\boldsymbol{u}$. $\epsilon(\cdot)$ is a color distance measure in RGB. We used the sum of squared differences of the separate color channel, but also more elaborated comparisons could be used. The *active pixels* are defined as those pixels in the output view V which receive a valid color contribution from the input views $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$. The segmentation generated in section 6.4 is used to reliably resolve occlusions.

The score for a joint in a view V is the normalized sum of all pixels

$$s_{I_1,I_2}(V) = \frac{\sum_{\boldsymbol{u}\in V} s_{I_1,I_2}(\boldsymbol{u}) n(\boldsymbol{u})}{\sum_{\boldsymbol{u}\in V} n(\boldsymbol{u})}. \tag{6.2}$$

The normalization factor $n(\boldsymbol{u})$ is 1, if at least one of the two pixels is active and 0, otherwise. Thus, the scoring function measures the matching of texture values, while $n(\boldsymbol{u})$ penalizes non-aligned parts as in figure 6.5(a). These pixel-wise operations are efficiently implemented on the GPU using fragment shaders.

For more than two input views, we define the score as a weighted average of all camera pairs, where the weight depends on the angle β_{I_1,I_2} between the respective viewing directions, with narrow angles receiving a higher weight:

$$s(V) = \frac{\sum\limits_{(I_1,I_2)\in\mathcal{I}} s_{I_1,I_2}(V)\omega(\beta_{I_1,I_2})}{\sum\limits_{(I_1,I_2)\in\mathcal{I}} \omega(\beta_{I_1,I_2})}, \tag{6.3}$$

where $\mathcal{I}$ is the set of all pairs of input views and $\omega(\beta)$ is a Gaussian weight:

$$\omega(\beta) = e^{-\frac{\beta^2}{2\sigma^2}}. \tag{6.4}$$

The value for σ was empirically determined to be 0.32. Finally, the score of the joint position is the normalized sum of the scores in all evaluated views:

$$S_{\mathcal{V}} = \frac{1}{|\mathcal{V}|}\sum_{V\in\mathcal{V}} s(V), \tag{6.5}$$

where $\mathcal{V}$ is the set of all evaluated views.

Since the scoring of the joint position depends on the evaluated views, we need a suitable set $\mathcal{V}$. This set can consist of the views of the original input cameras but also any arbitrary view can be used. Theoretically, an evaluation set $\mathcal{V}$ would be optimal if it contains all views that will be used later on in the rendering process. However, this would be computationally too expensive and also we can not assume to know this set of views that will be used. In order to reduce the computation time but still cover a reasonable range of viewing positions, we evaluate the scoring function at the camera positions of all input views and at the virtual views in the center between each camera pair.

6.5.2 3D Pose Optimization

The scoring function to evaluate one skeleton pose can be used directly to optimize the joint positions. The position of each joint can be optimized according to the following method.

After evaluating $S_{\mathcal{V}}$ of the current joint configuration, we evaluate $S_{\mathcal{V}}$ at spatially close candidate positions around the current joint position on a discrete, adaptive 3D grid. If a better (higher) score is reached at one of this possible positions, then this will be set in a greedy manner to the current optimal position. This is repeated iteratively. However, the grid size is reduced in every iteration to a finer resolution. These steps are repeated until a given grid resolution is reached (empirically set to 1.2 cm).

To avoid degenerate configurations with billboard fans of zero length, we additionally consider the anthropometric consistency [NASA, 2009] during the evaluation of each pose. A joint position receives a zero score if one of the following constraints does not hold:

- The joint is on or above the ground.
- Lengths of topologically symmetric skeleton bones (e.g., left/right arm) do not differ more than 10%.
- The lengths of adjacent bones are within anthropometric standards.
- Distances to unconnected joints are within anthropometric standards.

For the last two constraints, we use the 5th percentile of female subjects rounded down as minimal lengths and the 95th percentile of male subjects rounded up as maximal lengths. These values were taken from [NASA, 2009].

This grid-search optimization process is iteratively repeated over the skeleton. In our experiments, we found that it typically converges after 4 iterations. Figure 6.5 depicts an articulated billboard model before and after optimization. It shows that the area of overlap is enlarged and also the colors match better, i.e., the ghosting is reduced.

6.5.3 Texture Seam Correction

Articulated billboards are rendered with a projective texturing approach according to the segmentation masks. This is illustrated in figure 6.6(a). Due to the sampling according to the textures, small discontinuities between adjacent billboards might appear in the output view, visible as cracks in the surface. An example for this is shown in figure 6.6(b). This happens if the geometry changes within the view frustum of one pixel, i.e., if parts of this pixel are on the surface of one body part and parts on the surface of another body part. In the segmentation the pixel is labeled to belong to only one of the two body parts. When rendering a closeup or just a different view, however, the sampling of the rendering might be finer or a bit shifted than the one of the original camera view which defined the mask. This is exactly

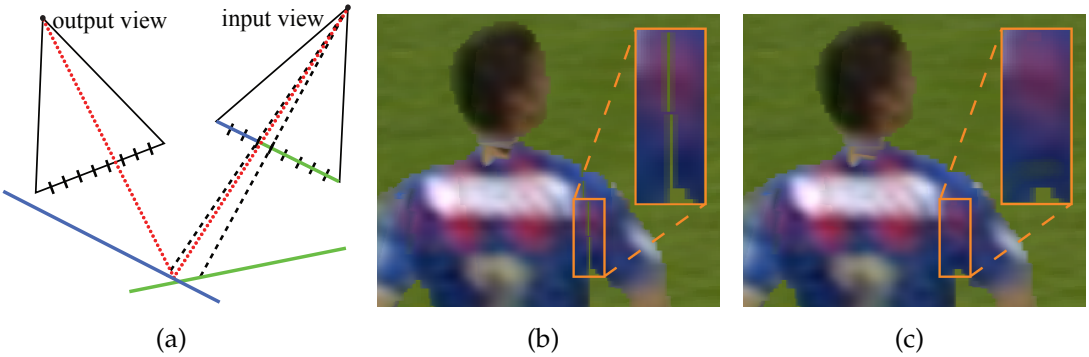


Figure 6.6: *Seam correction. (a) Sampling errors in the segmentation mask cause cracks, e.g., the look-up of a pixel on the blue billboard ends up on the mask of the green billboard from an adjacent billboard fan. (b) Corresponding rendering artifact. (c) Result after our seam correction.*

what happens in figure 6.6(a). Thus a ray trough a pixel might end up on the surface of body part b while the mask lookup still results in body part a. Therefore, the pixel will not be rendered at all, neither when rendering part a nor part b.

To overcome this problem, these *seam pixels* have to be rendered for both adjacent billboards. Therefore, we mark pixels as seam pixels in the masks of the input views if they cover billboards on two adjacent skeleton bones. In the example image, figure 6.6(a), this would be the pixel enclosed by the dashed lines.

In order to mark the seam pixels, we need a method to detect them. This can be done once in a pre-process by traversing the segmentation mask of each input view. A pixel $\boldsymbol{u}$ is marked as seam pixel, if it fulfills both of the following conditions:

- At least one pixel $\boldsymbol{u}'$ in its 4-neighborhood has a different label but comes from the same subject
- $|\text{depth}(\boldsymbol{u}) - \text{depth}(\boldsymbol{u}')| < \varphi$

where $\text{depth}(\cdot)$ is the depth value at this pixel, which can be computed according to the billboard placement. The cracks only appear at connected body parts and not, e.g., between parts that are far from each other in depth or even belong to another subject. Therefore, the threshold φ distinguishes between occluding parts and connected parts. It was empirically set to $\varphi = 3$ cm. An example for the seam corrected segmentation mask and the resulting rendering improvement is shown in figure 6.6(c).

6.6 Rendering

In this section, a photo-realistic rendering method for articulated billboards is presented. Buehler et al. introduced general criterias for an optimal rendering in their paper on the unstructured lumigraph [Buehler et al., 2001]. Our rendering method fulfills *use of geometric proxies*, since we use the articulated billboards as geometric proxy. It also fulfills *unstructured input*, as we do not have any restrictions on the camera placement, as well as the *real-time* criterion, since or method is able to render in real-time. However, for the remainder of the criteria of Buehler et al., it is not always possible to fulfill them in our challenging setup with calibration errors and very sparse camera positioning. Therefore, besides the already mentioned criteria, our particular focus is on the following goals:

- *Coherent Appearance*: Adjacent billboards should intersect without cracks or disturbing artifacts and blend realistically with the environment.
- *Visual Continuity*: Billboards should not suddenly change or pop up when moving the viewpoint.
- *View Interpolation*: When viewing the scene from an original camera angle and position, the rendered view should reproduce that of the input camera.

The *continuity* criteria by Buehler et al. follows directly from the *visual continuity*.

The input to the rendering procedure is the articulated billboard model which consists of the 3D body pose and the segmented input views $\mathcal{I}$ (section 6.4) with the seams computed in section 6.5.3. Again, the calibrations of the source cameras are used too. The billboards are placed as described in section 6.2 such that they are on the axis of the corresponding bone and face their source camera.

For each rendered output frame, the body parts of the articulated billboards are sorted back-to-front for a proper handling of occlusions. In order to meet the above goals, we perform a per-pixel blending procedure. We separate between per-camera weights which are computed once per billboard and the final per-pixel weights. We will describe next the per-camera blending weights in section 6.6.1. They are then used to build the per-pixel weights and do the blending, which will be described in section 6.6.2.

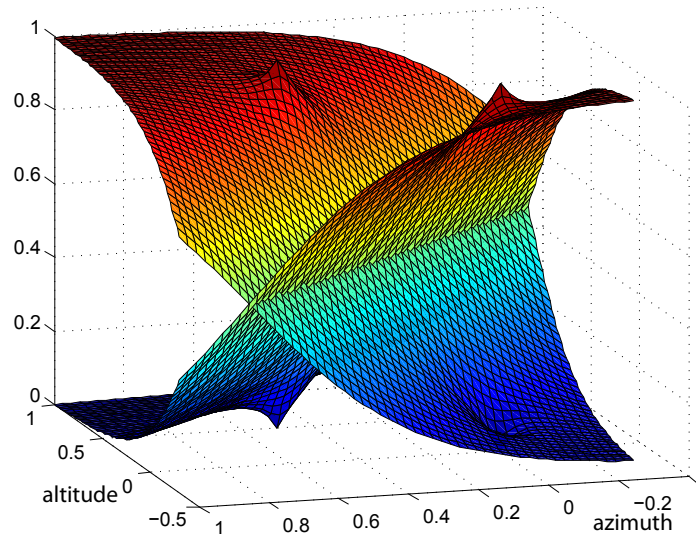


Figure 6.7: *Blending weight example for two cameras. The angles are the spherical coordinates of the view position.*

6.6.1 Per-Camera Blending Weights

Per-camera blending weights are global for all pixels of the same fan of billboards (i.e., body part) and the same camera. We could use for them the same Gaussian weight as in equation 6.4. However, this weight would not guarantee the *view interpolation* criterion. Therefore, we introduce an attenuation function which ensures that all views from an original camera perspective are identical to the corresponding camera source images while still assuming a smooth transition between different views.

The attenuation function is defined as $f(\boldsymbol{I}_{\omega_{Max}}) = 1$ for the source view $\boldsymbol{I}_{\omega_{Max}}$ with the highest value of $\omega(\cdot)$ and

$$f(\boldsymbol{I}_{\omega_{Max}}) = 1 - exp\left(-\frac{d(\boldsymbol{c}_v, \boldsymbol{c}_{\omega_{Max}})^2}{2\sigma^2}\right) \tag{6.6}$$

for all other cameras $\boldsymbol{I}_j$. $d(\boldsymbol{c}_v, \boldsymbol{c}_{\omega_{Max}})$ is the Euclidean distance from the viewers position $\boldsymbol{c}_v$ to the camera position $\boldsymbol{c}_{\omega_{Max}}$ of view $\boldsymbol{I}_{\omega_{Max}}$. The constant σ is empirically determined to be 1 meter, which is lower than the minimal distance between two cameras and thus does not lead to any discontinuities.

6.6.2 Per-Pixel Processing

The billboards of a billboard fan are blended per-pixel. As shown in figure 6.6(a), a camera look-up in the corresponding segmentation mask of each

billboard is performed. This determines if a current output pixel $\boldsymbol{u}$ is on the body part belonging to this billboard. If so, then the corresponding color contribution $V_{\boldsymbol{I}_j}(\boldsymbol{u})$ from source view $\boldsymbol{I}_j$ and its alpha value $\alpha_{\boldsymbol{I}_j}(\boldsymbol{u})$ can be added to the output view V. Otherwise, we set $\alpha_{\boldsymbol{I}_j}(\boldsymbol{u}) = 0$, i.e., transparent. The latter case also occurs when the corresponding body part is occluded in $\boldsymbol{I}_j$ and the color information should be taken from other cameras.

The resulting color value $V(\boldsymbol{u})$ of the screen pixel is then

$$V(\boldsymbol{u}) = \frac{\sum\limits_{\boldsymbol{I}_j \in \mathcal{I}} V_{\boldsymbol{I}_j}(\boldsymbol{u}) w(\boldsymbol{I}_j, \boldsymbol{u})}{\sum\limits_{\boldsymbol{I}_j \in \mathcal{I}} w(\boldsymbol{I}_j, \boldsymbol{u})} \tag{6.7}$$

with the per-pixel weights

$$w(\boldsymbol{I}_j, \boldsymbol{u}) = \alpha_{\boldsymbol{I}_j}(\boldsymbol{u}) \omega(\beta_{\boldsymbol{I}_j}) f(\boldsymbol{I}_{\omega_{Max}}). \tag{6.8}$$

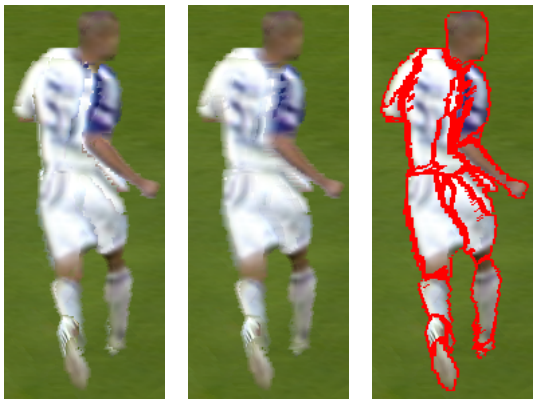
This is done for all color channels separately. The resulting alpha value is

$$\alpha_V(\boldsymbol{u}) = \begin{cases} \alpha_{\boldsymbol{I}_{\omega_{Max}}}(\boldsymbol{u}), & \text{if } w(\boldsymbol{I}_{\omega_{Max}}, \boldsymbol{u}) \neq 0 \\ \frac{\sum\limits_{\boldsymbol{I}_j \in \mathcal{I}} \alpha_{\boldsymbol{I}_j}(\boldsymbol{u}) w(\boldsymbol{I}_j, \boldsymbol{u})}{\sum\limits_{\boldsymbol{I}_j \in \mathcal{I}} \alpha_{\boldsymbol{I}_j}(\boldsymbol{u}) \omega(\beta_{\boldsymbol{I}_j})}, & \text{otherwise} \end{cases} \tag{6.9}$$

where the first case applies, if the closest camera is used for this pixel. Equation 6.7 and equation 6.9 make sure that the color values are blended such that the factors sum up to 1. However, the alpha values do not have to sum up to 1, e.g., if continuous alpha mattes are available instead of binary segmentation masks.

In addition to this, billboards seen at an oblique angle or from the backside, i.e., having a normal in an angle close to or more than 90 degrees away from the viewing direction, are simply faded out. For simplification, these factors are not shown in the equations.

An example for blending of intensities (i.e., one color channel) of two cameras is shown in figure 6.7 where the azimuth and altitude angles are from spherical coordinates of the view position around the fan of billboards. The two peak points at $(0.0, 0.0)$ and $(0.5, 0.5)$ correspond to the positions of the source cameras. As it can be seen in the plot, when approaching these points the corresponding camera's weight increases to 1.0 and all other camera weights decrease to 0.0. Therefore, in this case only the source camera is used which results in the exact reproduction of the source image. However, the functions remain smooth and thus preserve the visual continuity.



(a) (b) (c)

Figure 6.8: *Smoothing. (b) Rendering without smoothing. (c) Adaptive smoothing enabled. (d) Marked discontinuities where smoothing has been applied.*

Smoothing

Finally, to prevent non smooth edges at the boundaries of a fan of billboards with respect to the background, other billboard fans, and at locations where other input views receive the highest weight (e.g., due to occlusions on a billboard), an additional Gaussian smoothing step is applied. This is done adaptively as a post-process only at discontinuities detected and stored while rendering the billboards. A pixel $\boldsymbol{u}$ of the target view is marked as discontinuity if for at least one of its four direct neighbors, called $\boldsymbol{v}$ the following holds:

- $\boldsymbol{v}$ belongs to the same subject as $\boldsymbol{u}$
- $\boldsymbol{v}$ belongs to a different body part as $\boldsymbol{u}$
- The depth values of $\boldsymbol{v}$ and $\boldsymbol{u}$ and differ more than 3cm

The Gaussian smoothing, applied at such a detected pixel $\boldsymbol{u}$, consists of a simple convolution with a 5×5 Gaussian kernel. Figure 6.8 shows an example of the detection of discontinuities and the result of the adaptive smoothing.

6.7 GPU Implementation

The planar representation of billboards and thus also of articulated billboards has the advantage, that the memory to store the geometry is small what makes the rendering process simpler and faster. Besides low memory consumption,

articulated billboards have also the advantage that the rendering can be implemented efficiently since fragment shaders can be used optimally for the per-pixel blending.

The camera image as well as the corresponding segmentation mask can be stored on two textures per camera. We use the three color channels of the mask textures to store the subject ID, the billboard ID (i.e., the body part) and whether it is a seam pixel or not.

The blending process then computes the camera weights and passes them to the fragment shader. The fragment shader computes the final per-pixel weight and does the actual blending.

Finally, the adaptive smoothing is implemented using multiple render targets such that the locations which require smoothing can be stored in the same render pass. In a second pass, this information is used for the smoothing. This results in two render passes for the entire rendering of a novel view.

6.8 Results

We applied our method to footage of real scenes of a soccer game, captured by TV cameras, which were used to broadcast the game in HD resolution (1920×1080, interlaced). In all renderings, the background (pitch, stadium) was simply represented as large planes using projective textures blended on top of them.

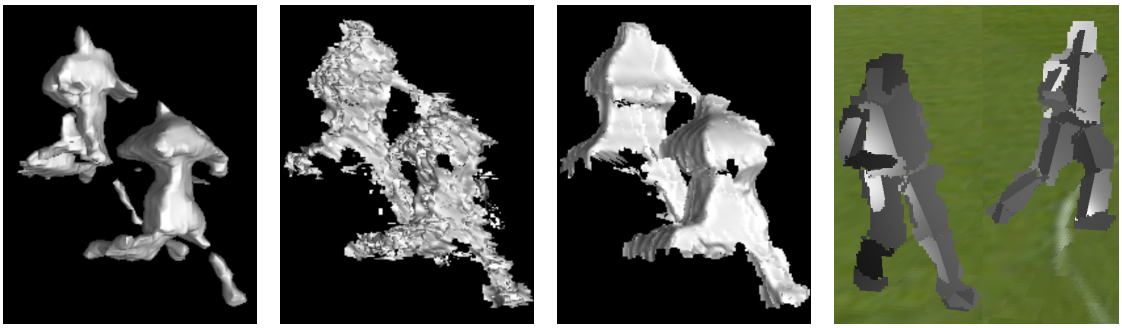
Figure 6.9 shows a direct comparison of our articulated billboards to standard billboard rendering with one billboard for each player and input camera. Whereas simple billboarding suffers from duplications of arms and legs, our method keeps the 3D perception, e.g., self-occlusions, correct due to the adaptive geometry. Even in the worst case, i.e., the view from a position in the middle of two source cameras, the overall body pose is preserved. In figure 6.10 a bird's eye view is depicted. It shows how standard billboarding simply tilts the large billboards, whereas articulated billboards provide a realistic rendering of the entire pose.

Figure 6.12 shows a comparison of our rendering to ground truth data using a leave-one-out test, which shows that our method is able to realistically reproduce the visual appearance of a scene from only two input views even at distant novel viewing positions.

A qualitative comparison of the shape representation with articulated billboards to visual hulls, stereo reconstruction, and the method by Guillemaut et al. [2009] is shown in figure 6.11. Due to its articulated structure with a

Figure 6.9: *Direct comparison for views in the middle of two source cameras. Each player is rendered with a standard billboard technique and with articulated billboards. The standard billboards exhibit considerable ghosting artifacts.*

Figure 6.10: *An example of a bird's eye perspective. While standard billboards are simply tilted (left), articulated billboards give an impression of the actual 3D pose.*



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

Figure 6.11: *Qualitative comparison. (a) Visual hulls cannot capture geometric detail and are sensitive to camera calibration errors. (b) Stereo reconstruction is problematic due to low texture resolution and noise. (c) The method of Guillemaut et al. [2009] improves the silhouette segmentation considerably. However, the shape reconstruction is still rather inaccurate, leading to ghosting artifacts. (d) The articulated billboard reconstruction (for a similar scene) captures the geometry much more faithfully. (a)-(c) Courtesy of Guillemaut et al. [2009]*

planar geometric proxy for each limb and input view, our method generally provides a better geometrical approximation of the subject's shape, in particular for challenging and inaccurate input data as in our application setting. The benefit is an improved rendering quality with less ghosting artifacts.

Figure 6.13 shows virtual views of two different scenes. Due to our articulated billboard structure, the 3D poses of the players remain consistent even for extreme viewpoint changes and the viewer gets a clear impression of the actual positions of different body parts.

The scenes can be rendered for free-viewpoint video with our system in real-time at HD resolution ($>$ 40 fps). It is also possible to render scenes as a video replay from the virtual viewing positions, i.e., render sequences from novel viewpoints. To our knowledge this has not been shown before in a similarly challenging application scenario with only two cameras.

Optionally, the automatic segmentation of body parts can be manually corrected in ambiguous regions. We performed slight corrections in the single-frame examples for maximum quality. However, such manual corrections of the segmentation were not done for the dynamic scenes with video replay but they achieve a similar result.

The timing for automatic processing of our system is generally a matter of seconds. Since the body pose estimation from chapter 5 only takes about 5 to

8 seconds per subject and frame, the algorithm is mostly determined by the result-based joint position optimization, which is the most expensive step. It takes maximally 30 seconds per subject for three cameras, which is acceptable for post-production in commercial systems.

6.9 Discussion and Outlook

In this chapter we presented a novel representation and rendering method for human bodies suited for challenging uncontrolled outdoor setups. Our articulated billboards provide an improved geometric shape approximation for challenging acquisition conditions, where methods based on accurate silhouettes, stereo correspondences, or calibration generally fail. They combine the robustness of billboard techniques with the articulated pose representation of template based or visual hull based methods.

Together with the pose estimation from chapter 5, the model computation provides a practical solution even in challenging setups, and our pixel-accurate processing results in high quality renderings with a realistic reproduction of the subject's appearance for novel views. Self occlusions and parallax of the articulated body are preserved and thus give a correct 3D impression of the body pose. We have shown results in a quality comparable to the source images from HD-TV broadcast cameras. With their simple representation, articulated billboards can be rendered highly efficiently and thus will be applicable even for mobile devices.

Designed for low quality input data recorded with large base-lines, articulated billboards are an optimal approximation if the player has a height up to about 200 pixels in the original as well as in the rendered image. However, if higher resolutions or dense camera setups are available, more complex primitives should be used. The reason for this is that at higher resolutions ghosting artifacts within a single billboard fan can appear. An extreme situation would be a closeup of only a face. Since the face is represented as a single billboard fan, its non-planarity would cause ghostings.

The view range, i.e., possible positions and directions of novel views is naturally limited to the vicinity of the source camera. This means that, e.g., it is not possible to show a subject from the back if this subject is only covered in the input cameras from front views. Nevertheless, as shown in figure 6.12, our method features a quite large viewing range even from only two input cameras. Most notably, it allows views from not only the direct connection line of these two cameras but from the entire vicinity of the input cameras.

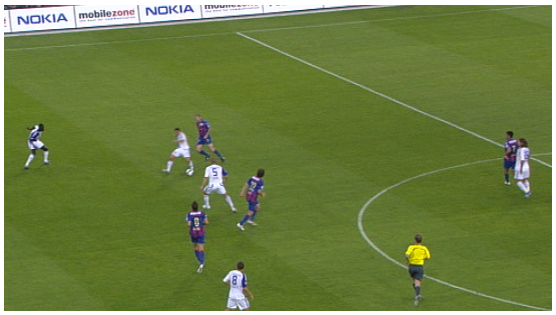
During rendering, occlusions are only a problem at pixels which are not visible in any of the cameras. For these cases we plan to apply a hole filling algorithm. Another idea would be to use texture information of previous (or subsequent) frames, where this body part was not occluded, similar to the use of geometry information in the work by Sand et al. [2003]. For every body part, a cache of previous textures could be stored. If this body part is occluded in a camera view, then a cached texture could be used instead of the current frames camera texture.

Due to our current depth sorting according to the distance to the bone, flickering artifacts can appear if a view is selected where the sorting changes. Therefore, we will investigate the computation of a per-pixel depth based on the billboard planes.

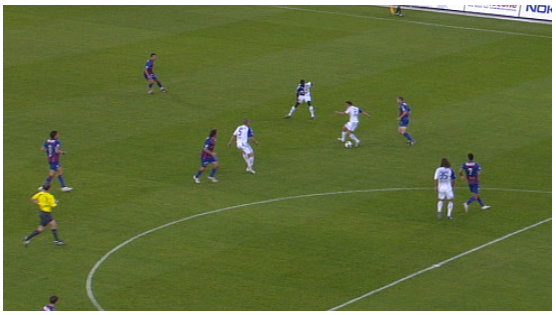
In order to improve temporal coherence, we plan to investigate global optimization of the billboard positions over all video frames. This could also be done for the segmentations.

Furthermore, for the color comparison of the optimization, we would like to investigate on other methods to evaluate matches. A first step would be more advanced color space to remove lighting changes.

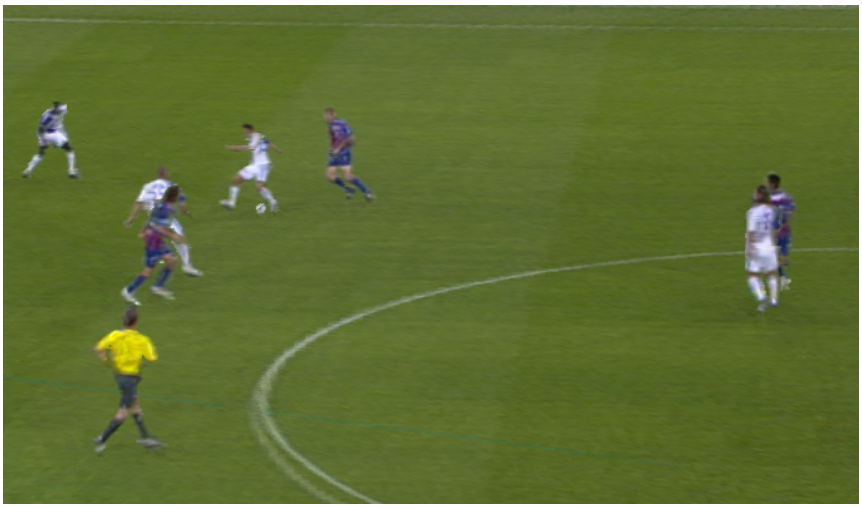
Finally, an interesting improvement could be to add semi-transparent parts at the silhouette borders to allow a fade-out at depth discontinuities and reduce the aliasing effect. This is possible since the equations for the blending allow also for non-binary alpha values.



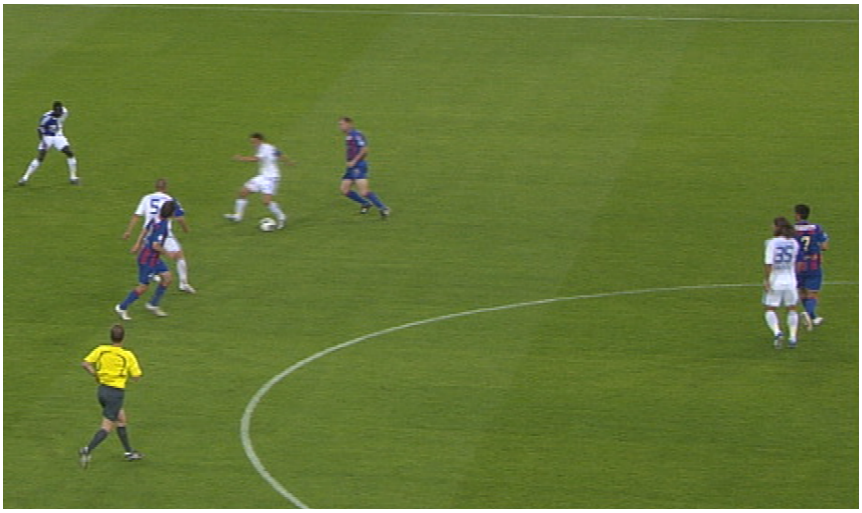
(a)



(b)



(c)



(d)

Figure 6.12: *Leave-one-out example. (a) and (b) are two wide-baseline input views. (c) From these two input views we computed a virtual view of the scene from a novel viewing position. (d) Ground-truth view of an actual camera at this position. Note that the ghosting on the ground plane in (c) stems from the camera calibration.*

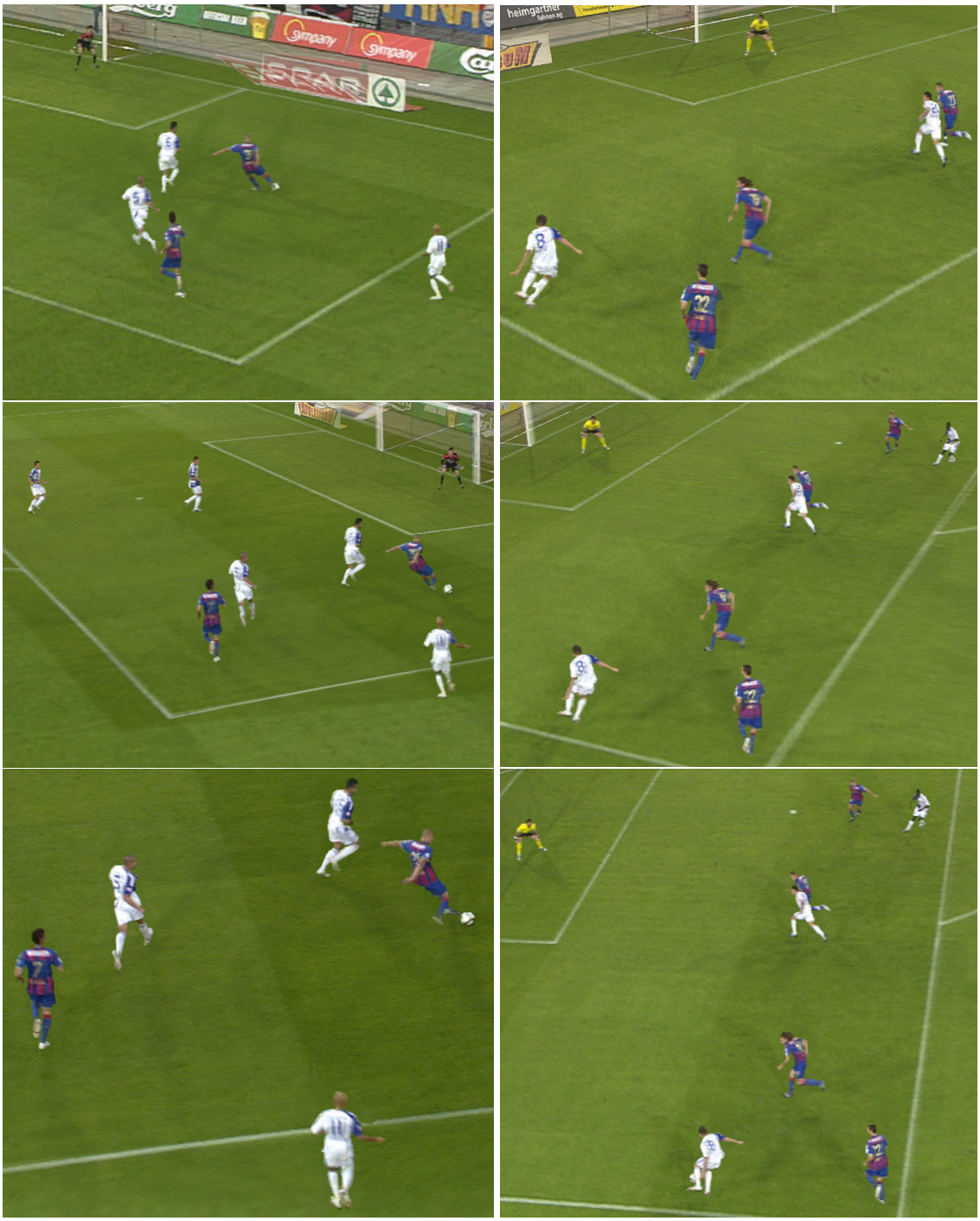


Figure 6.13: *Novel views of a soccer game. The 3D pose and realism of the players is preserved for strongly varying viewpoints, e.g., the arms and the legs within the rotation.*

CHAPTER 7

Adaptive View-Dependent Geometry

In this chapter we present a novel approach for 3D reconstruction and view synthesis of non-studio setups. It can be seen as an alternative to the novel-view synthesis that was presented in the last two chapters, i.e. the articulated billboards (chapter 6) together with their reconstruction based on our body pose estimation (chapter 5).

As seen in chapter 3, there are only very few fully automatic methods that reliably reconstruct the human body in challenging outdoor scenes as well as allow for renderings from novel viewpoints. Either they only interpolate between two views and thus only allow flights along the direct path between two cameras or they suffer from the calibration errors. Billboarding is able to achieve arbitrary views but causes ghosting artifacts. The articulated billboards presented in chapter 6 reduce the artifacts to a minimum. However, even with the body pose estimation presented in chapter 5 they still need manual interaction in many frames. This is much less than, e.g., manually correcting the pose of the previous frame to the current frames input, where you would have to adjust almost every joint in every frame and camera. In our pose estimation only flips have to be selected by the user, which occur roughly every 10 frames per camera and player. This makes it possible to render high quality novel views for one frame or short sequences. But for longer sequences, also this manual interaction is too cumbersome and would take a lot of time.

The method for novel-view synthesis presented in this chapter is a fully automatic algorithm and thus allows also for longer sequences. Our method consists of an adaptive reconstruction technique and a view dependent rendering. It is designed for outdoor sports games and remains robust to calibration errors and low resolution of the players. It allows to reconstruct and render scenes from as few as two cameras, even with wide baselines.

The first part, the reconstruction is a coarse-to-fine method with triangles as primitives. For every camera a 2.5D reconstruction is computed. It starts with large triangles that represent the camera image and computes their optimal depth according to feature points. Only triangles that do not fit accurately enough the real surface are subdivided, and the process is iterated and finished by a final refinement according to back-projections. The 2.5D reconstructions of all cameras are then simply merged into one triangle soup. The rendering uses a force field that approximates the calibration errors to deform the geometry according to the viewpoint such that it perfectly fits the original camera views. This view-dependent geometry interpolation as well as a view-dependent texture blending leads to convincing novel view synthesis results even in the challenging video setups of conventional TV broadcasts.

The idea of subdividing the space into small planar patches is similar to microfacets [Yamazaki et al., 2002; Goldlücke and Magnor, 2003]. In contrast to microfacets our method adaptively refines the geometry only where it is required and possible. Therefore, we require less feature matches and the refinement can inherit geometric information throughout the subdivision. Additionally, our triangles are oriented according to the geometry and not to the viewer.

The work described in this chapter is accepted for publication and will be presented in Germann et al. [2012].

7.1 Overview

Figure 7.1 shows the entire algorithm in an overview schema. The details are only shown for camera 1 as base camera, but the same steps (a)-(d) are also done for camera 2.

The main part of the reconstruction method is an adaptive top-down technique that is able to retrieve a 2.5D triangulation of the players in each camera. We call the selected camera the *base camera*. The reconstruction starts with a simple triangulation in the base camera (figure 7.1(a)). The triangles are placed according to a sparse 3D point cloud (figure 7.1(b)), which is generated

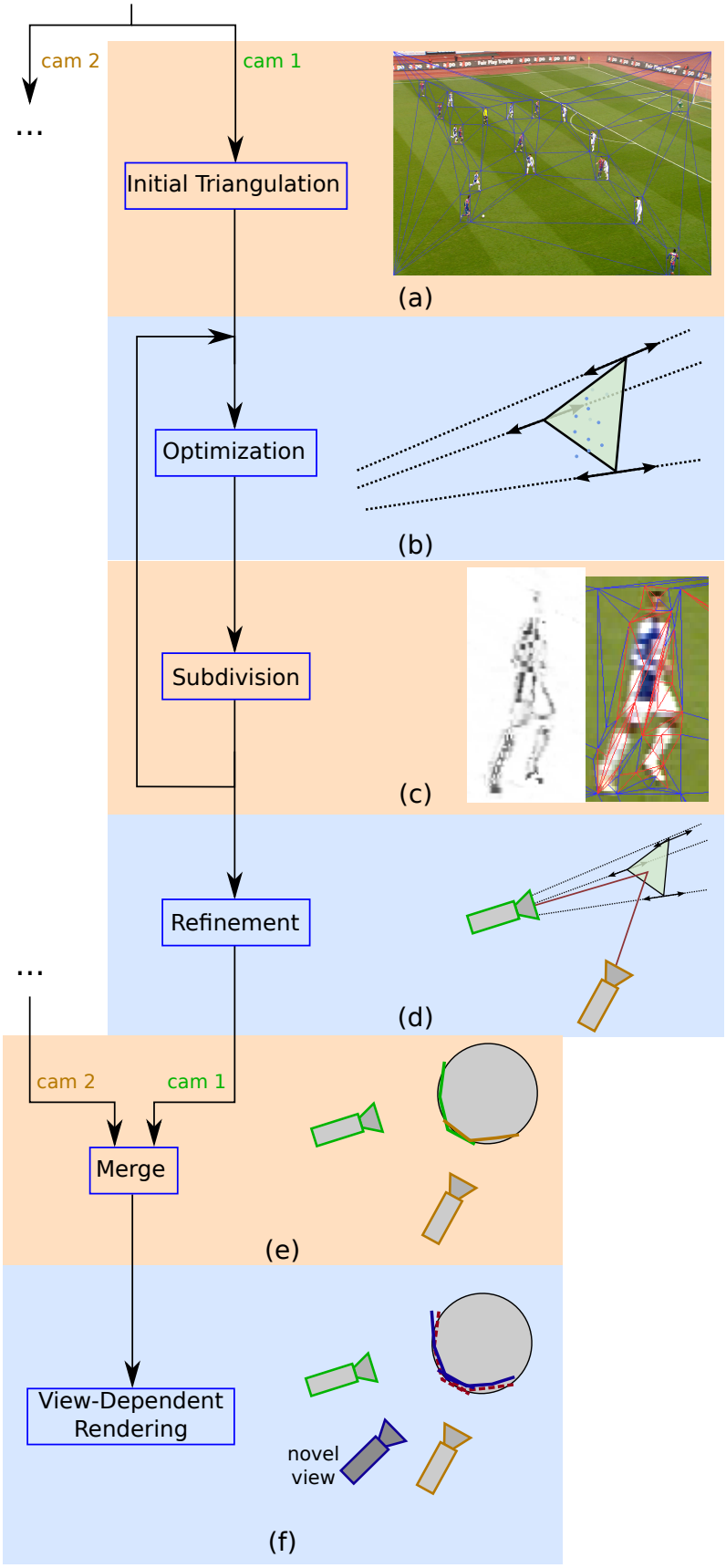


Figure 7.1: *Overview of the algorithm: (a)-(e) Adaptive reconstruction. (f) View-dependent rendering.*

using an extended version of the DAISY features [Tola et al., 2008]. If the triangles are too large and do not represent the shape of the object accurately (red triangles in figure 7.1(c)), they are subdivided. This process is repeated until the triangles are just small enough to approximate the shape of the object. This way the reconstruction method inherently adapts to the geometric complexity as well as to the resolution of the represented object. In an additional refinement step (figure 7.1(e)) the vertex depths of those triangles that do not contain any features are set to optimal depth values of neighboring triangles including random perturbation tests. The adaptive level of detail and the reconstruction lead to a robust geometry. To cover also parts of the scene that are occluded in one camera, we repeat the reconstruction for each camera as base camera and merge these 2.5D reconstructions into a final 3D geometry (figure 7.1(e)).

The resulting geometry can be rendered from an arbitrary viewpoint by simply drawing all triangles and blending the color information from the available cameras using projective texturing. However, inherent calibration errors will yield rendering artifacts such as ghosting. Therefore, we propose a method that morphs the geometry based on the viewing positions to compensate for calibration errors (figure 7.1(f)).

In section 7.2 the adaptive reconstruction is elaborated in detail. The view-dependent geometry morph and rendering is described in section 7.3. Implementation details can be found in section 7.4. Our proposed method is evaluated and discussed in section 7.5 and section 7.6.

7.2 Adaptive Reconstruction

We chose to represent the geometry as a per camera triangle soup. The advantage of a triangle soup as opposed to a connected mesh is that it allows to place these triangles independently solving implicitly for depth discontinuities. Our reconstruction algorithm proceeds as follows (Figure 7.1):

1. *Initial Triangulation:* an initial set of triangles is created from the image of one of the cameras. The triangulation of this base camera is illustrated in figure 7.1(a). The triangles are aligned with the view of the base camera such that the only degree of freedom is the depth of its vertices in camera coordinates. This allows for a low-degree of freedom optimization that facilitates the process of positioning them in 3D, but without sacrificing precision.

2. *Triangle Optimization:* in this step each triangle of the current set is positioned independently in 3D by optimizing the depth of its

vertices only. As a result the projection of the triangle onto the base camera never changes. The optimization process uses robust sparse feature correspondences from pixels inside the projection of the triangle to determine the depth of each vertex.

3. *Subdivision:* the 3D triangles may not approximate well the local geometry. For instance, in figure 7.1(a) the players are initially approximated only by two triangles. In this case we subdivide the triangle if the error evaluated by a robust error metric of texture re-projection is too big. Figure 7.1(c) shows triangles to be subdivided in red. We repeat the optimization and subdivision step until the re-projection error is small for all triangles.

4. *Refinement:* due to occlusions and the low resolution of the image, it is not always possible to find image features in every triangle. If a triangle has no features it inherits its vertex depths from the previous subdivision. However, these depths could be wrong and as a result we might have rendering artifacts such as missing parts of the players. To further refine the position of such triangles, we employ a heuristic to determine their 3D location based on depth guesses from the neighboring triangles combined with random perturbations on these depths.

5. *Merge:* We reconstruct a 2.5D geometry the same way for every input camera as base camera. The union of these 2.5D geometries results in a final 3D reconstruction (figure 7.1(e)).

At the end, a smoothing is done, to avoid discontinuities on connected surface parts. This is achieved by simply connecting the triangles that are close together. More specifically, if two or more triangles share a vertex in the 2D triangulation and their depth values at this vertex are very similar, then all of them are set to the average value of these depths.

The following subsections present the above listed steps of the reconstruction algorithm in detail.

7.2.1 Inital Triangulation

To start the algorithm, an initial triangulation in 2D in the base camera is needed. The simplest approach would be to use two large triangles with the camera image corners as vertices. They would be fitted to the feature points, subdivided, and their children iteratively processed. However, this would involve already many subdivision steps to adapt to the players. Therefore,

we choose to use detected bounding boxes to start. We use a delaunay triangulation of the four corners of the image and the corners of the bounding boxes of each player (Figure 7.1(a)). The bounding boxes of each player can be detected automatically according to Fleuret et al. [2008]. Note that we do not rely on an accurate bounding box detection. As one can see in figure 7.1(a), players who are occluding each other are generally classified into one box. Also the goal keeper's box is shifted. This does not create any problems for the algorithm since it is only used as an initial triangulation, i.e. to speed up the process. This procedure gives the initial set of 2D triangles in a base camera.

In addition to this, an initial depth for the vertices of the triangles is required. It is computed by projecting them onto the ground plane. The intersection of the vertices with the ground plane gives their initial 3D positions and thus the depth values. The ground plane is given by the calibration (chapter 4.5).

7.2.2 Triangle Optimization

In every iteration step, the vertex depths of the triangles, and thus their 3D positions along a ray from the base camera, are optimized. This is done for each triangle independently of the other triangles. Since only the depth values of the vertices are optimized, we have only 3 degrees of freedom per triangle. This reduces the search space and allows a fast processing - especially when comparing to bottom-up approaches where a depth for every pixel has to be computed. Another advantage of optimizing only depth values is that it prevents triangles to shift over each other in the base camera and thus assures that all triangles are completely visible in their corresponding base camera. An illustration of a triangle with the rays from the base camera can be seen in Figure 7.1(b).

To find the three depth values that position the triangle along the surface in the scene, we developed an optimization technique that combines two criteria: one is based on a background color model and one based on robust feature matches.

Background Color Model Criterion

In a first criterion, foreground and background triangles are differentiated according to color models. Triangles are part of the background if more than 95% of the pixels are classified as such by the background color model. We use a simple color model, but basically any background subtraction could

be used here. A more elaborated background subtraction technique for this purpose is described by Zach et al. [2007].

If a triangle is classified as background, its vertices are pushed to the ground, i.e., they receive the depth values of the intersection with the ground plane. These background triangles are not optimized any further in this step of the iteration and the second criteria is simply ignored for them.

Feature Match Criterion

The second criterion uses feature matches to determine the optimal depth values for the vertices of non-background triangles. In a first step, a robust set of matching pairs is computed as described in the next subsection. Only pairs that have one feature point in the base camera are used. Each of these 2D matching pairs represent a point in 3D that can be computed by triangulation (see section 4.5.1). Due to calibration errors the rays of the triangulation usually do not intersect. In such cases the point on the base camera's ray that is closest to the other ray is used, as explained in section 7.3. This results in a 3D point cloud, that belongs to the scene geometry of the base camera. According to the triangulation in the base camera, each of the feature points is assigned to exactly one triangle. Thus, for every triangle a set of feature points can be derived. If this set contains 3 or more points, the triangle can be placed according to the points in the set, otherwise it is left unchanged. The placing according to the feature points consists of two steps: First, by applying a RANSAC technique [Fischler and Bolles, 1981] a plane is fitted accurately and robustly to the set of 3D points. Once the best fitting plane is determined, in a second step the rays from the camera through the vertex points of the triangle are intersected with this plane. The intersections directly give the 3D vertex points and thus the depth values.

Feature Match Computation

The above described method to position a triangle in 3D requires reliable image feature matches between the views. For this purpose we selected the DAISY features as presented by Tola et al. [2008]. These features are designed for wide-baselines and therefore most suitable for our setup. We described them already in section 4.3.

However, due to the low resolution of each player and the lack of features on the pitch, the feature detection contains a lot of wrongly matched pairs. Therefore, we added more constraints to the matching operator to get more reliable, albeit fewer, matches. We find robust matches in three steps:

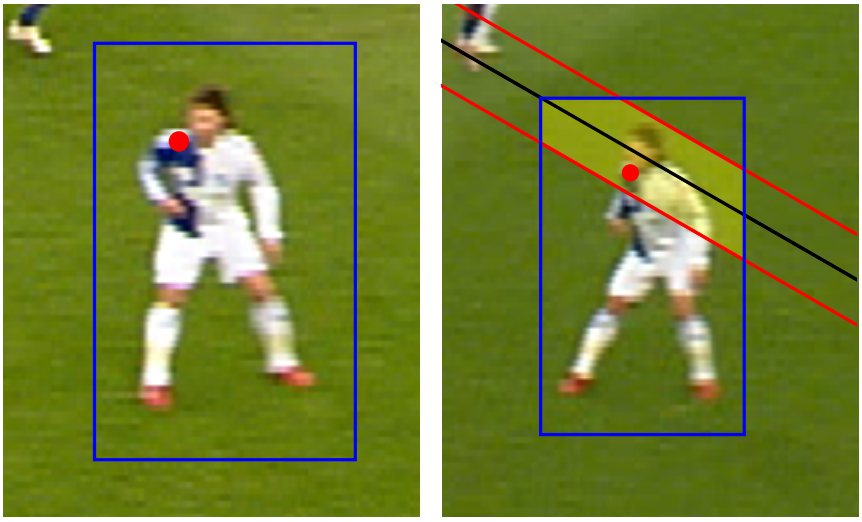


Figure 7.2: *Feature correspondence search for a pixel in the left image within a cropped epipolar stripe in the image of the other camera.*

1. For every pixel in the base camera we restrict the search space of possible matches in the second camera to the *epipolar stripe*. We define the epipolar stripe as a band of $d = 20$ pixels around the epipolar line, whereof only pixels lying inside the bounding box are considered. This is illustrated as the yellow area in figure 7.2.
2. Only matches with a DAISY error value below 1.4 and below 0.8 times the average of the matches within this stripe are considered.
3. We verify that the match is symmetric. That is, if $\boldsymbol{u}_{c_0}$ is a pixel in the base camera and $\boldsymbol{u}_{c_1}$ is its corresponding DAISY match in the second camera, the DAISY match of pixel $\boldsymbol{u}_{c_1}$ has to lie within five pixels of $\boldsymbol{u}_{c_0}$.

The use of the epipolar stripe instead of just the epipolar line is important as it copes with calibration errors.

The resulting matches are a relatively sparse set of reliable matches and they are used in the triangle optimization process. Depending on the scene, the number of feature matches per player varies from 20 to 300.

7.2.3 Adaptive Subdivision

For planar parts of the scene, the geometry can be approximated by large triangles. However, if the scene geometry represented by a triangle significantly differs from a planar surface, the approximation is too coarse. In this case,

the triangle is subdivided and the positioning, i.e. the depth optimization is repeated for the children. To evaluate, if a triangle has to be subdivided, the back-projection error is computed. If the projective textures of a triangle from the respective source cameras do not match, then the triangle is subdivided.

The subdivision step is iterated with the position optimization. The repetitions end if either the number of iterations is larger than a given number or if the triangles are smaller than the image pixels in the base camera. The resulting triangle soup only features small triangles where the underlying 3D geometry requires a refinement.

In the following, we will first describe the mentioned error metric, which is based on back-projection errors. Second, we will describe the actual process of subdividing those triangles that were selected by the error metric.

Error metric

For every triangle a back-projection error value is computed to decide if the triangle has to be subdivided or not. To do so, the current geometry is used to reproject the textures from all cameras into the base camera. The error of a single triangle is then computed as the average of its pixel errors. For every pixel $\boldsymbol{u}$ in the base camera c_b, the pixel error $e_{\boldsymbol{u}}$ is defined as

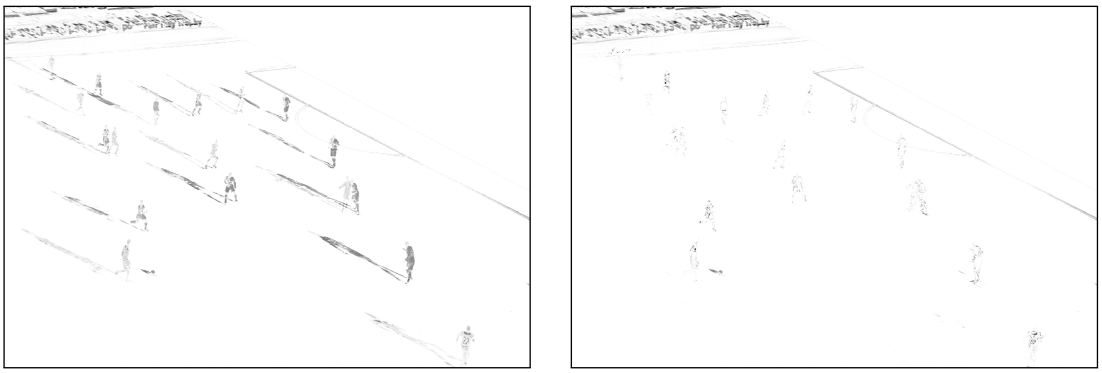
$$e_{\boldsymbol{u}} = \frac{1}{|C(\boldsymbol{u})|} \sum_{c \in C(\boldsymbol{u})} e_{\boldsymbol{u}}(c), \tag{7.1}$$

where $C(\boldsymbol{u})$ is the set of all non-base cameras where the back-projection of $\boldsymbol{u}$ is not occluded, i.e. where the surface point belonging to $\boldsymbol{u}$ is visible. The per-camera pixel error $e_{\boldsymbol{u}}(c)$ between the base camera and a given camera c at pixel $\boldsymbol{u}$ is defined as the color difference (in RGB space) between the values of $\boldsymbol{u}$ in the base camera and the projected point in camera c.

To have comparable image sources, the appearances of the images are roughly adjusted by adding a compensating color shift as described in section 7.3.

Inaccurate calibration will inherently introduce a bias in this error metric as the projection of a 3D point into the cameras suffers from calibration errors. We address this problem by adding a geometric shift to each triangle vertex. This view-dependent geometry is described in detail in section 7.3.

An example of the initial per pixel error viewed from the base camera is shown in Figure 7.3(a) where the entire geometry is approximated as the ground plane. The error after the reconstruction is shown in figure 7.3(b). Please note that the areas which look like shadows of the players are not



(a) (b)

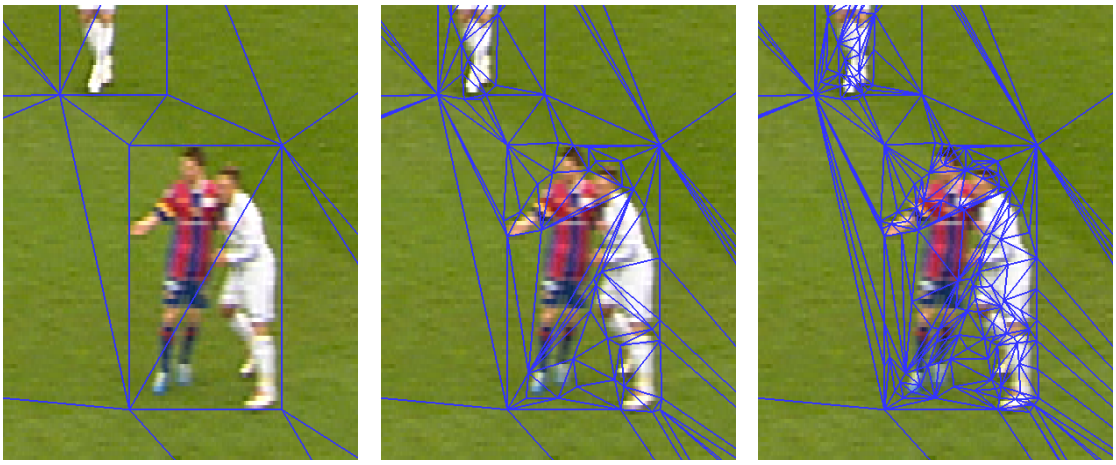
Figure 7.3: *(a) Example for an initial color error when using the ground plane as the geometric proxy. The errors appear at pixels with wrong depth values. (b) The error after the reconstruction. Both images are enhanced in contrast and brightness to better visualize the error.*

their shadows. These areas show the error from the ghost image of the other camera (see Taneja et al. [2010]). After the reconstruction, they are also reduced to a minimum.

Subdivision

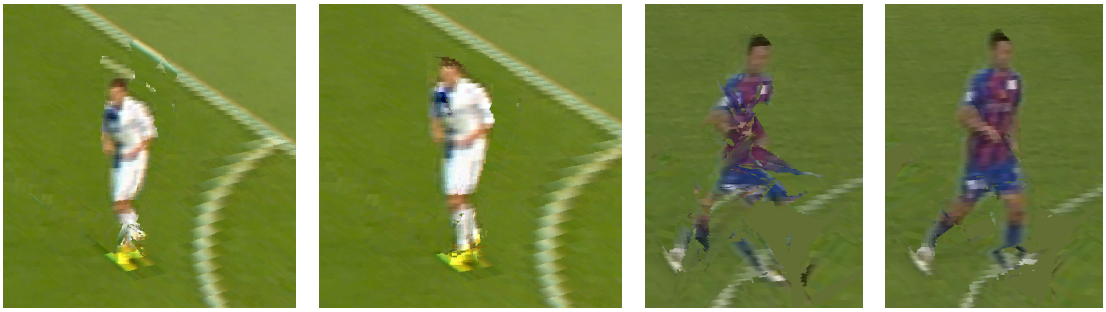
The subdivision is a simple splitting of the triangle into two halves at the longest edge, including splitting of neighbors sharing the same edge. This directly inherits the position in 3D to the children. Therefore, even if no feature point is available for a child (e.g. at occlusions), it still inherits a most plausible 3D position and orientation.

In the first iteration step, we use the background color model and a blob detection to get contour lines. These contour lines guide the first subdivision of the initial triangulation as shown in figure 7.4(b). This speeds up the reconstruction and avoids high triangle counts, since it adds edges at potential depth discontinuities. In all examples we therefore used only at most 3 subdivision steps. One might argue, that these contour lines could already be used for the initial triangulation and not only in the first subdivision step. However, this would cause the initial triangulation to consist of already very small triangles with many of them having no feature points at all. If such a triangle is part of a human body, it would then stay on the ground and not profit from nearby feature points.



(a) (b) (c)

Figure 7.4: *(a) Initial triangulation. (b) First subdivision based on the silhouettes of the player. (c) Final subdivided geometry.*



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

Figure 7.5: *Two examples show the improvement of the result before and after applying the refinement step.*

Figure 7.4(c) shows an example for the final triangulation resulting after 3 iteration steps of optimization and subdivision.

7.2.4 Refinement

As already mentioned, there are triangles that have no or not enough DAISY features to compute their position. These triangles receive reasonable depth values for their vertices by inheritance in the subdivision. However, for a small subset of triangles, this inherited positioning can be wrong. This may lead to rendering artifacts as shown in figure 7.5(a) and figure 7.5(c).

To resolve these, we assume that for a triangle usually those triangles that

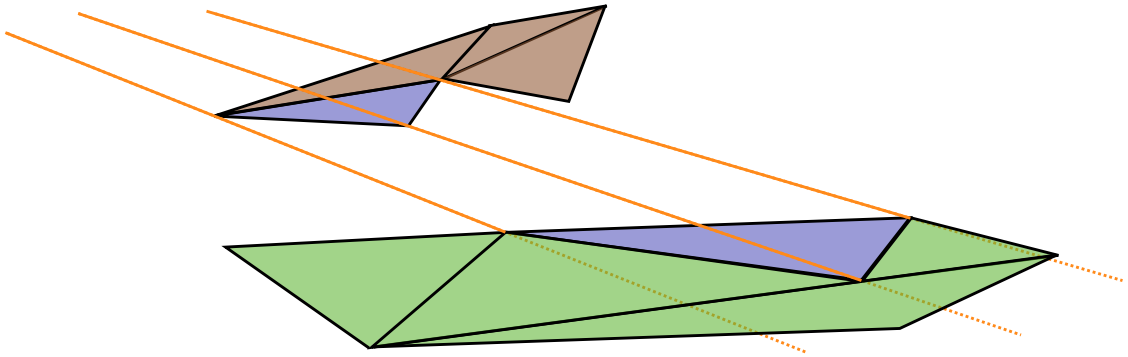


Figure 7.6: *Schematic view of the refinement idea: the blue triangle is difficult to position due to lack of features. Neighboring triangles are used to infer its 3D position. Shown are two possible positions of the blue triangle.*

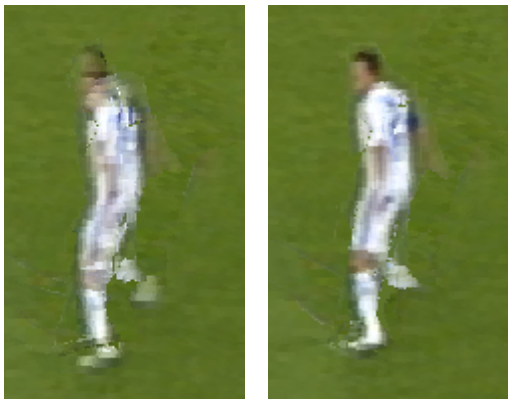
are adjacent in the base camera have similar 3D depths values. A schematic example is depict in figure 7.6. The orange lines are the rays of the triangle vertices from the base camera. The triangle could be part of the ground (green triangles) but falsely inherited the depth values of parts of the player (brown triangles). In the refinement steps both positions would be tested (shown as the two blue triangles). Additionally also depth values around these two positions would be tested, with Gaussian distributed random depth perturbation independent on each vertex. From all these possible positions, the one is selected which minimizes the back-projection error of the triangle.

More specifically, for every triangle o, the set $\mathcal{W}_o$ of triangles that are adjacent to o in the base camera (in 2D) is built. For every triangle $o' \in \mathcal{W}_o$ the vertex rays of o are intersected with the plane defined by o'. The intersection results in possible depth values for o, for which the back-projection error is evaluated. Additionally, also Gaussian distributed random values around these depth values are tested. From all these tests against all triangles in $\mathcal{W}_o$ the depth values with the lowest error are set for o.

Similar to the generalized PatchMatch method [Barnes et al., 2010] this is done iteratively over all triangles 10 times with 10 random perturbations per vertex and iteration. Figure 7.5 shows the improvement of this step.

7.3 View-dependent Geometry and Rendering

Our representation could be rendered as simple triangles in 3D with the color values from projective texturing of the source camera images, using some smart blending function, e.g. the one presented by Buehler et al. [2001]. Due to the inherent errors in the calibration process, these projections of the source



(a) (b)

Figure 7.7: *(a) Rendering with fixed geometry and projective texturing. (b) Rendering with view-dependent geometry and texturing.*

cameras do not match to each other - even on a perfectly reconstructed geometry. These projection errors result in rendering artifacts such as blurriness and ghosting. For the same reason, the geometry projected into the camera views is usually not consistant, such that wrong parallax shifts appear. An example for a resulting rendering from a view between the two source cameras is shown in figure 7.7(a).

7.3.1 3D Geometry Morph

To address the problem described above, our triangles are not fixed in space. The location of the three vertices change according to the position and direction of the artificial viewer camera. We will analyze and introduce our view-dependent geometry and rendering method using the illustration in figure 7.8. The blue dots in the two camera views are reliable feature matches from one camera to the other. If the calibration is correct then the corresponding rays (the red lines are an example for one match) would intersect. This is generally not the case due to the calibration errors. Wherever we position the 3D point, in at least one camera it will not project back to the 2D position where the feature point was detected. To solve this, we shift (morph) this point in 3D along the line of the shortest path between the two rays (the red arrow) when changing the viewing position. We call this shortest path a displacement. These displacements describe the geometric 3D morph function from every feature point in the base camera to the corresponding feature point in the other camera. In the figure, we illustrated this also for the other feature points by arrows.

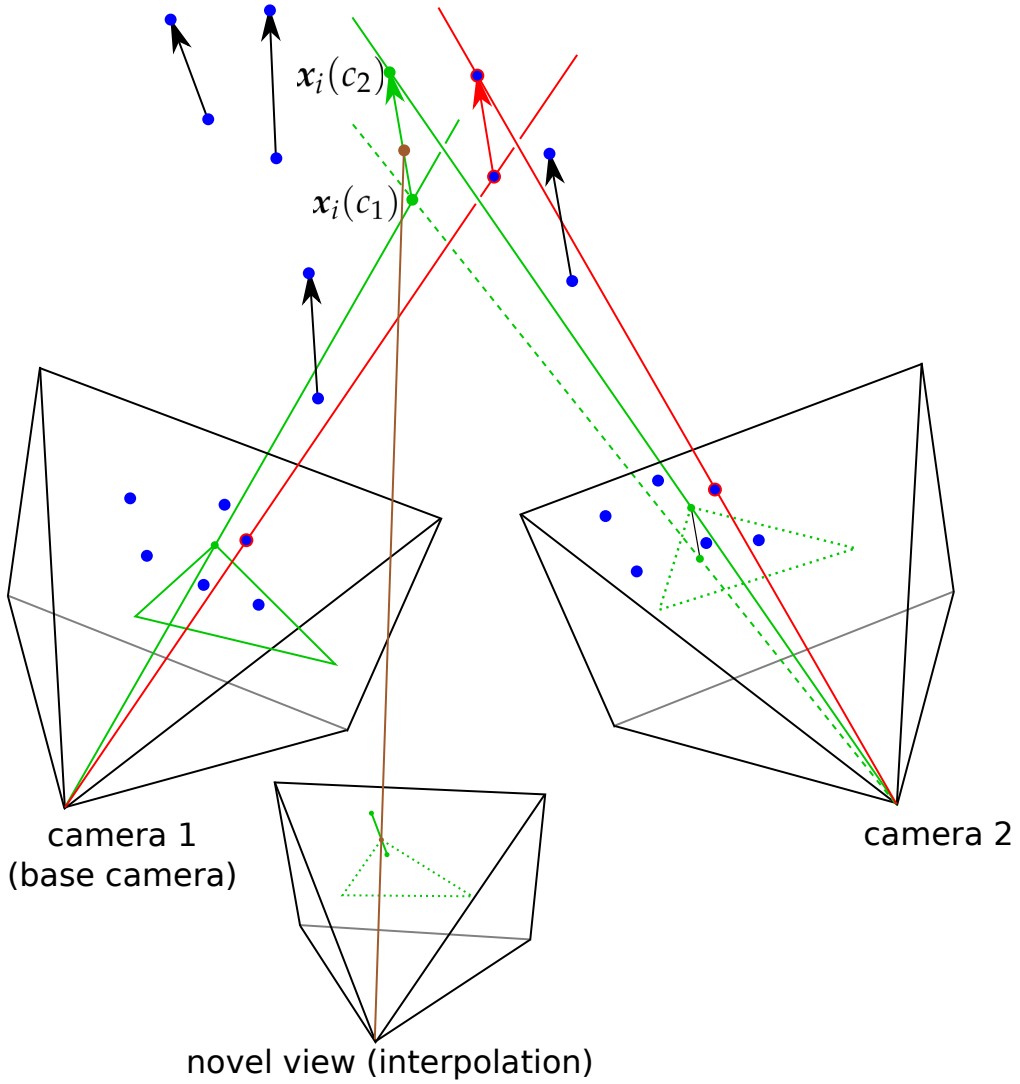


Figure 7.8: *Sketch of the view dependent geometry. The arrows indicate the view dependent geometric shift, i.e. the displacement.*

We know now how the feature points have to move in 3D. They are only subset of points in the entire space. However, we can use these to define a force field for the entire space. Every geometry point in 3D is then morphed according to a weighted average of displacements of nearby feature points. For our rendering we need to compute the displacements of triangle vertices. In the figure, an example triangle is shown in green. Without loss of generality, we assume this triangle has camera 1 as base camera. Projecting a vertex into space (green ray) at its given depth, results in a 3D point $x_i(c_1)$. To calculate the view-dependent morph function of this vertex, we interpolate all the displacements of neighboring features using a weighted average. This results in the green arrow, which is the displacement when moving

from the view of the base camera to the other cameras view. The weighting function is a Gaussian, while only features lying in a radius of 1 meter of the vertex are considered being neighbors. To render a novel view, the view-dependent position shifts along this arrow according to the angles between the cameras. An example for a novel view is shown in the image with the vertex interpolated in the brown point.

It is important to note that this view-dependent rendering (figure 7.1(f)) is not a 2D morph but a morph of the 3D geometry. It should not be confused with the merge of the 2.5D reconstructions (figure 7.1(e)). The merge is a simple union of triangles. These triangles only differ in the source camera they come from but are independent of each other. The geometry morph does *not* morph from the triangles of one base camera into those of another camera. It morphs all triangles according to the force field computed for them.

This geometry morph resolves most of the rendering artifacts as demonstrated in figure 7.7.

Formal Definition

In the following, we give a more formal definition of the morph described above.

Let $c_1 \in C$ be the base camera of a triangle and $\boldsymbol{x}_i(c_1)$ the projected 3D position (at given depth d_i) of a 2D vertex $\boldsymbol{x}_i$ of this triangle as illustrated in figure 7.8. The same vertex in another camera c_2 can be computed as:

$$\boldsymbol{x}_i(c_2) = \boldsymbol{x}_i(c_1) + \mathbf{s}, \tag{7.2}$$

where the displacement $\mathbf{s}$ is computed as average over the displacements of nearby feature points:

$$\mathbf{s} = \frac{1}{u} \sum_{\mathbf{f} \in F} w_{\mathbf{f}} s_{\mathbf{f}}(c_1, c_2), \tag{7.3}$$

with $s_{\mathbf{f}}(c_1, c_2)$ the displacement of feature point $\mathbf{f}$ from camera c_1 to camera c_2 and F the set of all feature points between these two cameras that have their 3D distance $dist(\boldsymbol{x}_i(c_1), \mathbf{f})$ to the vertex smaller than 1 meter. The weights are defined as

$$w_{\mathbf{f}} = e^{-\frac{(dist(\boldsymbol{x}_i(c_1), \mathbf{f}))^2}{2\sigma^2}}, \tag{7.4}$$

with $\sigma = 0.4$. u is the sum of all weights.

Let $\boldsymbol{x}_i(\hat{c})$ be the view-dependent 3d position that we need to compute to render the vertex $\boldsymbol{x}_i$ from a novel viewpoint $\hat{c}$. $\boldsymbol{x}_i(\hat{c})$ can be computed as a weighted sum of its corresponding positions in all cameras C:

$$\boldsymbol{x}_i(\hat{c}) = \sum_{c \in C} \lambda_c(\hat{c}) \boldsymbol{x}_i(c) \tag{7.5}$$

where the weights λ_c correspond to the blending weights

$$\lambda_c(\hat{c}) = \omega(\beta_c(\hat{c})) f(\boldsymbol{I}_{\omega_{Max}}) \tag{7.6}$$

as described in section 6.6.1. The angles $\beta_c(\hat{c})$ used for the blending are defined as the angle between the vector from $\boldsymbol{x}$ to the viewers position $\hat{\boldsymbol{c}}$ and the vector from $\boldsymbol{x}$ to the position of camera c, with

$$\boldsymbol{x} = \frac{1}{|C|} \sum_{c \in C} \boldsymbol{x}_i(c). \tag{7.7}$$

7.3.2 Texture Blending

For the texture blending, we use projective textures. However, it is important that for the projection from a source camera c onto a triangle we do not use the interpolated geometry (vertex $\boldsymbol{x}_i(\hat{c})$) but the geometry relating to camera c (vertex $\boldsymbol{x}_i(c)$). Otherwise the texture would be shifted on top of the triangle.

The color values are blended with the same weights λ_c used above. However, at occlusions, the λ_c of any camera c that does not cover the corresponding pixel is set to 0 and the weights are re-normalized.

View-Dependent Appearance

The cameras are typically not radiometrically calibrated. This causes also non-specular parts of the object to have a different appearance, e.g. a different brightness, in every camera. Especially at the border between an area visible in both cameras and an area occluded in one camera, this would cause artifacts like a border of two different brightness levels or color tones. To prevent this, we also compute the color shifts between cameras as follows.

In a pre-process, an average color R_c, G_c, B_c is computed per camera c. It is the average of the color values of those pixels of the image c that project onto 3D positions covered by more than one camera. Then, in the rendering, the above used blending weights λ_c are used to compute a color shift of the

current view for every camera. The average color value of all cameras is shifted, by shifting all its pixels equally, such that it matches

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

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

This view-dependent appearance shift preserves the interpolation at original camera views, since a flight into this camera view smoothly shifts the appearance to the one of this camera.

7.3.3 Use for Reconstruction

This method for computing a view-dependent geometry as well as the view-dependent texturing are also used for the reconstruction (section 7.2). Mainly, this is used for the computation of the back-projection error in section 7.2.3 and section 7.2.4. This is possible because the force field is independent of the triangulation and can be computed in a pre-process when computing the feature matches.

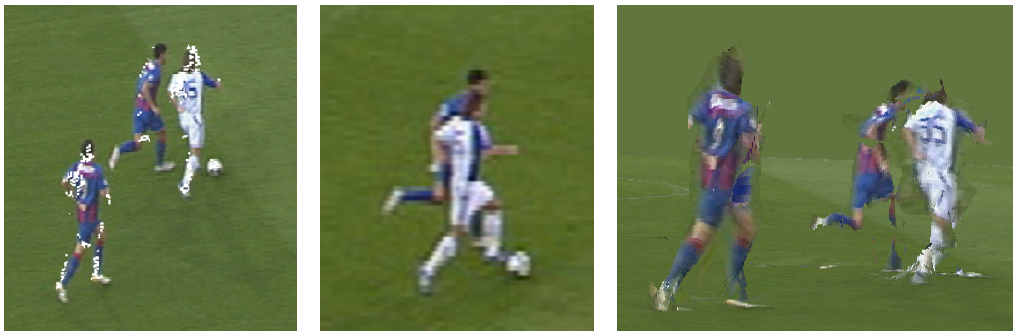
7.4 Implementation

An interesting implementation issue is the back-projection error. For the subdivision (section 7.2.3) it is computed on the GPU using a fragment shader to compute the per-pixel error. But for the refinement (section 7.2.4) the transfer back and forth between GPU and CPU is too slow such that a computation on the GPU is of any advantage. Therefore, for the refinement, the back-projection error is computed on the CPU.

The texture blending is implemented in a fragment shader. Instead of shifting the geometry for each camera for the projective texturing, we simply shift the texture lookup coordinates in the fragment shader to match each camera. This allows to use standard OpenGL projective texturing.

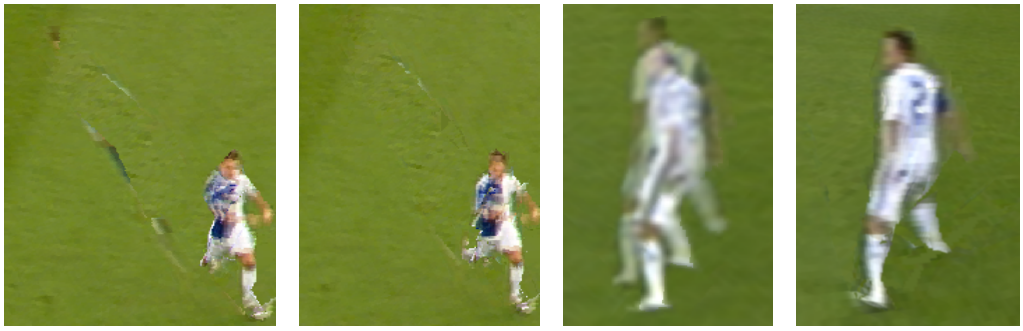
7.5 Results

Similar to the previous two chapters we used footage of original TV broadcasts of several soccer games. All computations were done on a standard



(a) (b) (c)

Figure 7.9: *Difficult occlusion. (a) View in camera 1 with feature points shown as white dots. (b) View in camera 2 where one player is almost entirely occluded. (c) Novel view.*



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

Figure 7.10: *(a) Reconstruction with triangulation in camera 1. The ghostings on the ground are parts not seen and thus not reconstructed in camera 1. (b) Merge of the reconstructions of camera 1 and camera 2. (c) A rendering using billboarding. (d) The same view using our method.*

desktop computer with an Intel Core i7 CPU with 3.20GHz. All results are reconstructions based on only two input cameras.

Despite low resolutions and calibration errors, our algorithm fully automatically reconstructs plausible novel views as shown in figure 7.10(d). Figure 7.7 illustrates the effect of the view dependent geometry that is able to reduce calibration error artifacts.

With our improved DAISY feature matching, the feature matches are reduced to a set of reliable correspondences. Sometimes this set contains only a few matches per player, but nevertheless our adaptive reconstruction method recovers a good approximation of the players geometry as shown in figure 7.9.

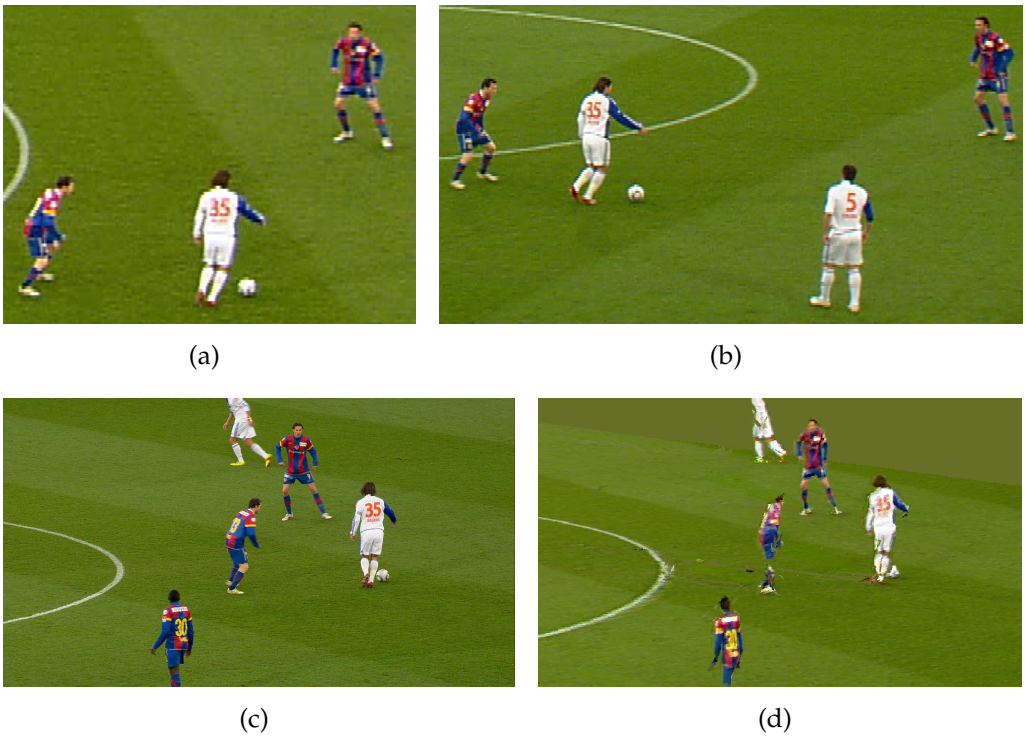
Our method performs a reconstruction for each camera as a base camera and merges these reconstruction into a final geometry. This can be used to recover parts occluded in one camera but visible in another. The result of merging two reconstructions is shown in figure 7.10, where figure 7.10(a) shows a novel view using the reconstruction of camera 1 only and figure 7.10(b) shows the same view with the unified reconstruction of the two cameras. With the reconstruction of only camera 1, parts of players visible only in the camera 2 are not reconstructed. The image parts in camera 2 belonging to them are thus projected onto the ground, which is clearly visible in the novel view. With the merged reconstruction we also get a valid depth for these parts allowing to reconstruct occlusions (figure 7.1(e)). Figure 7.9 demonstrates this with another example.

Figures 7.10(c) and 7.10(d) show a comparison to billboarding in a view halfway between the two source cameras. Billboarding shows ghosting artifacts due to two main reasons. First, there is a global offset which can be seen in the vertical shift between the two blended camera source images. This is because of calibration errors. Second, there are ghostings because the geometric proxy is planar and does not cover the articulation. This can be seen, e.g., at the duplication of the legs. On the other hand, our method corrects the calibration errors and preserves the pose of the player in the novel view, as shown in figure 7.10(d).

Figure 7.11 depicts a ground truth comparison to a third camera which was not used in the reconstruction. Nevertheless the third camera has a high resolution coverage while the first two cameras used for the reconstruction are wide angle shots, the comparison demonstrates that the result of our reconstruction is correct and with similar quality as the input images.

More results are shown in figure 7.12 and figure 7.13. The first two images of every block are always the input images followed by the novel views. They show views not lying directly in between the cameras, e.g. top views or views from the height of the players heads. This is not possible by simple image interpolation methods.

To test a possible application, we also produced flights over time. Despite not using any temporal information, coherence or smoothing, these dynamic scenes still result in plausible renderings. Artifacts are visible mostly when players come into or leave a cameras view range. Since we reduce the search for DAISY feature matches to the bounding box found by the player detection, a player that is not detected will not be reconstructed. An example for this is also the ball, which was not detected and thus not reconstructed at all. For the ball we plan to use tracking methods (e.g. Choi et al. [2006]).



(a) (b)

(c) (d)

Figure 7.11: *Leave-one-out example: (a) and (b) Closeups of the two source camera images used. (c) High resolution ground truth from a third (not used) camera that zoomed in. (d) Our reconstruction of the same view*

The rendering of novel views is done in real-time, i.e., more than 60 frames per second for HD resolution. Our fully automatic algorithm for reconstruction takes on average 1 minute per frame. The exact timing depends on the scene complexity, but none of our examples required more than 2 minutes to reconstruct. About 19 seconds of this time are DAISY feature vector computation and our feature matching. The rest of the time is spent about equally for positioning, and for the refinement.

7.6 Discussion and Outlook

In this chapter we presented a fully automatic novel-view synthesis method suitable for conventional TV sport broadcasts. The setup consists of only two wide-baseline video cameras. The main ingredients of our method is an adaptive reconstruction technique that can reconstruct players even at very low resolutions and a view-dependent geometry morph and rendering technique.

Figure 7.12: *Results: the first row (little images) always shows the input camera views.*

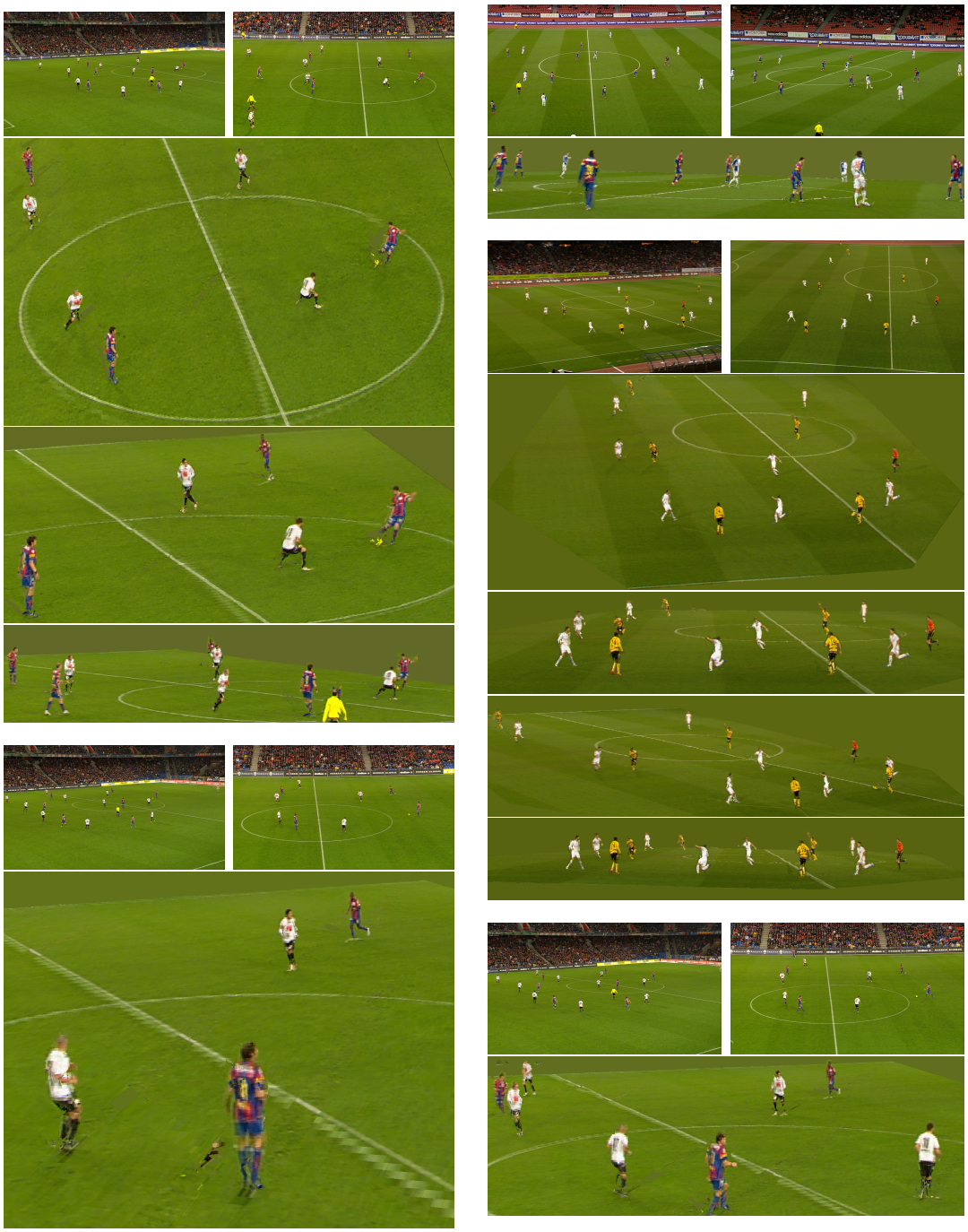


Figure 7.13: *Results: the first row (little images) always shows the input camera views.*

The geometry is reconstructed in a top-down fashion. Geometric detail is added gradually in the areas where it is needed based on a reliable sparse feature matching technique. It also robustly handles areas with no image features by inferring the position based on the neighboring triangles and back-projection error. The geometry is then rendered from a novel viewpoint using a view-dependent geometry morphing and texture interpolation technique that alleviates rendering artifacts stemmed from calibration errors. We proved the visual quality and reliability of our technique by applying it to footage of soccer broadcasts.

Although our approach produces convincing results for a two wide-baseline camera setup, one can still spot some minor visual artifacts. Many of these are caused by inherent limitations by the fact that we only use two cameras. For instance, subjects too close to the front will have nearly no overlapping parts and, therefore, cannot be reconstructed. Also, some subjects are visible only in one camera and cannot be reconstructed. Due to lack of good robust matches occasionally triangles are not positioned optimally causing cracking artifacts. Although our method can recover a plausible depth in most occlusion cases where parts are only visible in one camera, there are situations where too few or no features were found in the neighborhood and thus leading to visual artifacts. Simply adding more cameras to the setup will automatically improve these issues. The camera weights λ, the view dependent morph (equation (7.5)) and the color shifts can be computed for any arbitrary number of cameras.

The time required per frame could be reduced by parallelization in several ways:

- The reconstruction for each base camera is done completely independently (figure 7.1). This could be done in parallel in on thread per camera.
- The refinement is done independently per triangle, since no occlusion tests are done in this step. A parallelization of this could be done using CUDA to also reduce the transfer between GPU and CPU by doing everything on the GPU.
- There are already methods for a parallel implementation of DAISY feature matching [Fischer et al., 2011]. This could be extended for our filtering of the DAISY features too.

We would like to investigate on ideas to use the computed geometry morph vectors for correcting the camera calibrations. This could allow to reduce the geometry morph to deformations (i.e., lens distortions or similar artifacts)

that are not covered by the calibration. The rest would be already covered by the camera calibration.

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

CHAPTER 8

Conclusion

In this chapter, we conclude the main contributions of this thesis as a summary and compare the two presented approaches. We will also discuss future work inspired by or built on top of methods presented in this thesis.

8.1 Summary

In this thesis we proposed two different solutions for a full video-based rendering pipeline. They introduce novel techniques for reconstruction, novel representations and novel rendering methods - all of them designed for broadcast material from outdoor sports. Such footage is difficult to process due to wide baselines between sparsely placed cameras, low resolutions, difficult lighting conditions and thus inaccurate camera calibrations.

In chapter 5 we introduced a two stage body pose estimation for these challenging setups. In a first part, the pose is roughly estimated by transferring 2D poses from a database of annotated silhouettes. The comparison with the database entries is done in a sliding window approach by comparing silhouettes and aspect ratios of the silhouette sizes. The k best 2D pose guesses are verified by 3D triangulation and the combination that matches best in 3D is taken as the initial pose estimation. However, this is still a rough guess of the pose because it is taken from an unchanged database entry. To adapt to the

actual input data and smooth in between frames, we employ a space-time pose optimization. It changes the joint angles and the global orientation of the 3D skeleton to fit the 2D silhouette projections, to smooth between the frames and also to fulfill anthropometric constraints. The final pose estimation is robust and does not suffer from drift like tracking methods. It is able to obtain the full body poses even with partial occlusions and low resolution coverage of the subject.

The body pose estimation is directly used for the construction of our proposed articulated billboard representation, presented in chapter 6. Articulated billboards are a simple but powerful model for humans in outdoor setups with sparse camera placement. They consist of a skeleton with billboard fans attached to every body part. The texture for the billboards are obtained in a segmentation method that adaptively learns from reliable regions to segment unreliable pixels. The reliable regions are detected by employing a projection of a template model - grown or shrunk to different sizes. To render articulated billboards, the billboards of each body part are blended view-dependently to achieve smooth and realistic interpolations at arbitrary viewpoints. Articulated billboards are robust to calibration errors but preserve the perception of the body pose including self-occlusions.

We presented an alternative approach for a video-based rendering method in chapter 7. The proposed coarse-to-fine reconstruction algorithm also targets on low resolution outdoor setups with weak calibrations. A separate 2.5D reconstruction is obtained in every camera image in use of the other camera images, and these reconstructions are merged to a final 3D reconstruction. To achieve a 2.5D reconstruction, the base camera image is simply triangulated with a coarse mesh. However, the triangles are not connected and in the subsequent depth optimization, they receive independent depth values for every vertex according to feature point matches. We employ improved DAISY feature matches for this purpose to result in sparse but reliable correspondences. The optimization process is iterated with a subdivision of only those triangles that do not match the surface yet. This results in further computation only at regions where this is needed, thus lowering the computational costs. After these iterations, a refinement step according to back-projections and neighbor look-ups improves the found vertex depths and results in the final 2.5D reconstruction of this camera. The merged 3D reconstruction is rendered not only with a view-dependent blending but also with a view-dependent geometry. This morph of the geometry is achieved by a force field computed from non-epipolar feature matches, where the distance of a match to the correct epipolar correspondence gives the amount of shift at this 3D position. The reconstruction is robust to several errors occurring particularly in outdoor setups and the view-dependent rendering corrects

for calibration errors, for which no 3D reconstruction would fit to all camera images.

We showed for both solutions of video-based rendering results from several soccer games. The input videos are standard TV broadcast footage. Our results show that the rendered images are comparable to the quality of the input images and usually difficult to distinguish from them. Also the transitions from real into rendered images is achieved without any jumps or changes, resulting in smooth output videos.

8.2 Comparison of the two Solutions

At this point we would like to compare the two solutions presented in this thesis, the body pose estimation (chapter 5) combined with articulated billboards (chapter 6) to the adaptive geometry reconstruction and view-dependent morph (chapter 7). We will refer to the first one simply as articulated billboards and to the second one as view-dependent geometry.

A main advantage of the articulated billboards is that they achieve a slightly better visual quality than the view-dependent geometry. The reason for this is mainly that the articulated billboards assure a human body pose, whereas the view-dependent geometry does not use any prior to make sure the final reconstruction is human like. However, in some situations where the pose estimation fails, articulated billboards not only loose this advantage, but also result in rendering artifacts.

The view-dependent geometry on the other hand has the big advantage that it does not require any manual interaction in the reconstruction. In the pose estimation for the articulated billboards there is still a few mouse clicks required to assure no left-right flips occur. The reconstruction of the view-dependent geometry fully automatically solves for occlusion problems and calibration errors. It is even able to recover the 3D position of parts that are covered in only one camera as long as there are a few feature points nearby.

As a result of this, we propose to use articulated billboards for offline setups, where more time is available and high quality results are expected. This could also be the analysis of a single freeze-frame of a moment that was very important for the sports game. In such a setup, one could also add more possibilities to interact with the system, e.g. besides correcting the body pose also refining the segmentation to get even better results. On the other hand, we propose to use the fully automatic view-dependent geometry for instant replays or mid- and post-game analysis where fast results are required. Also

it can be used for the reconstruction of longer sequences where a manual interaction is too cumbersome.

8.3 Future Work

The two approaches for video-based rendering presented in this thesis are suited well for uncontrolled outdoor setups and showed results in a comparable quality of the input images. However, there are still a lot of ideas for future work. Most of them were already discussed at the end of the corresponding chapters, i.e. in sections 5.7, 6.9, and 7.6. In this section, we will discuss some further ideas and preliminary results to extend, or improve the work of this thesis.

8.3.1 Image Enhancement

Video-based rendering relies only on the input camera images. Their quality is crucial. Therefore, possible future work is the enhancement of the camera textures in the initialization phase.

The TV footage we used is interlaced, i.e., only every other pixel row is changed per frame, iterating between the even and the odd lines of pixels. In a de-interlacing step the missing lines are reconstructed. We currently use a simple copy of the corresponding neighbor row. This has the advantage that it preserves sharp edges. However, in terms of signal theory this is not the optimal reconstruction. We would like to try several state of the art methods to achieve a better de-interlacing.

Another way of improving the quality of the camera textures is super-resolution. Techniques for super-resolution aim to improve images by replacing one pixel by several pixels, whose color values are computed by sophisticated interpolations and statistical or data-driven methods. In our case, there is a lot of prior knowledge about the scene that directly can be used for this. First of all, we know what type of game it is. Secondly, we usually can apply a simple background subtraction to distinguish between players and background and thus know for every pixel also to where it belongs to. First steps into this direction have been pursued in the master thesis on texture enhancement for video-based rendering by Remo Frey [Frey, 2009]. Based on the approach of Freeman et al. [2002], image patches are compared on a pixel-basis to downsampled patches from a database and replaced by the high resolution database patch if there is a high similarity and the patch fits to neighboring patches. This was improved by using prior knowledge in

(a) (b) (c)

Figure 8.1: *Preliminary result for a super-resolution algorithm applied on a player image taken from a TV broadcast. (a) The deinterleaced input image. (b) Enlargement by bicubic interplation for comparison. (c) Our result.*

several ways. The input patches are only compared to those patches in the database that correspond to the same size in the real world. This prevents from comparing, e.g., body parts of the size of a leg to body parts of the size of a nose. For this, not only pairs of patches (the original and a downsampled version) are kept in the database but an entire *band-pass pyramid* which contains several levels of downsampled versions. Every layer of the pyramid corresponds to one pixel size in meter in the real world. Another improvement was achieved by also adding rotations to the patch comparison, which increased the effective size of the database without additional data or storage. A preliminary result of this algorithm is shown in figure 8.1[1]. In this image the resolution is doubled from the original to the enhanced version. This and other results showed that a data-driven approach is able to at least slightly improve the texture quality compared to bicubic interpolation. However, at bigger magnifications artifacts are introduced. Therefore, we would like to investigate further into such methods.

8.3.2 Slow Motion

In conventional video, slow motion is achieved by increasing the time span of each frame to be shown. This introduces non-continuous motions which are unpleasing for the viewer. Therefore, usually the optical flow is used

[1]This figure is from the master thesis of Remo Frey [2009]

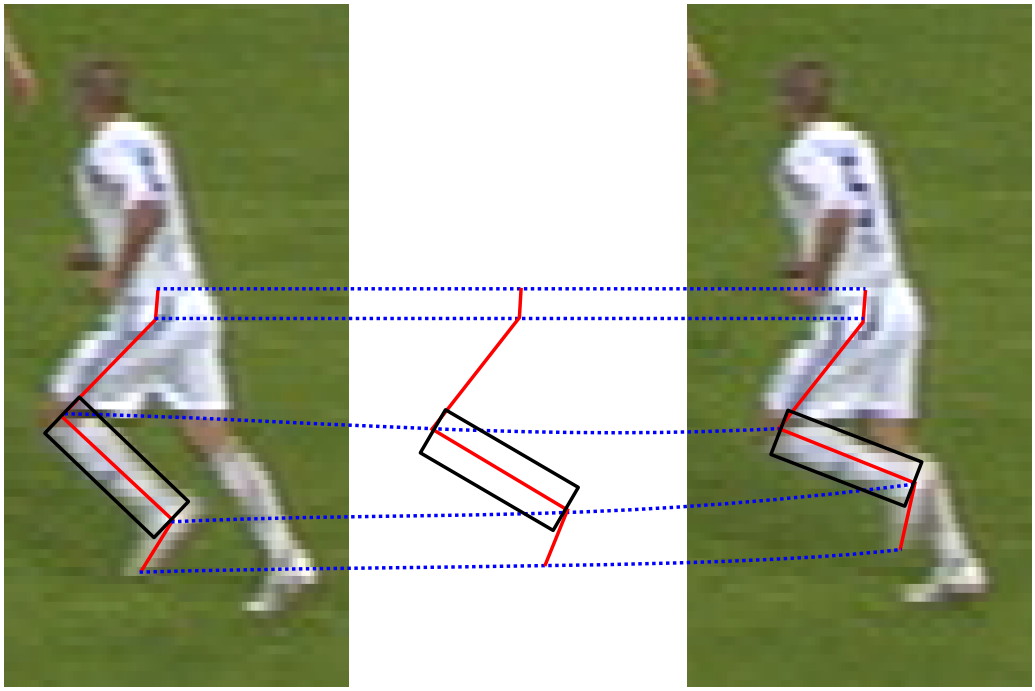


Figure 8.2: *Two subsequent frames with a possible interpolation for slow motion. The black rectangle indicates the position of one of the articulated billboards.*

to achieve smooth interpolations. This requires reliable computation of the optical flow. Also, this is done in 2D without using additional knowledge for the human body (i.e., use its 3D articulation).

An extension for the articulated billboards in the temporal dimension could be the computation of a smooth slow motion directly from the 3D model. This would rely on the body pose estimation and not on the optical flow, which is in our setup difficult to reliably compute. In the following, we give a quick overview how such an approach could be achieved.

Assuming that for two subsequent frames, the articulated billboards representation are computed and that we know the correspondences between players and models, i.e. for every articulated billboards model in frame t, we know which articulated billboards model corresponds to the same player in frame $t+1$. An example of two subsequent frames for a sequence with 25 frames per second is shown in figure 8.2. We can now define a motion function between the two frames. The root bone is simply translated and rotated from its position in t to the position in $t+1$, directly shifting the entire representation. For the rest of the bones, the angles between them are changed such that they fit the corresponding bones in the next frame - according to the kinematic chain of the body. From this motion function for the skeleton we can directly derive the motion function of the billboards, since they are defined by the

bone and the camera positions. For the texture and the alpha mask on top of a billboard, either simple blending or a morphing approach can be used.

With this method of interpolating the motion and the texture, it is possible to render extreme slow motions smoothly. A further improvement would be to insure also c_1 continuity of the motion between several subsequent frames.

8.3.3 Full Scene Reconstruction

The adaptive reconstruction (chapter 7) is not only a video-based rendering technique. It is also able to reconstruct a full 3D model that can be used for many other applications. However, currently we focus only on the human body. We would like to extend this to background objects, like the soccer stadium or any other scenery.

Preliminary test showed that the in section 7.2.4 presented refinement can be used for the field to correct for non-flat surfaces. A reason for such non-flat surfaces is that a soccer field is at the center slightly higher and falls off towards the borders, to make sure rain water can drain off, called field crown.

For the triangle optimization in the construction phase (section 7.2.2), an extension to background objects is difficult. The reason is that on the field as well as in the crowd of viewers, the quality of feature points is too low to achieve good matches. Therefore, to compute the optimal plane in these regions either an approach with other feature matching methods should be pursued, or an approach based on back-projection errors similar to the refinement should be developed. Also, an improvement of the calibration would be very helpful. With more elaborated calibration models and methods, epipolar constraints could be used in the feature matching instead of just near epipolar correspondences. Thus, this would allow for more and reliable features also in these areas of the image.

8.3.4 3D TV and Augmented Reality

Methods for computing 3D TV output could directly gain from both approaches presented in this thesis. This is especially the case for the adaptive reconstruction, where the resulting 3D model can be used to generate not only renderings but also depth maps from arbitrary viewpoints. Many 3D TV formats use such a depth map to generate the per pixel disparities. Other formats use multiple views, i.e. renderings from usually two viewpoints, as input. Also this can be generated easily from both our solutions.

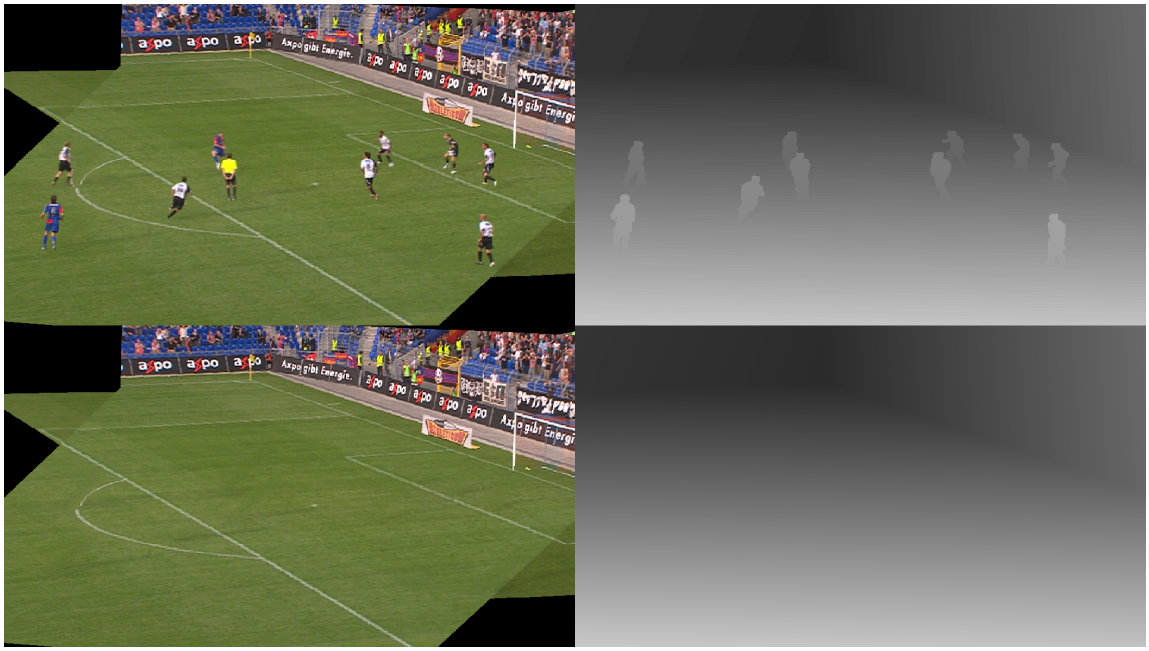


Figure 8.3: *An example for a WoW format view of a soccer scene as an input to a 3D TV. The upper row shows the texture and the depth map of the foreground and the bottom row the same of the background.*

In the master thesis of Marcel Müller [Müller, 2009], preliminary results for a 3D TV output computation were generated. Philips 3D TV WoW was used to show scenes from soccer games in two different approaches. In a first approach, the above described procedure was implemented, i.e., the for the Philips display required front and back layer textures as well as depth maps were computed directly on the GPU. The second approach targeted on a direct conversion of footage from a single 2D camera into 3D display output. Using camera calibration and a coarse blob detection, the players were folded up resulting in one billboard per player. The results already showed a clear and pleasant better perception of the 3D positions of the players. Figure 8.3[2] shows 3D footage generated according to this method. However, this could be improved by using either articulated billboards or the view-dependent geometry as a basis instead of single billboards per player.

In a similar fashion, we would like to investigate the use of augmented reality in our scenes. Some results were already achieved in the master thesis of Marcel Müller, with virtual 3D drawings and objects. Other effects could be motion streaks or blur in 3D, or space-time ramps. An ultimate goal would be to render the studio expert, who analyzes the game, directly onto the pitch. Similar to tele-presence systems, the expert would be somewhere in the

[2]This figure is from the master thesis of Marcel Müller [2009]

studio in an acquisition setup, but her or his virtual representation would be rendered and directly shown as a 3D object in the scene, able to walk between the field players.

8.3.5 Merge of the Two Solutions

Finally, we would like to investigate on a possible merge of the two solutions. Even though, a direct merge would be difficult, at least parts of one method could be used for parts of the other method and vise-versa. An example for this is the pose estimation. The adaptive reconstruction could profit from a pose estimation, e.g. in the initialization phase. The bone positions or a template model fitted to these positions could be used for the initialization of the triangle depths.

Also we would like to investigate if the adaptive reconstruction could be used for a 3D body pose estimation, where the template skeleton is not fitted to the silhouettes but to the reconstruction directly, similar to the method by Carranza et al. [2003] but on a coarser low-resolution level.

Bibliography

[Agarwal and Triggs, 2006] A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(1):44 –58, January 2006.

[Andriluka et al., 2009] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2009 IEEE Conference on*, pages 1014 –1021, June 2009.

[Andriluka et al., 2010] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Monocular 3d pose estimation and tracking by detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 623 –630, June 2010.

[Andújar et al., 2007] Carlos Andújar, Javier Boo, Pere Brunet, Marta Fairén González, Isabel Navazo, Pere-Pau Vázquez, and Alvar Vinacua. Omnidirectional relief impostors. *Computer Graphics Forum (Proceedings of Eurographics)*, 26(3):553–560, 2007.

[Aubel et al., 1999] A. Aubel, R. Boulic, and D. Thalmann. Lowering the cost of virtual human rendering with structured animated impostors. In *Proceedings on WSCG'99*, 1999.

[Balan et al., 2007] Alexandru O. Balan, Leonid Sigal, Michael J. Black, James E. Davis, and Horst W. Haussecker. Detailed human shape and pose from images.

In *Computer Vision and Pattern Recognition (CVPR), 2007 IEEE Conference on*, pages 1 –8, June 2007.

[Ballan and Cortelazzo, 2008] Luca Ballan and Guido Maria Cortelazzo. Markerless motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *3DPVT*, Atlanta, GA, USA, June 2008.

[Ballan et al., 2010] Luca Ballan, Gabriel J. Brostow, Jens Puwein, and Marc Pollefeys. Unstructured video-based rendering: interactive exploration of casually captured videos. In *ACM Transactions on Graphics*, SIGGRAPH '10, pages 87:1–87:11, New York, NY, USA, 2010. ACM.

[Barnes et al., 2010] Connelly Barnes, Eli Shechtman, Dan B. Goldman, and Adam Finkelstein. The generalized patchmatch correspondence algorithm. In *Proceedings of the 11th European conference on computer vision conference on Computer vision: Part III*, ECCV'10, pages 29–43, Berlin, Heidelberg, 2010. Springer-Verlag.

[Behrendt et al., 2005] Stephan Behrendt, Carsten Colditz, Oliver Franzke, Johannes Kopf, and Oliver Deussen. Realistic real-time rendering of landscapes using billboard clouds. *Computer Graphics Forum (Proceedings of Eurographics)*, 24(3):507–516, 2005.

[Beier and Neely, 1992] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *ACM Transactions on Graphics*, SIGGRAPH '92, pages 35–42, New York, NY, USA, 1992. ACM.

[Bouguet, 2000] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm, 2000. Intel Corporation Microprocessor Research Labs.

[Bradley et al., 2008] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. In *ACM Transactions on Graphics*, SIGGRAPH '08, pages 99:1–99:9, New York, NY, USA, 2008. ACM.

[Buehler et al., 2001] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *ACM Transactions on Graphics*, SIGGRAPH '01, pages 425–432, New York, NY, USA, 2001. ACM.

[Canny, 1986] J Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 8:679–698, June 1986.

[Carranza et al., 2003] Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. In *ACM Transactions on Graphics*, SIGGRAPH '03, pages 569–577, New York, NY, USA, 2003. ACM.

[Choi et al., 2006] Kyuhyoung Choi, Booil Park, Subin Lee, and Yongduek Seo. Tracking the ball and players from multiple football videos. *International Journal of Information Acquisition*, 3(2):121–129, 2006.

[CMU, 2012] CMU. CMU Graphics Lab Motion Database, 2012. http://mocap.cs.cmu.edu.

[Connor and Reid, 2003] Keith Connor and Ian Reid. A multiple view layered representation for dynamic novel view synthesis. In *British Machine Vision Conference*, BMVC '03, 2003.

[de Aguiar et al., 2008] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. In *ACM Transactions on Graphics*, SIGGRAPH '08, pages 98:1–98:10, New York, NY, USA, 2008. ACM.

[Debevec et al., 1996] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *ACM Transactions on Graphics*, SIGGRAPH '96, pages 11–20, New York, NY, USA, 1996. ACM.

[Décoret et al., 2003] Xavier Décoret, Fredo Durand, and Francois X. Sillion. Billboard clouds. In *Symposium on Computational Geometry*, SCG '03, pages 376–376, New York, NY, USA, 2003. ACM.

[Dempster et al., 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[Efros et al., 2003] Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *International Conference on Computer Vision*, ICCV '03, pages 726–733, Washington, DC, USA, 2003. IEEE Computer Society.

[Eisemann et al., 2008] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson de Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating textures. *Computer Graphics Forum (Proc. of Eurographics)*, 27(2):409–418, April 2008.

[El-Melegy and Farag, 2003] Moumen T. El-Melegy and Aly A. Farag. Nonmetric lens distortion calibration: Closed-form solutions, robust estimation and model selection. In *International Conference on Computer Vision*, volume 1 of *ICCV '03*, page 554, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[Felzenszwalb and Huttenlocher, 2000] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient matching of pictorial structures. In *Computer Vision and*

Pattern Recognition (CVPR), 2000 IEEE Conference on, volume 2, pages 66 –73 vol.2, 2000.

[Felzenszwalb and Huttenlocher, 2005] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61:55–79, January 2005.

[Ferrari et al., 2008] Vittorio Ferrari, Manuel Marín-Jiménez, and Andrew Zisserman. Progressive search space reduction for human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on*, pages 1 –8, June 2008.

[Fischer et al., 2011] Jan Fischer, Alexander Ruppel, Florian Weisshardt, and Alexander Verl. A rotation invariant feature descriptor O-DAISY and its FPGA implementation. In *IROS*, pages 2365–2370. IEEE, 2011.

[Fischler and Bolles, 1981] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commununications of the ACM*, 24:381–395, June 1981.

[Fleuret et al., 2008] François Fleuret, Jérôme Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):267 –282, feb. 2008.

[Forsyth and Ponce, 2002] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 1 edition, August 2002.

[Fossati et al., 2009] Andrea Fossati, Mathieu Salzmann, and Pascal Fua. Observable subspaces for 3d human motion recovery. In *Computer Vision and Pattern Recognition (CVPR), 2009 IEEE Conference on*, pages 1137 –1144, June 2009.

[Franco and Boyer, 2009] Jean-Sébastien Franco and Edmond Boyer. Efficient polyhedral modeling from silhouettes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31:414–427, March 2009.

[Fraundorfer et al., 2006] Friedrich Fraundorfer, Konrad Schindler, and Horst Bischof. Piecewise planar scene reconstruction from sparse correspondences. *Image Vision and Computing*, 24:395–406, April 2006.

[Freeman et al., 2002] William T. Freeman, Thouis R. Jones, and Egon C. Pasztor. Example-based super-resolution. *Computer Graphics and Applications, IEEE*, 22(2):56 –65, mar/apr 2002.

[Freixenet et al., 2002] Jordi Freixenet, Xavier Muñoz, D. Raba, Joan Martí, and Xavier Cufí. Yet another survey on image segmentation: Region and boundary

information integration. In *Proceedings of the 7th European Conference on Computer Vision-Part III*, ECCV '02, pages 408–422, London, UK, UK, 2002. Springer-Verlag.

[Frey, 2009] Remo Frey. *Texture Enhancement for Video-based Rendering*. Institute of Visual Computing, ETH Zurich, 2009.

[Gallup et al., 2010] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1418 –1425, June 2010.

[Germann et al., 2007] Marcel Germann, Michael D. Breitenstein, In Kyu Park, and Hanspeter Pfister. Automatic pose estimation for range images on the gpu. In *Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling*, 3DIM '07, pages 81–90, Washington, DC, USA, 2007. IEEE Computer Society.

[Germann et al., 2010] Marcel Germann, Alexander Hornung, Richard Keiser, Remo Ziegler, Stephan Würmlin, and Markus Gross. Articulated billboards for video-based rendering. *Computer Graphics Forum (Proceedings of Eurographics)*, 29(2):585–594, 2010.

[Germann et al., 2011] Marcel Germann, Tiberiu Popa, Remo Ziegler, Richard Keiser, and Markus Gross. Space-time body pose estimation in uncontrolled environments. In *Proceedings of the 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 3DIMPVT '11, pages 244–251, Washington, DC, USA, 2011. IEEE Computer Society.

[Germann et al., 2012] Marcel Germann, Tiberiu Popa, Richard Keiser, Remo Ziegler, and Markus Gross. Novel-view synthesis of outdoor sport events using an adaptive view-dependent geometry. *Computer Graphics Forum (Proceedings of Eurographics)*, 2012. To appear.

[Goesele et al., 2010] Michael Goesele, Jens Ackermann, Simon Fuhrmann, Carsten Haubold, Ronny Klowsky, Drew Steedly, and Richard Szeliski. Ambient point clouds for view interpolation. In *ACM Transactions on Graphics*, SIGGRAPH '10, pages 95:1–95:6, New York, NY, USA, 2010. ACM.

[Goldlücke and Magnor, 2003] Bastian Goldlücke and Marcus Magnor. Real-time microfacet billboarding for free-viewpoint video rendering. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 3, pages III – 713–16 vol.2, September 2003.

[Gortler et al., 1996] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *ACM Transactions on Graphics*, SIGGRAPH '96, pages 43–54, New York, NY, USA, 1996. ACM.

[Grau et al., 2007] Oliver Grau, Graham A. Thomas, Adrian Hilton, Joe Kilner, and Jonathan Starck. A robust free-viewpoint video system for sport scenes. In *3DTV Conference, 2007*, pages 1 –4, May 2007.

[Grauman et al., 2003] Kristen Grauman, Gregory Shakhnarovich, and Trevor Darrell. Inferring 3d structure with a statistical image-based shape model. In *International Conference on Computer Vision*, ICCV '03, pages 641–648. IEEE Computer Society, 2003.

[Gross et al., 2003] Markus Gross, Stephan Würmlin, Martin Naef, Edouard Lamboray, Christian Spagno, Andreas Kunz, Esther Koller-Meier, Tomas Svoboda, Luc Van Gool, Silke Lang, Kai Strehlke, Andrew Vande Moere, and Oliver Staadt. blue-c: a spatially immersive display and 3d video portal for telepresence. In *ACM Transactions on Graphics*, SIGGRAPH '03, pages 819–827, New York, NY, USA, 2003. ACM.

[Guillemaut and Hilton, 2011] Jean-Yves Guillemaut and Adrian Hilton. Joint multi-layer segmentation and reconstruction for free-viewpoint video applications. *International Journal of Computer Vision*, 93:73–100, 2011.

[Guillemaut et al., 2009] J. Y. Guillemaut, J. Kilner, and A. Hilton. Robust graph-cut scene segmentation and reconstruction for free-viewpoint video of complex dynamic scenes. In *International Conference on Computer Vision*, ICCV '03, Kyoto, Japan, September 2009.

[Gupta et al., 2008] Abhinav Gupta, Anurag Mittal, and Larry S. Davis. Constraint integration for efficient multiview pose estimation with self-occlusions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(3):493–506, January 2008.

[Hartley and Zisserman, 2003] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[Hayashi and Saito, 2006] Kunihiko Hayashi and Hideo Saito. Synthesizing Free-Viewpoint Images from Multiple View Videos in Soccer Stadium. In *CGIV '06: Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation*, pages 220–225, Washington, DC, USA, 2006. IEEE Computer Society.

[Heigl et al., 1999] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc Van Gool. Plenoptic modeling and rendering from image sequences taken by hand-held camera. In *Mustererkennung 1999, 21. DAGM-Symposium, Proceedings*, pages 94–101. Springer, 1999.

[Hilton et al., 2011] Adrian Hilton, Jean-Yves Guillemaut, Joe Kilner, Oliver Grau, and Graham A. Thomas. 3D-TV production from conventional cameras for sports broadcast. *Broadcasting, IEEE Transactions on*, 57(2):462 –476, June 2011.

[Hogg, 1983] David Hogg. Model-based vision: a program to see a walking person. *Image and Vision Computing*, 1(1):5 – 20, 1983.

[Hornung et al., 2007] Alexander Hornung, Ellen Dekkers, and Leif Kobbelt. Character animation from 2D pictures and 3D motion data. *ACM Trans. Graph.*, 26, January 2007.

[Hough, 1962] Paul V.C. Hough. Method and means of recognizing complex patterns. *US Patent 3,069,654*, 1962.

[Inamoto and Saito, 2002] Naho Inamoto and Hideo Saito. Intermediate view generation of soccer scene from multiple videos. In *ICPR (2)*, pages 713–716, 2002.

[Inamoto and Saito, 2007] Naho Inamoto and Hideo Saito. Virtual viewpoint replay for a soccer match by view interpolation from multiple cameras. *Multimedia, IEEE Transactions on*, 9(6):1155 –1166, oct. 2007.

[Jaeggli et al., 2005] Tobias Jaeggli, Thomas P. Koninckx, and Luc Van Gool. Model-based sparse 3D reconstruction for online body tracking. In *Proceedings of IS&T/SPIE's 17th annual symposium on electronic imaging - videometrics VIII*, volume vol. 5665, pages 226–234 (5665–31), January 2005.

[Kanade et al., 1995] Takeo Kanade, P.J. Narayanan, and Peter W. Rander. Virtualized reality: concepts and early results. In *Representation of Visual Scenes, 1995. Proceedings IEEE Workshop on*, pages 69 –76, jun 1995.

[Kanade et al., 1997] Takeo Kanade, Peter W. Rander, and P.J. Narayanan. Virtualized reality: constructing virtual worlds from real scenes. *Multimedia, IEEE Transactions on*, 4(1):34 –47, jan-mar 1997.

[Kilner et al., 2007] Joe Kilner, Jonathan Starck, Adrian Hilton, and Oliver Grau. Dual-mode deformable models for free-viewpoint video of sports events. *3DIM*, pages 177–184, 2007.

[Kinect, 2010] Kinect. Microsoft kinect, 2010. http://www.xbox.com/en-US/kinect.

[Laurentini, 1994] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16:150–162, February 1994.

[Le et al., 2005] Oliver Le, Anusheel Bhushan, Pablo Diaz-Gutierrez, and M. Gopi. Capturing and view-dependent rendering of billboard models. In *Advances in*

Visual Computing, First International Symposium, ISVC 2005, Lake Tahoe, NV, USA, December 5-7, 2005, Proceedings, ISVC, pages 601–606. Springer, 2005.

[Levoy and Hanrahan, 1996] Marc Levoy and Pat Hanrahan. Light field rendering. In *ACM Transactions on Graphics*, SIGGRAPH '96, pages 31–42, New York, NY, USA, 1996. ACM.

[Lewis et al., 2000] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *ACM Transactions on Graphics*, SIGGRAPH '00, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[Li et al., 2004] Ming Li, Marcus Magnor, and Hans-Peter Seidel. A hybrid hardware-accelerated algorithm for high quality rendering of visual hulls. In *Graphics Interface*, GI '04, pages 41–48. Canadian Human-Computer Communications Society, 2004.

[LiberoVision, 2012] LiberoVision, 2012. http://www.liberovision.com.

[Lipski et al., 2009] Christian Lipski, Christian Linz, Kai Berger, and Marcus Magnor. Virtual video camera: image-based viewpoint navigation through space and time. In *ACM Transactions on Graphics*, SIGGRAPH '09, pages 93:1–93:1, New York, NY, USA, 2009. ACM.

[Lowe, 1999] David G. Lowe. Object Recognition from Local Scale-Invariant Features. In *International Conference on Computer Vision*, ICCV '99, pages 1150–1157 vol.2, Los Alamitos, CA, USA, August 1999. IEEE Computer Society.

[Mahajan et al., 2009] Dhruv Mahajan, Fu-Chung Huang, Wojciech Matusik, Ravi Ramamoorthi, and Peter Belhumeur. Moving gradients: a path-based method for plausible image interpolation. In *ACM Transactions on Graphics*, SIGGRAPH '09, pages 42:1–42:11, New York, NY, USA, 2009. ACM.

[Marr and Poggio, 1979] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 204(1156):301–328, 1979.

[Matusik and Pfister, 2004] Wojciech Matusik and Hanspeter Pfister. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *ACM Transactions on Graphics*, SIGGRAPH '04, pages 814–824, New York, NY, USA, 2004. ACM.

[Matusik et al., 2000] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *ACM Transactions on Graphics*, SIGGRAPH '00, pages 369–374, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[Menache, 1999] Alberto Menache. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1999.

[MiddleburyMVS, 2012] Middlebury multi-view stereo evaluation, 2012. http://vision.middlebury.edu/mview/.

[MiddleburyS, 2012] Middlebury stereo evaluation, 2012. http://vision.middlebury.edu/stereo/.

[Miller and Hilton, 2006] Gregor Miller and Adrian Hilton. Exact view-dependent visual hulls. In *Proceedings of the 18th International Conference on Pattern Recognition - Volume 01*, ICPR '06, pages 107–111, Washington, DC, USA, 2006. IEEE Computer Society.

[Moeslund and Granum, 2001] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81:231–268, March 2001.

[Moeslund et al., 2006] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104:90–126, November 2006.

[Mori and Malik, 2002] Greg Mori and Jitendra Malik. Estimating human body configurations using shape context matching. In *Proceedings of the 7th European Conference on Computer Vision-Part III*, ECCV '02, pages 666–680, London, UK, UK, 2002. Springer-Verlag.

[Mori and Malik, 2006] Greg Mori and Jitendra Malik. Recovering 3D human body configurations using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28:1052–1062, July 2006.

[Mori et al., 2004] G. Mori, Xiaofeng Ren, A.A. Efros, and J. Malik. Recovering human body configurations: combining segmentation and recognition. In *Computer Vision and Pattern Recognition (CVPR), 2004 IEEE Conference on*, volume 2, pages II–326 – II–333 Vol.2, June 2004.

[Mori, 1970] Masahiro Mori. Bukimi no tani [The uncanny valley]. *Energy*, 7(4):33–35, 1970.

[Mori, 2005] Greg Mori. Guiding model search using segmentation. In *International Conference on Computer Vision*, volume 2 of *ICCV '05*, pages 1417–1423, 2005.

[Mor, 1978] Jorge Mor. The Levenberg-Marquardt algorithm: Implementation and theory. In *Lecture Notes in Mathematics*, volume 630, pages 105–116. Springer, 1978.

[Müller, 2009] Marcel Müller. *A 3D TV Application for Sports*. Institute of Visual Computing, ETH Zurich, 2009.

[NASA, 2009] NASA. Anthropometry and biomechanics, 2009. http://msis.jsc.nasa.gov/sections/section03.htm.

[Oggier et al., 2005] Thierry Oggier, Mike Stamm, Matthias Schweizer, and Jörn Pedersen. User manual swissranger 2 rev. b. version 1.02, 2005.

[Park and Hodgins, 2006] Sang Il Park and Jessica K. Hodgins. Capturing and animating skin deformation in human motion. In *ACM Transactions on Graphics*, SIGGRAPH '06, pages 881–889, New York, NY, USA, 2006. ACM.

[Park and Ramanan, 2011] Dennis Park and Deva Ramanan. N-best maximal decoders for part models. In *International Conference on Computer Vision*, ICCV '11, pages 2627–2634. IEEE, 2011.

[Petit et al., 2010] Benjamin Petit, Jean-Denis Lesage, Clément Menier, Jérémie Allard, Jean-Sébastien Franco, Bruno Raffin, Edmond Boyer, and François Faure. Multicamera real-time 3D modeling for telepresence and remote collaboration. *International Journal of Digital Multimedia Broadcasting*, 2010.

[Ramanan and Forsyth, 2003] Deva Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *Computer Vision and Pattern Recognition (CVPR), 2003 IEEE Conference on*, volume 2, pages II–467–II–474 vol.2, June 2003.

[Ramanan et al., 2005] Deva Ramanan, D. A. Forsyth, and Andrew Zisserman. Strike a pose: tracking people by finding stylized poses. In *Computer Vision and Pattern Recognition (CVPR), 2005 IEEE Conference on*, volume 1, pages 271 – 278 vol. 1, June 2005.

[Ren and Malik, 2003] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *International Conference on Computer Vision*, ICCV '03, pages 10–, Washington, DC, USA, 2003. IEEE Computer Society.

[Sand et al., 2003] Peter Sand, Leonard McMillan, and Jovan Popović. Continuous capture of skin deformation. In *ACM Transactions on Graphics*, SIGGRAPH '03, pages 578–586, New York, NY, USA, 2003. ACM.

[Scharstein and Szeliski, 2002] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, April 2002.

[Scharstein and Szeliski, 2003] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition (CVPR), 2003 IEEE Conference on*, volume 1, pages I–195 – I–202 vol.1, June 2003.

[Schreiner et al., 2004] John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. Inter-surface mapping. In *ACM Transactions on Graphics*, SIGGRAPH '04, pages 870–877, New York, NY, USA, 2004. ACM.

[Seitz and Dyer, 1996] Steven M. Seitz and Charles R. Dyer. View morphing. In *ACM Transactions on Graphics*, SIGGRAPH '96, pages 21–30, New York, NY, USA, 1996. ACM.

[Serra, 1983] Jean Serra. *Image Analysis and Mathematical Morphology*. Academic Press, Inc., Orlando, FL, USA, 1983.

[Shakhnarovich et al., 2003] Gregory Shakhnarovich, Paul A. Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *International Conference on Computer Vision*, ICCV '03, pages 750–759, 2003.

[Shotton et al., 2011] Jamie Shotton, Andrew W. Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1297–1304, June 2011.

[Sigal et al., 2004] Leonid Sigal, Sidharth Bhatia, Stefan Roth, Michael J Black, and Michael Isard. Tracking loose-limbed people. In *Computer Vision and Pattern Recognition (CVPR), 2004 IEEE Conference on*, volume 1, pages 421–428, Los Alamitos, CA, USA, June 2004.

[Sinha et al., 2009] Sudipta N. Sinha, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *International Conference on Computer Vision*, ICCV '09, pages 1881–1888, 2009.

[Siu and Lau, 2004] Angus M.K. Siu and Rynson W.H. Lau. Image-based modeling and rendering with geometric proxy. In *Proceedings of the 12th annual ACM international conference on Multimedia*, MULTIMEDIA '04, pages 468–471, New York, NY, USA, 2004. ACM.

[Skycam, 2012] Skycam, 2012. http://www.skycam.tv.

[Spidercam, 2012] Spidercam, 2012. http://www.spidercam.net.

[Stich et al., 2008] Timo Stich, Christian Linz, Georgia Albuquerque, and Marcus Magnor. View and time interpolation in image space. *Computer Graphics Forum (Proceedings of Pacific Graphics)*, 27(7):1781–1787, February 2008.

[Sullivan and Carlsson, 2002] Josephine Sullivan and Stefan Carlsson. Recognizing and tracking human action. In *Proceedings of the 7th European Conference on Computer Vision-Part I*, ECCV '02, pages 629–644, London, UK, UK, 2002. Springer-Verlag.

[Taneja et al., 2010] Aparna Taneja, Luca Ballan, and Marc Pollefeys. Modeling dynamic scenes recorded with freely moving cameras. In *ACCV*, volume 6494 of *Lecture Notes in Computer Science*, pages 613–626. Springer, 2010.

[Theobalt et al., 2004] Christian Theobalt, Edilson de Aguiar, Marcus A. Magnor, Holger Theisel, and Hans-Peter Seidel. Marker-free kinematic skeleton estimation from sequences of volume data. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '04, pages 57–64, New York, NY, USA, 2004. ACM.

[Thomas, 2006] Graham A. Thomas. Real-time camera pose estimation for augmenting sports scenes. In *Conference on Visual Media Production*, pages 10–19, 2006.

[Thomas, 2007] Graham A. Thomas. Real-time camera tracking using sports pitch markings. *Journal of Real-Time Image Processing*, 2(2-3):117–132, 2007.

[Tola et al., 2008] Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. In *Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on*, pages 1 –8, June 2008.

[Tola et al., 2010] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32:815–830, May 2010.

[Tuytelaars and Mikolajczyk, 2008] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Found. Trends. Comput. Graph. Vis.*, 3:177–280, July 2008.

[Urtasun and Darrell, 2008] R. Urtasun and T. Darrell. Sparse probabilistic regression for activity-independent human pose inference. In *Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on*, pages 1 –8, June 2008.

[Vicon, 2012] Vicon, 2012. http://www.vicon.com.

[Vlasic et al., 2008] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. In *ACM Transactions on Graphics*, SIGGRAPH '08, pages 97:1–97:9, New York, NY, USA, 2008. ACM.

[Waschbüsch et al., 2007] Michael Waschbüsch, Stephan Würmlin, and Markus Gross. 3D video billboard clouds. *Computer Graphics Forum*, 26(3):561–569, 2007.

[Yamazaki et al., 2002] Shuntaro Yamazaki, Ryusuke Sagawa, Hiroshi Kawasaki, Katsushi Ikeuchi, and Masao Sakauchi. Microfacet billboarding. In *Proceedings of the 13th Eurographics workshop on Rendering*, EGRW '02, pages 169–180, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[Yamazaki et al., 2005] Shuntaro Yamazaki, Katsushi Ikeuchi, and Yoshihisa Shinagawa. Plausible image matching: determining dense and smooth mapping between images without a priori knowledge. *IJPRAI*, 19(4):565–583, 2005.

[Yang and Ramanan, 2011] Yi Yang and Deva Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1385 –1392, June 2011.

[Zach et al., 2007] Christopher Zach, Thomas Pock, and Horst Bischof. A globally optimal algorithm for robust tv-l1 range image integration. In *International Conference on Computer Vision*, ICCV '07, pages 1–8. IEEE, 2007.

[Zitnick et al., 2004] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. In *ACM Transactions on Graphics*, SIGGRAPH '04, pages 600–608, New York, NY, USA, 2004. ACM.

Curriculum Vitae

Marcel Germann

Personal Data

March 28, 1980	Born in Schlieren, Switzerland
Nationality	Swiss

Education

Mar., 2012	Ph.D. defense.
Sep. 2007 – Dec. 2011	Research assistant and Ph.D. student at the Computer Graphics Laboratory of the Swiss Federal Institute of Technology (ETH) Zurich, Prof. Markus Gross.
Feb. 2007	Diploma degree in Computer Science.
Oct. 2000 – Feb. 2007	Diploma Studies of Computer Science, ETH Zurich, Switzerland. Specialization: Computational Sciences

Awards

May 2011	Best Paper Award "Body Pose Estimation in Uncontrolled Environments" at 3DIMPVT 2011.

Scientific Publications

M. GERMANN, T. POPA, R. KEISER, R. ZIEGLER, and M. GROSS. Novel-View Synthesis of Outdoor Sport Events Using an Adaptive View-Dependent Geometry. To appear in *Proceedings of Eurographics (Cagliari, Italy, May, 2012), Computer Graphics Forum, vol. 31, no. 2.*

M. GERMANN, T. POPA, R. ZIEGLER, R. KEISER, and M. GROSS. Space-time Body Pose Estimation in Uncontrolled Environments. In *Proceedings of 3DIMPVT (Hangzhou, China, May, 2011), (Best Paper Award).*

M. GERMANN, A. HORNUNG, R. KEISER, R. ZIEGLER, S. WÜRMLIN, and M. GROSS. Articulated Billboards for Video-based Rendering. In *Proceedings of Eurographics (Norrköping, Sweden, May, 2010), Computer Graphics Forum, vol. 29, no. 2, pp. 585-594.*

I. K. PARK, M. GERMANN, M. D. BREITENSTEIN, and HANSPETER PFISTER. Fast and Automatic Object Pose Estimation for Range Images on the GPU. In *Machine Vision and Applications, 2010, vol. 21, no. 5, pp. 749-766.*

G. GUENNEBAUD, M. GERMANN, and M. GROSS. Dynamic Sampling and Rendering of Algebraic Point Set Surfaces. In *Proceedings of Eurographics (Crete, Greece, April, 2008), Computer Graphics Forum, vol. 27, no. 2, pp. 653-662.*

M. GERMANN, M. D. BREITENSTEIN, I. K. PARK, and HANSPETER PFISTER. Automatic Pose Estimation for Range Images on the GPU. In *Proceedings of the Sixth International Conference on 3D Digital Imaging and Modeling, 3DIM, (Montreal, Canada, August, 2007).*

Employment

Sep. 2007 – Dec. 2011	Research assistant at ETH Zurich, Zurich, Switzerland.
Apr. 2006 – Oct. 2006	Internship at Mitsubishi Electric Research Laboratories, Cambridge, USA.