FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Silvain Bériault

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Multi-Camera System Design, Calibration and 3D Reconstruction for Markerless Motion Capture

TITRE DE LA THÈSE / TITLE OF THESIS

Prof. P. Payeur

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Prof. E. Dubois

Prof. P.Bose

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Multi-Camera System Design, Calibration and 3D Reconstruction for Markerless Motion Capture

by:

Silvain Bériault

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements for the degree of
Masters of Applied Science – Electrical Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-50855-8
Our file Notre référence
ISBN: 978-0-494-50855-8

Multi-Camera System Design, Calibration and 3D Reconstruction for Markerless Motion Capture

M. A. Sc. Thesis – Abstract

Silvain Bériault

Recently, significant advances have been made in many sub-areas regarding the problem of markerless human motion capture. However, markerless solutions still tend to introduce major simplifications, especially in early stages of the process, that temper the robustness and the generality of any subsequent modules and, consequently, of the whole application. This thesis concentrates on improving the aspects of multi-camera system design, multi-camera calibration and shape-from-silhouette volumetric reconstruction. In Chapter 3, a thoughtful system analysis is first proposed with the objective of achieving an optimal synchronized multi-camera system. Chapter 4 proposes an easy-to-use multi-camera calibration technique to estimate the relative positioning and orientation of every camera with sub-pixel accuracy. In Chapter 5 a robust shape-from-silhouette algorithm, with precise voxel coloring, is developed. Overall, the proposed framework is successful to reconstruct various 3D human postures and, in particular, complex and self-occlusive pianist postures in real-world (minimally constrained) scenes.

Abstract

Markerless motion capture, in contrast to marker-based gesture analysis, consists of extracting high-level human body kinematic information using passive vision only. The goal of markerless motion capture is to remove severe cumbersomeness issues regarding current commercial, and state-of-the-art, marker-based systems. In particular, marker-based solutions require the wearing of uncomfortable markers, which can interfere with natural gesture of performers. On the counter-part, markerless solutions lack robustness and tend to impose unacceptable constraints into the working environment for many applications. Currently, these limitations prevent markerless motion capture to replace marker-based systems in most real-world markets. Addressing these issues, the thesis presents a framework for markerless motion capture with specific interest on increasing the robustness of early topics within the chain of modules that compose such an application. In particular the topics of multi-camera system design, multi-camera calibration and volumetric reconstruction and coloring are studied.

The main goal of the project is to perform a complete system design analysis that will lead to the development of a reconfigurable synchronized multi-camera system which is optimized for the specific application of motion capture. A solid high-level software framework for multi-view application is also developed and is effective to encapsulate low-level interactions with the camera hardware. The designed acquisition setup is calibrated using a convenient framework for multi-camera calibration that allows free camera positioning. The proposed multi-camera calibration approach is able to reach precision up to an average reprojection error below ½ pixel. The use of a novel and reconfigurable dual-marker target is proposed to achieve complete calibration with no scale factor ambiguity (i.e. metric calibration). The full registration of all cameras composing the network enables shape-from-silhouette volumetric reconstruction using voxel data. The proposed implementation is effective at computing the binary voxel occupancy information even in the presence of imperfect silhouette data. Beyond voxel occupancy computation, foreground voxels are also augmented with color texture. The

color information contained in the multiple streams of video is effectively mapped onto the voxel data with proper visibility test to detect situations of occlusion. Experimental results presented at the end of this thesis demonstrate successful 3D human body reconstruction with adequate accuracy for motion capture. The example of motion capture for piano-playing performance evaluation is used to show the capability of the proposed framework to effectively reconstruct complex, self-occlusive, human postures.

Acknowledgements

I would like to thank my thesis supervisor, Dr. Pierre Payeur, for his guidance and assistance throughout my studies and my research. I would also like to thank the University of Ottawa, the Natural Sciences and Engineering Research Council of Canada and the Piano Pedagogy Research Laboratory for their financial and resource contributions. Special thanks go to the musicians who kindly accepted to participate to the motion capture experimentations. Finally, I would like to thank my family and friends for their support and encouragement.

Table of contents

List of Figures

List of Tables

Chapter 1. Introduction

Human motion capture is fundamentally concerned with the understanding and the analysis of human gesture. It consists of analyzing the evolution of the human posture over time. Initial attempts at motion capture were performed manually, simply by playing-back a video signal featuring a human performer. Video playback is still in use for numerous applications, especially in pedagogical setups. However, with recent advances related to digital sensors and computer technologies, it became strongly desirable to automate motion capture such that the position and the angle of body joints, and even the speed and acceleration of limbs, would be detected automatically and with high accuracy.

Computer-based motion capture is of great interest in a wide variety of applications. Nowadays, it is extensively used by the entertainment industry. Indeed, it became usual, in the latest generations of animated movies and video games, that real human motion is digitally captured and mapped into animated characters, with a high degree of realism. In the field of biomedical engineering, human motion capture finds applications in gait analysis for rehabilitation and prevention of injuries. In this field, automatic extraction of the human posture is of great assistance because it allows information about all human joints to be monitored simultaneously, while a human physician could only monitor a few joints at a time. Furthermore, prompt recognition of human activities can be applied for various surveillance tasks. Classical surveillance systems are limited to detecting the presence or absence of one or many human subjects within a volume of interest. It is expected that the next generation of surveillance systems would have great interest in understanding the actual and evolving behavior of human subjects in order to automatically detect and react to those judged undesirable.

In this research project, we are particularly interested in developing a multi-camera system dedicated to monitor human gesture in the context of piano-playing performance evaluation and prevention of injuries. Indeed, repetitive movements performed by

professional pianists can lead to long-term injuries, when incorrectly performed. Russell [1] reports that 39% to 47% of adult pianists will develop Playing-Related Health Problems (PRHP) throughout their career. Russell also reports that as high as 17% of student pianists tend to develop PRHPs. Thus, providing computerized motion capture instrumentation to musicians is of great interest because it would provide more accurate, more complete and less subjective gesture analysis, which could potentially lead to preventive detection of false movements prior to the occurrence of serious injuries. Such an achievement could benefit to a large population as it is estimated that over 100 000 students graduate every year from musical schools within North-America [1].

Computer-based motion capture constitutes a complex system design problem because multiple camera sensors need to be registered both temporally and geometrically. Frames of video must be precisely synchronized to ensure time-consistency in the data acquired across all cameras and computers over the network. Furthermore, the data from all viewpoints need to be geometrically registered to enable adequate fusion of data in the 3D space. Moreover, the human body is highly deformable and, hence, can adopt a wide variety of postures with different levels of complexity. Therefore, the particular example of pianist posture estimation with a markerless motion capture system is challenging because of the problem of self-occlusion and the presence of a piano which limits the positioning of cameras.

1.1 Motivations

Besides manual motion capture, systems for computerized motion capture can be distinguished in two main categories: marker-based and markerless systems. Marker-based systems [2-7] are characterized by the fact that the performers must wear multiple markers in order to capture the various movements. Such markers are chosen to be easy to identify in space and can consist of magnetic trackers, reflective markers or light-emitting devices (LEDs). For example, the Vicon Peak™ system [5] uses reflective markers, which are identified accurately using specially designed camera hardware.

Those markers are manually associated with specific body joints. This greatly simplifies the actual human posture estimation and thus the resulting human kinematics estimation is usually very reliable. In general, marker-based solutions are more robust than markerless solutions and they can operate at very high frame rates. They are particularly appealing to the cinema industry, but are also commonly used for advanced biomedical research. On the counter-part, marker-based solutions also admit several drawbacks. They typically require very specialized and high-cost equipment. It is time consuming to install many markers on each performer. Furthermore, wearing such markers can be cumbersome, uncomfortable, and can interfere with natural motion of the performers. In the particular application of interest, it is predictable that pianists may tend to modify their motion in order to compensate for some discomfort introduced by markers installed on wrists, finger tips, neck or face.

Markerless solutions attempt to remove those constraints by using solely passive vision for gesture monitoring. Markers are completely avoided and specialized camera hardware equipment is replaced by standard, off-the-shelf, color cameras. Unfortunately, current markerless systems are very limited and not yet ready for commercialization because of their lack of robustness, reliability and versatility. As it will be detailed in Chapter 2, current markerless systems impose severe constraints both on the clothing of the performers and on the content of the background. Furthermore, these systems are mostly limited to the detection of a subset of simple and non-occlusive human postures.

This thesis presents the design and implementation of a new markerless human motion capture system. The main motivation of this research project is to increase the overall robustness of markerless systems in order to facilitate their deployment in real world applications. In particular, the purpose of this work is to remove several constraints regarding the complexity of the working environment as well as the versatility of the reconstructed 3D human body postures. The implemented system is primarily dedicated to the monitoring of pianist postures, but is not restrained, in any way, to this sole application.

1.2 Objectives

The implementation of a robust markerless human motion capture system can be decomposed into several distinct modules, as illustrated in Figure 1.1. Obviously, the first topic is the elaboration of a physical multi-camera system which is designed to uniformly surround a bounded working environment and which will serve the purpose of gathering synchronized video data. The completion of this module enables multi-camera calibration and silhouette extraction which both require synchronized video recording capabilities. The multi-camera calibration task is a special initialization procedure which serves the purpose of registering, in 3D, the positioning of all camera sensors. The purpose of the silhouette extraction procedure is to decompose, frame-by-frame, the content of the synchronized video sources to separate pixels that pertains to the subject of interest from those that belong to the background. The volumetric reconstruction module consists of combining silhouette data and calibration data to obtain a consistent 3D model of the performer. This reconstructed model is used as the main cue to extract higher-level information about the human posture. This is, in fact, the objective of the human pose estimation and of the human gesture tracking and analysis modules.

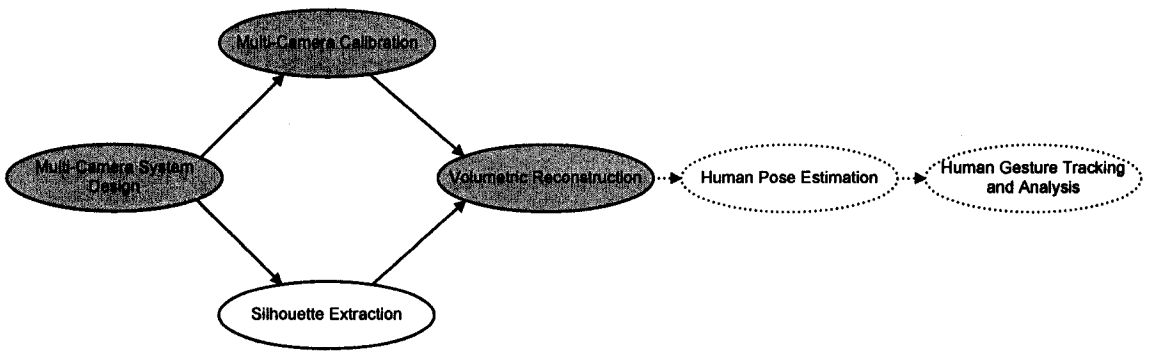


Figure 1.1. Various modules related to the design of a markerless human motion capture system

The work presented in this thesis concentrates on early modules, within the entire chain of activities, which are highlighted in gray in Figure 1.1. The silhouette extraction procedure is not highlighted because this task was performed concurrently by a colleague [8-10] and is not the focus of this thesis. As it will be established in Chapter 2, we can

denote, from the literature, that many simplifications are introduced in those areas, which temper the robustness and the generality of any subsequent modules and, consequently, of the entire application. In order to alleviate these limitations, the main objectives of the present work include:

- A complete formal design and implementation of a reconfigurable multi-camera system. This work also results in the design of a solid software framework for synchronized multi-camera applications that encapsulates and handles low-level interactions with the camera hardware.

- The development of a user-friendly framework for multi-camera calibration, which is resolved using a weighted camera graph and which results in the precise registration of a variable number of cameras to a common global coordinate system with an estimated accuracy below ½ pixel.

- The development of a shape-from-silhouette reconstruction algorithm extended with effective voxel coloring. The outcome of this work is the binary occupancy classification of each voxel that subdivides the working environment into either "background" or "foreground" with proper compensation for imperfection in 2D silhouette (segmented) images. The color information from all views is then analyzed and combined, with adequate occlusion detection, to determine a final color for each foreground voxel. The designed algorithm, while being deterministic (not iterative), provides suitable voxel coloring for the purpose of motion capture, even in presence of highly self-occluded human postures.

- The elaboration and formalization of a solid framework for markerless motion capture. Combining the robust technologies developed within this work results in the removal of a specific set of constraints related to the performer and to the working environment. Overall, this framework provides effective foundations for

versatile markerless motion capture therefore moving a step forward towards the deployment of markerless systems for real-world applications.

1.3 Thesis Outline

This thesis is organized as follows: Chapter 2 presents a survey of the most recent advances in the field of motion capture with special emphasis on the problems of multi-camera system design, calibration and volumetric reconstruction. Chapter 3 presents a complete analysis of multi-camera system design which addresses various topics ranging from adequate camera hardware selection to the implementation of a robust software framework for synchronized multi-video acquisition and processing. This exhaustive analysis is performed with the objective of providing higher quality input video data to the subsequent modules composing the system. In Chapter 4, the 3D registration of all cameras is estimated using a systematic framework for multi-camera calibration. The obtained calibration data is then used, in Chapter 5, to merge multiple synchronized video sources of a performer into consistent and colored 3D voxel models. Chapter 6 revisits the major topics discussed in this thesis with special emphasis on contributions and enhancements brought in specific areas within the field of markerless motion capture.

Chapter 2. Literature Review

Computerized motion capture has been of great interest over the last decade to overcome obvious drawbacks related to manual analysis of human gesture. However, computerized motion capture and, in particular, markerless motion capture still admits several flaws. As a result, this field of research remains open. This chapter presents a review of the latest advances in the field of motion capture and is subdivided as follows: The first section presents a global overview of the various technologies used for motion capture. The second section provides details about recent markerless motion capture systems since the aim of this project is the development of a system based solely on passive vision technologies. The following three sections provide reviews related to the three main topics of interest studied in this work: multi-camera system design, multi-camera calibration and volumetric reconstruction.

2.1 Overview of Motion Capture Applications

Methods for motion capture can be subdivided into three main categories: manual, marker-based, and markerless motion capture. Manual motion capture is the simplest form of gesture analysis and has been used for several years. Over the last decade, many marker-based systems for motion capture have been successfully commercialized. The current status of markerless motion capture remains mainly experimental and, to the best of our knowledge, no markerless system can fulfill all requirements needed for the vast majority of real-world applications.

2.1.1 Manual Motion Capture

Manual motion capture is the most simplistic form of motion capture. It consists of recording a video of a performer which is then analyzed manually by a human operator.

This form of motion capture is not optimal because the analysis is subjective and can differ from one operator to another. Furthermore, there is a loss of information because data acquired through an optical camera consists of a 2D perspective representation of a 3D reality. Nevertheless, manual motion capture remains useful in sports, for example to analyze, in very slow-motion, the swing of professional golfers or the service routine of tennis players. Commercial software, for example the Dartfish ProSuite™ application [11], has been designed to assist in performing such tasks.

2.1.2 Commercial Marker-Based Motion Capture

Marker-based systems constitute the state-of-the-art for human motion capture in real-world applications. Marker-based motion capture provides higher-level kinematics information inferred by detecting the position, in full 3D space, of multiple markers worn by a performer. To identify the position of these markers, multiple technologies have been used: magnetic trackers [2-4], optical markers [5-7] or even special phosphorescent makeup [12].

Magnetic tracking systems require a performer to wear multiple receiver sensors (markers). A stationary sensor (transmitter) is used to generate magnetic fields using a coil of wire. On the receiver side, the magnetic fields are measured and that resulting signal serves at estimating the position of the marker with respect to the transmitter. Magnetic tracking is advantageous because it is a low-cost solution and because the problem of occlusion is inexistent. Indeed, magnetic fields can travel across the human body without problem. However, magnetic tracking also admits severe drawbacks which limit its range of applications. Magnetic tracking is very sensitive to measurement errors and, in particular, to interference with metallic objects and electronic devices commonly present in real-world environments. Furthermore, magnetic tracking is very cumbersome because of the relatively large size of markers attached to the human body and because of the cabling required to report the measurements back to a control station. Nowadays, wireless markers are emerging, therefore partially removing the cumbersomeness

constraint (cabling is avoided) [3, 4]. Finally, the number of markers is physically limited, in hardware, thus reducing the number of points used for human kinematic estimation. For example, the Liberty Latus™ system [4] is limited to a maximum of 12 markers.

Many issues related to magnetic tracking are resolved with the use of optical markers. The Vicon Peak™ [5] and the Motion Captor™ [6] solutions both utilize optical reflective markers. Absolutely no cabling to the performer is required. Reliable tracking in the 3D space is assured using multiple instances of specialized and accurately calibrated high-cost camera hardware. The number of markers in use is scalable and is not limited in hardware. Furthermore, the size of markers is generally smaller in comparison to magnetic sensors. Thus, several feature points can be used for kinematics modeling. However, the wearing of markers remains uncomfortable and lengthy setup time is required prior to the motion acquisition. In order to reduce the overall application cost, the Phase Space Impulse™ system [7] utilizes light-emitting device (LED) markers fixed to a highly contrasting black costume worn by the performer. Multiple calibrated color cameras with very high frame rate are used to track the markers. However, wiring is required to activate the LEDs thus reducing the mobility of the performer.

With the use of either magnetic or optical markers, only a few points pertaining to the human body are reconstructed in 3D. This number is limited by the amount of markers installed on the performer. The Mova® Contour™ [12] innovates in replacing markers with a special phosphorescent makeup (for skin) and phosphorescent powder (for clothing) that glows in the dark. Over 100 thousands points can be collected such that facial expressions are captured with a very high degree of realism. This implementation is a first step towards markerless systems. However, it should be noted that wearing such a glowing makeup is only useful in indoor environments where lighting is precisely controlled and can be toggled (on/off) electronically at a very high frequency. Furthermore, wearing makeup is cumbersome especially for physically intensive activities and thus cannot be used, for example, to monitor the gesture of professional athletes. Finally, their proposed setup requires an expensive multi-camera

network containing over 40 cameras that operate at very high frame rate (>100 fps). The main application of this technology remains limited to movie making and computer gaming.

2.1.3 Markerless Human Motion Capture

Markerless gesture analysis is performed solely using passive vision technologies and is significantly advantageous because it removes the cumbersomeness of installing and wearing multiple markers on human performers. However, important robustness issues, which will be analyzed in depth in section 2.2, still prevent markerless gesture monitoring solutions from replacing marker-based systems in most real-world applications. A few commercial markerless products are starting to emerge but their application remains limited to very controlled working environment (i.e. movie making and computer gaming). For example, the markerless motion capture demo proposed by Organic Motion® [13] is constrained to a rather unrealistic and empty white room.

In general, research regarding markerless motion capture is subdivided in two main paradigms. Initial attempts consisted of fitting a 3D articulated model of the human body by projecting it in multiple image views [14, 15]. For example, in Delamarre *et al.*'s implementation [15], human kinematics was found by projecting and fitting such a 3D articulated model until it projects within the real human silhouette in every view simultaneously. Throughout this discussion, the term "human silhouette" corresponds to a 2D surface, within an image, that includes all the pixels pertaining to the targeted performer.

Over the years (2000 to present), an inverse paradigm has arisen. Instead of projecting a 3D articulated model into multiple image views, the content of these images are combined into a consistent 3D model used to infer human body kinematics information. Human silhouette data, in every view, is typically the main cue used to reconstruct the 3D shape of a performer. This procedure is referred as shape-from-

silhouette and consists of intersecting multiple image silhouettes, by back-projecting them into the 3D space, to obtain a reconstructed 3D volume. The reconstructed volume is usually represented using voxel data which is the logical Cartesian extension of a pixel in 3D. Almost all recent markerless applications [16-23] adopt this paradigm because it simplifies the kinematic analysis and because it yields to a less restrictive human posture analysis. With the latest advances related to computer technologies, some of these implementations even perform in real-time [18, 22].

2.2 Analysis of Markerless Human Motion Capture Applications

The goal of this research project is the development of a markerless motion capture application to remove the need for subjective inspections of the captured motion (manual playback of a video stream) and to remove the cumbersomeness and high-cost issues that make marker-based solutions inadequate for a variety of applications. In the previous section, two paradigms were proposed related to markerless solutions. Our implementation utilizes the shape-from-silhouette paradigm because it is well acknowledged to offer better potential for full 3D human kinematics extraction than the paradigm based on multiple images analysis. In this section, a high-level review of several recently developed markerless systems is presented. Special emphasis is put on specific flaws admitted by those implementations thus justifying the need to pursue research in this area.

Mikic *et al.* [16, 17] proposed a system to track the human body using binary shape-from-silhouette. The resulting shape is represented from the subdivision of the working volume into a set of small voxels which contain a binary occupancy classification state: background or foreground. Human posture estimation and tracking is fully automated and does not require any manual initialization in the sense that the performer is not required to adopt a specific starting posture. However several constraints are imposed by the system. The performer is required to wear tight clothing that must be of a distinct color. This constraint is used to simplify the extraction of

human silhouettes from the background. The working volume is totally empty and human postures are kept simple to remove any ambiguity problem that occurs with complex and self-occluding postures.

Cheung *et al.* [18] proposed a similar approach which is also based on binary voxels. Human silhouettes are computed using a background subtraction technique. A model of the background (the empty scene) is first gathered for each view. When the performer enters the volume of interest, the background model is subtracted from each image view such that pixels differing from the original model are classified as silhouette pixels. This removes the monochromatic clothing restriction but forces the background to remain static throughout the entire acquisition. The room's lighting also needs to remain constant throughout the video recording. As for the previous system, this implementation is also limited to simple human postures, but can operate in real-time. In a later iteration [19], color information is incorporated to the volumetric voxel model at the expense of removing the real-time functionality. Indeed, the voxel coloring scheme implemented for this system is highly demanding computationally, but it was found that color information can help resolving ambiguities within complex human postures.

The idea of incorporating color to volumetric models, in addition to the occupancy information, has been further investigated in the work of Kehl *et al.* [20, 21]. In their system, a fast voxel coloring scheme is proposed. It is not as precise as the coloring scheme of Cheung *et al.* [19], but it is less demanding computationally because it does not require multiple passes through the entire voxel data. In addition, image edges are incorporated as a third cue to increase the overall robustness of human posture analysis. However, no metrics are proposed to evaluate the beneficial impact of this third cue.

Very recently, Caillette [22] proposed a real-time motion capture implementation which utilizes both voxel occupancy and coloring cues. To achieve a real-time system, many tradeoffs had to be made which makes this system unsuitable for most real-world applications. Among other things, this system utilizes only a few unsynchronized camera sensors to reconstruct the full human body and their positioning is highly restrained

because they are registered using a simple and restrictive multi-camera calibration technique. The voxel coloring scheme is also less accurate than the scheme of Kehl *et al.* [20] and requires a manual initialization.

Voxel coloring is a neat and logical addition to voxel occupancy information. However, it remains unclear what is the limit of disambiguation that can be achieved through voxel coloring especially because of inherent variations in clothing styles: short versus long sleeves, dress versus shorts or pants, etc. Sundaresan [23] developed a volumetric segmentation scheme that transforms binary voxel data into the Laplacian Eigenspace. This scheme manages to separate and classify voxels in 6 major body parts: head, torso, left arm, right arm, left leg, and right leg. Results are very promising considering that only binary voxel data is used. However more testing is required especially with complex human body postures.

This analysis highlighted some drawbacks that are recurrent in all of these implementations and which prevent markerless solutions to make cumbersome marker-based applications obsolete. In particular, we denote that in all of these systems, many simplifications are imposed on the working environment to facilitate the recognition of the human body within full frames of video. Furthermore, current markerless systems can only capture simple human postures unlike marker-based systems. The purpose of this research project is to overcome specific flaws related to markerless systems by improving the robustness of a specific set of activities which are reviewed in depth in the upcoming sections.

2.3 Multi-Camera System Design for Motion Capture

Multi-camera system design is the first topic which is discussed in this thesis. The design of a multi-camera system is important because it can impact the quality of the input video data and thus has effects on all subsequent modules of the system. Surprisingly, this particular issue is often overlooked in most systems described in the

literature. Typically, this topic is only discussed in systems that claim to operate in real-time.

The original system of Cheung *et al.* [18] operates in real-time by distributing the work over several computer platforms. Each camera is connected to a separate computer node which has the obvious task of acquiring raw frames of video but also serves the purpose of extracting human silhouette information. Silhouette extraction is a single-view operation, which can be naturally parallelized onto several computational nodes for the case of multiple concurrent image views. Distributed binary silhouettes are efficiently compressed using bitmasks and transferred to a master computer. This master computer performs the actual silhouette reconstruction, the posture analysis and the displaying of results. Without the display unit, the system can support a frame rate of 16 frames per second (fps) which can be considered as acceptable for a subset of motion capture applications.

Doubek *et al.* [24] proposed a similar system implementation where silhouette extraction is performed on several computer nodes and is separated from the 3D analysis. In addition, the problem of inter-camera synchronization is discussed in depth. Frames of video are aligned asynchronously using a software trigger but that remains inaccurate due to communication latencies between the multiple acquisition nodes. All cameras are color calibrated in software [25] to ensure adequate uniformity among views and to obtain uniform voxel data coloring. This software calibration replaces the auto-calibration feature provided by most camera manufacturers because adjusting dynamically the calibration based on the evolving content of a scene can compromise many image processing algorithms and, in particular, almost all practical silhouette extraction implementations.

Unlike Cheung *et al.*'s [18] and Doubek *et al.*'s [24] systems, which are distributed over several computer nodes, Caillette [22] attempted to develop its real-time system using a single acquisition node. However, with the use of only one computer, the number of cameras is limited and cannot be extended. Unfortunately, volumetric reconstruction

using shape-from-silhouette requires enough camera views in order to be precise. Inconsistencies are also denoted in the execution time calculations provided in his work. Indeed, we can observe that each module of the proposed system, when treated separately, can operate in real-time at an acceptable frame rate (i.e. 15 fps). However, when all modules are concatenated together, the system is apparently not capable of supporting an acceptable frame rate. On the other hand, frame synchronization is avoided by the fact that all cameras are connected to a single acquisition node. Thus, the maximal variation in time (jitter) for a group of frames is bounded by the frame rate duration which can be attenuated by forcing cameras to operate at higher frame rate.

2.4 Multi-Camera Calibration

Camera calibration is an essential step in many computer vision algorithms and especially those that require the fusion of data acquired from multiple cameras or, equivalently, multiple viewpoints. This is strongly the case for applications dedicated to 3D human motion capture. Such applications require a multi-camera setup composed of at least 5 to 10 cameras to achieve a fairly complete reconstruction. The reconstruction of a reliable 3D model of the human body is only made possible assuming accurate knowledge about the pose of all cameras involved in the system. Camera calibration is the second major topic covered in this work. The following sections present a detailed analysis of current camera calibration methodologies.

2.4.1 Modeling the Behavior of Cameras

The goal of camera calibration is to allow a systematic mapping of any 3D point, in an arbitrary working volume, to its 2D projection, in the image plane of a camera. Before discussing the actual problem of camera calibration, a camera model first needs to be established in order to encapsulate the behavior of cameras into a finite set of parameters. Tsai [26] separated the camera behavior into two sets of parameters: the intrinsic and extrinsic camera parameters. Intrinsic parameters are used to model the actual

perspective transformation required to map any 3D point to its 2D camera projection. Extrinsic parameters are concerned about the estimation of the position and orientation (registration) of a camera with a global coordinate system.

Intrinsic Camera Parameters

To model the intrinsic camera behavior, Tsai [26] based his analysis on the perfect pinhole camera model, shown in Figure 2.1. A 3D frame (R_{cam}) is attached to the optical center of the camera and corresponds to the camera frame. A 2D frame (R_{im}) is attached at the intersection of the optical axis (Z_{cam}) with the image plane. This frame is then translated to the top-left corner of the image plane (R_{pixel}) such that projected points are expressed in pixel coordinates.

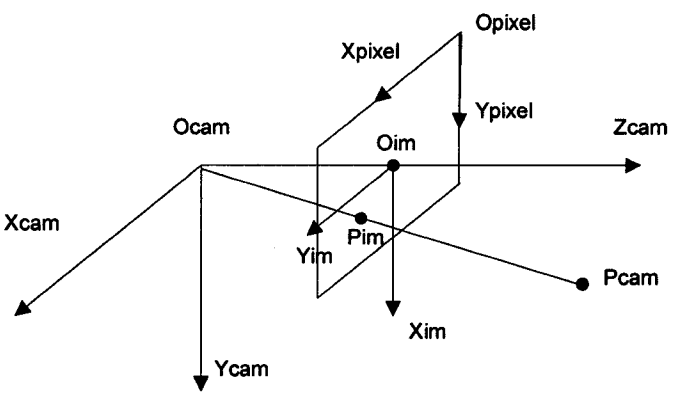


Figure 2.1. Camera model for intrinsic camera calibration

The perspective projection (P_{im}) of any 3D point (P_{cam}) can be found using 5 intrinsic camera parameters presented in the form of a 3x3 intrinsic matrix as per equation (2.1). In equation (2.1), f is the lens focal length, s_x and s_y are the effective pixel sizes and (O_x, O_y) is the pixel position of the image center (O_{im}).

$$K = \begin{bmatrix} \frac{f}{s_x} & 0 & O_x \\ 0 & \frac{f}{s_y} & O_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

With the knowledge of K, the perspective projection, P_{im}, can be found for P_{cam}, and expressed in pixel coordinates, using equation (2.2) followed by equation (2.3). Careful readers may notice a potential division by zero in the case where $x_3 \approx 0$. However, it is trivial to show that this case only happens if $z_{cam} \approx 0$. In such case, P_{cam} would be located behind the image plane and thus is irrelevant.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = K \begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{bmatrix} \tag{2.2}$$

$$P_{im} = \begin{bmatrix} x_{pixel} \\ y_{pixel} \end{bmatrix} = \begin{bmatrix} x_1/x_3 \\ x_2/x_3 \end{bmatrix} \tag{2.3}$$

Extrinsic Camera Parameters

Because the exact location of the optical center is a virtual point in space, the camera frame is intangible. For convenience, it is often desired to align the camera frame to a meaningful 3D coordinate system. The transformation linking a camera frame to a world frame is illustrated in Figure 2.2.

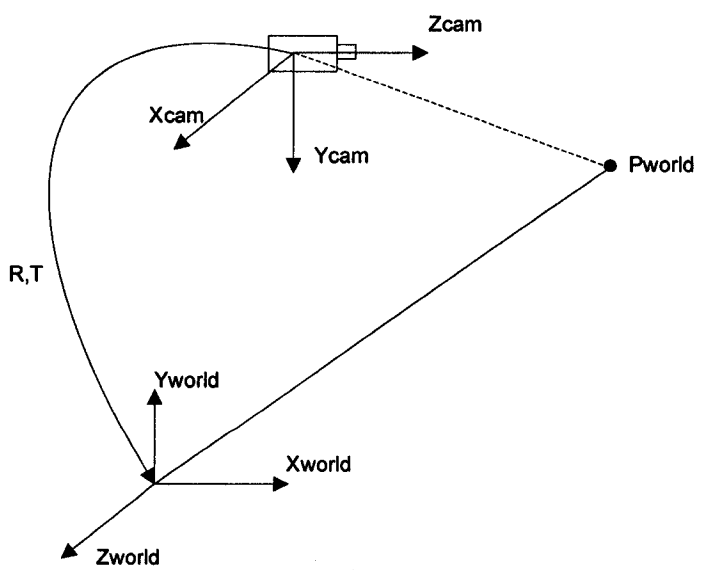


Figure 2.2. Extrinsic camera transformation graph

This 3D transformation is expressed using a 3x3 rotation matrix (R) and a 3x1 translation vector (T) that are referred as the extrinsic camera parameters:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.4}$$

A 3D point defined with respect to a world reference frame can be expressed with respect to the camera frame using:

$$P_{cam} = R \cdot P_{world} + T \tag{2.5}$$

In summary, any point defined with respect to a world reference frame is first transformed to be expressed with respect to the camera reference frame using the extrinsic R and T parameters and equation (2.5). Then, the transformed 3D point P_{cam}, is projected using the perspective transformation defined by the intrinsic matrix K and equation (2.2) followed by equation (2.3).

Lens Distortion and Other Extensions

Up to this point, extrinsic and intrinsic camera behaviors have been modeled and, therefore, any 3D point described with respect to a tangible 3D coordinate system can be first transformed and expressed with respect to an intangible camera frame and finally projected into the camera's image plane. This latter part is achieved assuming a perfect pinhole camera behavior. Unfortunately, this assumption is invalid since camera lenses introduce various forms of distortion. The most typical form of lens distortion is the radial distortion. Tsai [26] introduces, into his model, two additional parameters to compensate for the radial distortion in the x and y directions. More recent models also compensate for tangential distortion [27-29] which is caused by imperfect centering of the lens surfaces. Zhang [30] further extended the aforementioned camera model by the introduction of a skew coefficient used to define the angle between the two image axes. However, Bouguet [29] reports that rectangular pixel (zero skew) is an acceptable assumption nowadays.

Up to this point, the full camera behavior has been modeled by a set of intrinsic and extrinsic parameters as well as few lens distortion coefficients. The purpose of camera calibration consists of the development of automatic techniques to accurately estimate those parameters: the K matrix of equation (2.1), the R and T matrices of equation (2.4) and, generally two to six coefficients for lens distortion. In the literature, those techniques differ, by their complexity and methodology, according to the number of cameras to be calibrated.

2.4.2 The Calibration of a Single Camera

The full calibration of a single camera has been widely covered in the literature. The general idea is to collect the position of many 3D points and their corresponding position in the 2D image plane. Based on these matches, both the intrinsic and extrinsic parameters can be estimated. To acquire those matches, specially designed calibration patterns need to be built (see Figure 2.3). Both Tsai [26] and Zhang [30] used square landmarks as per Figure 2.3a. The corners of each square landmark are used as calibration points. Nowadays, disconnected square landmark are discouraged [29, 31] because they tend to erode in images taken with lenses that are slightly out-of-focus, therefore leading to inaccurate pixel position estimations. Heikkilä *et al.* [27] resolved this issue by using the center of circular landmarks, as shown in Figure 2.3b, as calibration points. However, detecting the actual center of a circular landmark, in an image, is not trivial because of deformations introduced by perspective projection and, thus, because of the distinction that needs to be made between the "center of a projected circle" and the "projection of a circle's center". A solution often recommended in practical implementations is the use of a checkerboard pattern [28, 29], as per Figure 2.3c, because corner extraction is simpler and checkerboard landmarks do not suffer from the erosion of disconnected square landmarks.

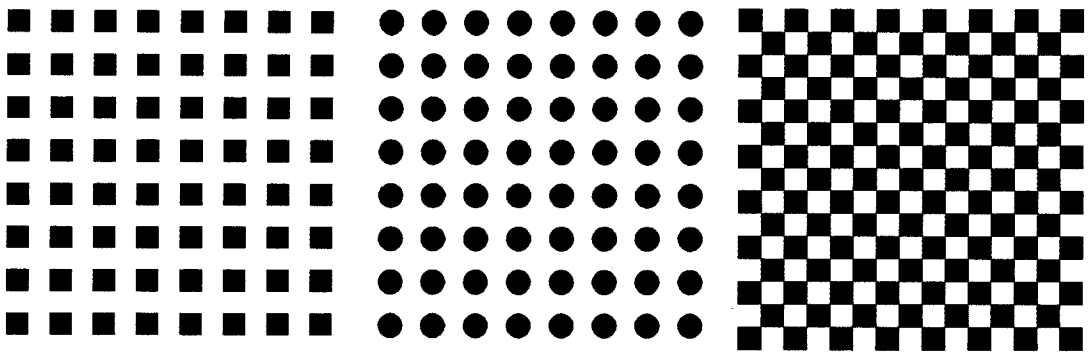


Figure 2.3. Example of 2D calibration patterns

Tsai [26] mentioned that a better accuracy is obtained when the calibration points are not all co-planar. This can be achieved using a 3D calibration structure (such as the calibration cube of Figure 2.4) but such structure is more complex to build and is not easily scalable.

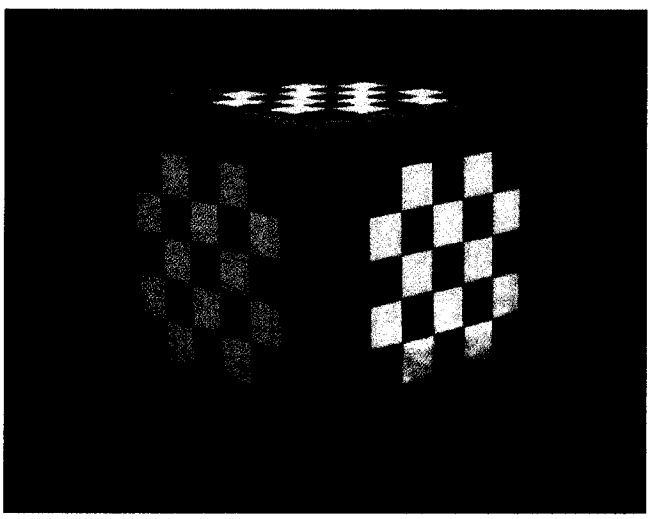


Figure 2.4. The design of a $(20\ cm)^3$ calibration cube

When only the intrinsic parameters need to be found, a very elegant solution is proposed again by Tsai [26] and later refined by Zhang [30]. Multiple views of a single 2D calibration pattern, under various poses, are taken and used for the calibration. While the extrinsic parameters will change between each view, the intrinsic parameters will remain constant. This approach provides very precise intrinsic parameter estimation because a large number of calibration points can be obtained simply by introducing additional views of the calibration pattern. Furthermore, the conception of a 2D

calibration pattern is more convenient than that of a 3D calibration pattern. If required, extrinsic parameters can be estimated as a supplementary step afterward, using one view of either a 2D or 3D calibration pattern.

2.4.3 The Calibration of Two Cameras

Similarly to the case of a single camera, it is possible to achieve full camera calibration of a two-camera setup using a classical calibration target. However, if the two cameras have a large baseline or large rotation, it may become difficult to fit a large calibration pattern in both camera views simultaneously [32]. Fortunately, most stereo-vision applications only require the relative positioning between two cameras, rather than the absolute alignment of the cameras with a tangible 3D coordinate system. Hartley and Zisserman [33] derived a technique to estimate the extrinsic relationship between two cameras using epipolar geometry and prior knowledge of the cameras intrinsic parameters (i.e. using Zhang's multi-frame approach independently on each camera). This technique consists of decomposing a Fundamental matrix, F, into a stereo rotation matrix and translation vector. The main advantage of this technique is that the F matrix between two views is easy to compute as it only requires image correspondences rather than matches between 3D world points and 2D image points. With a known F matrix, extrinsic calibration is extracted as follows:

1) The Essential matrix (E) is calculated from the F matrix using equation (2.6). K_1 and K_2 are the intrinsic matrices for the first and second cameras respectively. F_{12} is the F matrix linking the first camera to the second camera.

$$E = K_2^T \cdot F_{12} \cdot K_1^T \tag{2.6}$$

2) The E matrix is then decomposed using a Singular Value Decomposition (SVD) and has the following form:

$$E = U \cdot S \cdot V^T \tag{2.7}$$

In equation (2.7), U is the 3x3 left orthogonal matrix and V is the 3x3 right orthogonal matrix. The resulting 3x3 singular-value matrix (S) contains only two non-zero and equal singular values according to the properties of the essential matrix [34].

3) From the SVD of E, the extrinsic parameters for both cameras can be obtained with respect to the first camera. The extrinsic parameters for the first camera are simply an identity rotation matrix (no rotation) and a null translation vector (0,0,0). The extrinsic parameters for the second camera are found up to a four-fold ambiguity (two possible rotations and two possible translations) and up to a scale factor (unity translation vector) such that:

$$R_1 = UWV^T \text{ and } R_2 = UW^TV^T \tag{2.8}$$

$$T_1 = +u_3 \text{ and } T_2 = -u_3 \tag{2.9}$$

In equations (2.8) and (2.9), u_3 is the third column of U, and W is the following orthogonal matrix:

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

Mathematical proofs of this decomposition are provided in Hartley *et al.* [33], but are outside the scope of this discussion. In section 4.3.3 of this thesis, a method is developed to resolve this four-fold ambiguity. To resolve the scale factor ambiguity, an absolute measurement is required. For example, the absolute measured distance between two matching features in a scene can be used to resolve this scale ambiguity. A solution to this issue is proposed in section 4.3.7.

2.4.4 The Calibration of a Generic Multi-Camera System

The calibration of a generic multi-camera system is the most complex case. Intrinsic parameters computation remains simple because it can be performed independently for each camera by applying Zhang's multi-frame approach described at the end of section 2.4.2. However, precise extrinsic registration of multiple cameras is very challenging, especially when free camera positioning is permitted.

Complex calibration rigs have been proposed [22, 32, 35] but several drawbacks can be observed. Caillette [22] proposed to use a single 2D calibration rig to perform the calibration of his multi-camera system. While this calibration approach is simple, it is prone to inaccuracies because all calibration points are co-planar. It also requires all cameras to see the full calibration rig, which imposes major constraints on the positioning of the cameras. Moreover, during the calibration procedure, the working volume must be completely empty as the calibration rig will occupy most of the floor. In order to overcome the coplanarity constraint, Rander [35] suggested to use calibration bars mounted on tripods. The bars are translated vertically and horizontally in the working volume to obtain full coverage and non-coplanar points. This approach is however very cumbersome as it requires numerous manipulations. It also imposes an empty working volume. Drouin *et al.* [32] suggested a method that computes pair-wise camera relations using multiple views of a 2D calibration rig. Pair-wise relations are then unified into a consistent camera graph. This approach also eliminates the coplanarity limitation but lacks flexibility for cases where cameras are orthogonal or are separated by large baselines. It is however a first step towards novel approaches discussed in the next paragraph.

Because of important problems with classical methods for multi-camera calibration, new approaches were recently investigated [36-38]. The strategy in all of these approaches is to first find a coarse estimate of the cameras registration which is then refined through an iterative method. To obtain the initial estimate, all of these

methods use a single visible feature point as the calibration target. The latter is being waved in the workspace to create a cloud of 3D calibration points. Pair-wise relationships between cameras are estimated and then, relations linking all cameras to a reference camera are found. Chen *et al.* [36] applied an extended Kalman filter iteratively to perform the final optimization. However, more accurate results are obtained by Ihrke *et al.* [37] and by Svoboda *et al.* [38] who use a bundle adjustment [39] as the final optimization step.

2.5 Volumetric Reconstruction

The full calibration of a camera network enables shape-from-silhouette reconstruction and is the last aspect discussed in this thesis. This section introduces the concept and limitations of shape-from-silhouette reconstruction but also surveys recent advances regarding optimization and robustness.

2.5.1 Concept of Shape-From-Silhouette

Shape-from-silhouette is a technique commonly used to compute a voxel map based on multiple 2D images of a targeted foreground object (in our case, a human performer). The main advantage of using shape-from-silhouette is that only silhouette data is required. This technique does not require that a scene contains sufficient texture as in, for example, feature-based stereo analysis.

A high-level block diagram of the shape-from-silhouette method is presented in Figure 2.5. Color images of a performer are acquired synchronously from all cameras. In each color image, the silhouette is extracted from the background resulting in a binary image where pixels are either classified as "foreground", for those that pertain to the human performer, or "background" otherwise. Silhouettes are back-projected in the 3D space using the camera calibration data. The intersection of all silhouettes results in the

computation of the performer's visual hull, i.e. the volume occupied by the subject (in our case, the reconstructed human model).

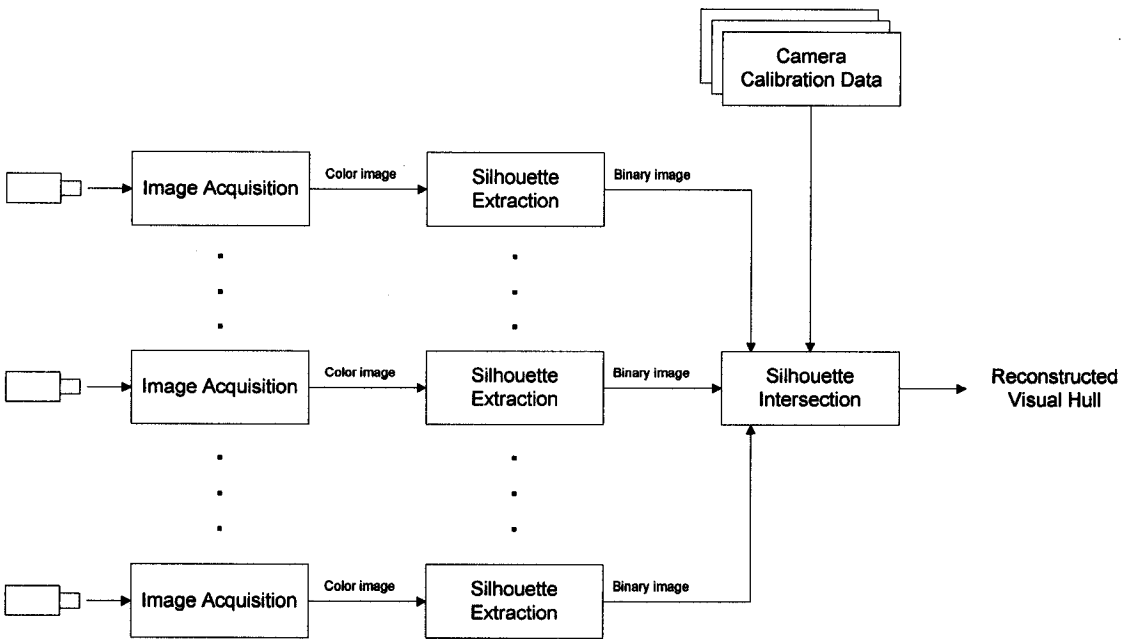


Figure 2.5. High-level block-diagram showing the major steps required for shape-from-silhouette reconstruction

In Figure 2.6a, a silhouette is back-projected in the 3D space to form a silhouette cone. In Figure 2.6b, two silhouette cones are intersected. Obviously, the intersection of only two silhouette cones is not sufficient to obtain a meaningful reconstruction. When more silhouette cones are added to the model, the resulting volume intersection is smaller and, thus, the reconstructed shape approaches the original shape. After a sufficient number of views are incorporated, and well balanced to surround the object of interest, adding extra views results in very minimal improvements to the visual hull reconstruction.

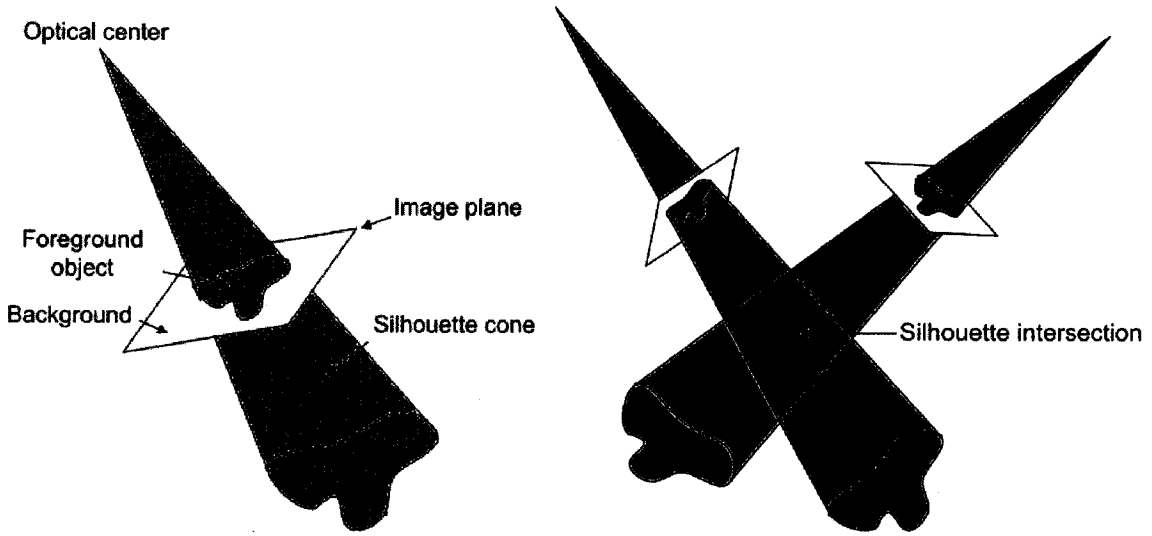


Figure 2.6. Basics of shape-from-silhouette volumetric reconstruction. (a) A silhouette cone is created by back-projecting all foreground pixels into the 3D space. (b) The object's visual hull is computed from the intersection of multiple silhouette cones of the same foreground object taken from different point of views.

2.5.2 Limitations of Shape-From-Silhouette

Laurentini [40] studied the problem of shape-from-silhouette reconstruction from a theoretical point of view. He formally defined the term visual hull as the closest approximation of a real object that can be obtained using shape-from-silhouette. This closest approximation is only achieved when an infinite number of viewpoints is used. However, Laurentini observed that even when all possible viewpoints are used, the object visual hull can still differ from the original object. Indeed, it was found that the visual hull of an arbitrary object is always a volume included within the range defined by the real object volume and its convex hull[1] as per equation (2.11). For the special case of convex shapes, perfect reconstruction is theoretically possible as per equation (2.12).

[1] The convex hull of a volumetric object is defined as the smallest convex volume that includes the 3D shape. If the 3D shape itself is convex, then its convex hull will admit the exact same volume.

$$GenericObject \leq VisualHull(GenericObject) \leq ConvexHull(GenericObject). \quad (2.11)$$

$$ConvexObject = VisualHull(ConvexObject) = ConvexHull(ConvexObject). \quad (2.12)$$

From this analysis, we can deduce that shape-from-silhouette is imperfect in the presence of concave shapes. Figure 2.7a illustrates, using a simple 2D example, the problem that occurs when reconstructing a concave shape. Figure 2.7b and Figure 2.7c illustrate two examples of 3D concave shapes. As opposed to 2D shapes, only a subset of 3D concave regions cannot be reconstructed using shape-from-silhouette depending on the existence or not of a viewpoint that can detect the concavity. In the example of Figure 2.7c there is a detectable concavity because a view in front or behind the two cylinders would easily detect the concavity. In spite of the flaws of shape-from-silhouette in presence of concave shapes, this technique remains suitable for the reconstruction of the human body because the concavities admitted by an articulated human are generally similar to the one of Figure 2.7c.

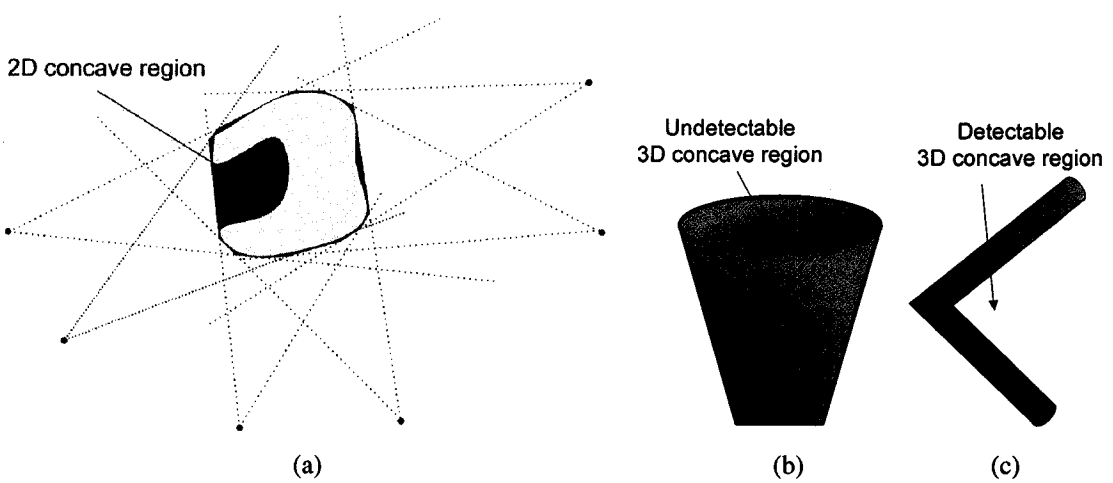


Figure 2.7. Examples of 2D and 3D concave regions

Because the use of an infinite number of viewpoints is impossible in practice, it became common, in the literature, to use the term "visual hull" to designate the outcome of a shape-from-silhouette reconstruction using a finite, but sufficient, number of viewpoints. For the case of a finite number of viewpoints, Laurentini [41] observed that

the quality of visual hull reconstruction depends on the viewpoint locations. Ideal viewpoint locations depend on the nature of the targeted object and on its orientation. Rules exist to determine the next best view for a particular object to reconstruct and thus can help minimizing the number of required camera sensors. Unfortunately, this observation does not hold for dynamically evolving objects and therefore is irrelevant for motion capture. Instead, high quality reconstruction of an evolving and deformable human body is usually achieved by introducing redundancy by means of using a larger number of viewpoints.

2.5.3 Practical Shape-From-Silhouette Implementations

From a practical point of view, the visual hull of a performer can be represented using simple equal-size voxel data. A standard procedure to compute shape-from-silhouette voxel data is presented in Algorithm 2.1 and is used in many vision-based motion capture applications [16-21]. A predefined working volume is subdivided in voxels at a desired resolution. Each voxel corresponds to a small volume within the whole working environment. The voxel map is then traversed sequentially. For each voxel, an occupancy test is performed. The goal of such test is to detect whether a voxel pertains to the foreground object or if it belongs to the background. A voxel occupancy test consists of projecting the voxel in every camera view using pre-determined camera calibration parameters. A voxel is labeled as foreground only if the voxel projects over a region that pertains to the targeted silhouette in all image views. Consequently, a voxel is labeled as background if it falls over a background region in at least one image view.

A hierarchical, multi-resolution, shape-from-silhouette is also widely used in the literature [42] and a practical multi-resolution implementation is given in Algorithm 2.2. The voxel map is instead represented using an octree. Each octree node is labeled either as background, foreground or edge. A node is classified as an edge if, in at least one view, its projection lies over an image region pertaining to both foreground and background, and, in all the other views, the voxel projects over a foreground region.

Edge nodes are typically subdivided into smaller nodes and re-evaluated unless the maximal resolution is achieved. Multi-resolution octrees require less memory and are, in general, faster to compute than single-resolution voxel maps. However, octrees are more difficult to post-process. For this reason, single resolution voxels are often preferred in motion capture applications although Caillette's implementation [22] consists of a hierarchical voxel reconstruction scheme that avoids the use of a tree data structure.

Algorithm 2.1. Single-resolution shape-from-silhouette algorithm

```
allVoxels[] = InitVoxelMap(desiredResolution);
    for all voxel in allVoxels[]
        voxel.Classify(foreground);
        for all cameras
            imageProj = CalculateVoxelProj(voxel, camera.GetCalibration());
            state = EvaluateVoxelProj(camera.GetImage(), imageProj);
            if (state == background)
                voxel.Classify(background);
                break; // no need to evaluate other camera views
            end if
        end for
    end for
```

Algorithm 2.2. Multi-resolution shape-from-silhouette algorithm

```
    allVoxels[] = InitOctree(coarseResolution);
    stack = InitStack();
    stack.Push(allVoxels[]);
    while(!stack.IsEmpty())
        voxel = stack.Pop();
        globalState = foreground;
        for all cameras
            imageProj = CalculateVoxelProj(voxel, camera.GetCalibration());
            state = EvaluateVoxelProj(camera.GetImage(), imageProj);
            if (state == edge && globalState == foreground)
                globalState = edge;
            else if (state == background)
                globalState = background;
                break;  // no need to evaluate other camera views since voxel is empty
            end if
        end for

        voxel.Classify(globalState);
        if (globalState == edge && voxel.depth() < MAX_DEPTH)
            allChilds[8] = voxel.Subdivide();
            stack.PushMultipleVoxels(allChilds[]);
        end if
    end loop
```

2.5.4 Enhancing Practical Shape-From-Silhouette Implementations

Many enhancements to the classical voxel reconstruction algorithm have been proposed in the literature. To accelerate the speed of reconstruction, the use of lookup-tables that pre-compute the numerous voxel projections is very attractive but requires a significant amount of memory. Kehl *et al.* [21] suggested maintaining a reverse lookup table that links each pixel to a collection of voxels affected by this pixel. This allows incremental updates, in time, of the voxel data. Cheung *et al.* [18] proposed a sparse voxel occupancy test (SPOT) to accelerate voxel classification. Only a fraction of the pixels pertaining to a voxel projection are evaluated and used to determine the voxel occupancy.

To increase the general accuracy of voxel models, Caillette [22] acknowledges the fact that 2D silhouettes are often imperfect. He proposed the use of an "unknown" label for silhouette pixel data and 3D voxel data classification. In a second pass, unknown voxels are reclassified either as "foreground" or "background" based on the classification of neighboring voxels. Sundaresan [23] observed that foreground pixels misclassified as background have a more negative impact than background pixels misclassified as foreground when computing the intersection of multiple silhouettes. Indeed, a voxel projection misclassified as foreground will most likely be cancelled out by another view. However, a voxel projection misclassified as background will result in the actual voxel to be labeled as background. He proposes to relax the voxel classification criteria for more robustness at the expense of accuracy. In his implementation, at least two or three voxel projections, depending on the total number of views, need to lie over a background image region in order for the voxel to be classified as background.

2.6 Chapter Summary

This chapter presented a review of state-of-the-art markerless human motion capture applications. Markerless solutions remain far from real-world applications

because of their lack of robustness in many aspects of the whole problem. Extended reviews were proposed for specific topics of interests of this thesis: multi-camera system design, multi-camera calibration and volumetric reconstruction using shape-from-silhouette.

Chapter 3. Reconfigurable Multi-Camera System Design

This chapter discusses the design of a reconfigurable multi-camera system which can be used in a variety of multi-camera applications and especially for the proposed application dedicated to human gesture monitoring in the context of piano-playing performance evaluation. Designing a multi-camera system is not a trivial task because many requirements need to be satisfied. Surprisingly, this topic is often avoided and even neglected in most motion capture systems presented in the literature. Yet, improper design can potentially lead to a sub-optimal multi-camera acquisition system. In Chapter 2, it was highlighted that proper system design is important especially for real-time systems that need to operate at a reasonable frame rate. The aim of this project is not focusing on real-time operation because robustness is privileged. However, adequate system design remains important in order to present high-quality input video data to the image processing (silhouette extraction) and 3D reconstruction modules.

This chapter attempts at resolving most of the flaws regarding multi-camera and multi-computer system design for motion capture. This discussion is subdivided into five main sections. The first section analyzes and highlights important camera hardware requirements regarding markerless motion capture. In the second section, all individual hardware components are interconnected, physically in hardware and logically in software, in order to obtain a full multi-camera system. The third section explains the frame synchronization mechanism used in our implementation. The remaining sections discuss about the development and implementation of a high-level software package designed to facilitate the interfacing of application layers with the camera hardware.

3.1 Hardware Setup

3.1.1 The Acquisition System

Our acquisition system is shown in Figure 3.1. It is composed of 3 Pentium IV 3.40 GHz computers and 8 Point Grey Research® Flea2 IEEE1394b Firewire cameras. All cameras are mounted to a reconfigurable structure. This structure allows free positioning of cameras all around the workspace. The structure itself can be enlarged to accommodate various sizes of working volume. The camera setup used to monitor the gesture of pianist musicians occupies a volume of approximately 2.5 m x 2.5 m x 2.5 m.

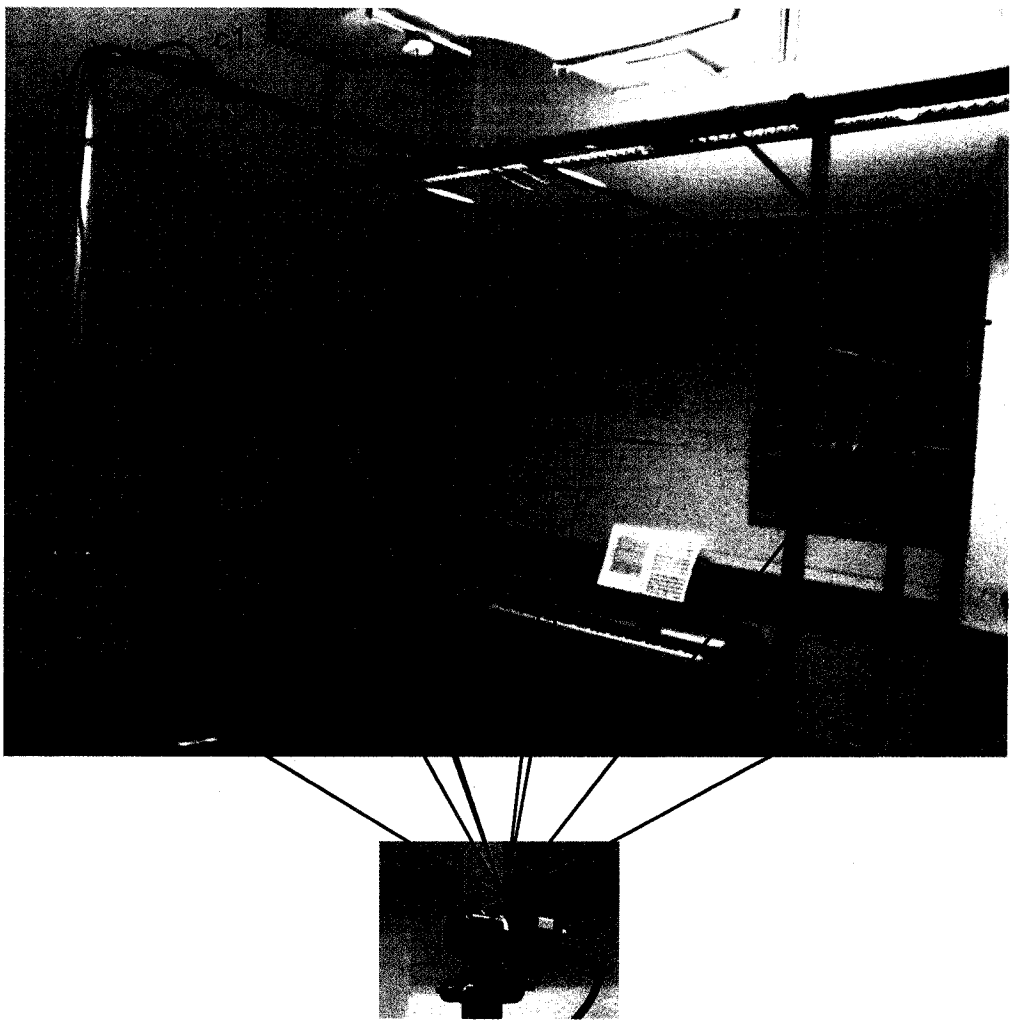


Figure 3.1. The designed multi-camera acquisition setup

3.1.2 Camera Hardware

The design of a sophisticated motion capture system with minimal constraints on the environment requires high quality hardware. The decision of using high quality cameras (PGR® Flea2), instead of cheaper webcams, is motivated by multiple factors, which are detailed in Table 3.1. To monitor human activities with high precision, the use of global shutter exposure is clearly a predominant requirement because it allows all pixels to be measured simultaneously in contrast with a rolling shutter where the pixels are measured sequentially line by line. The Flea2 cameras also allow multiple mechanisms for multi-camera frame synchronization, which is essential especially when cameras are distributed across multiple computers. These cameras can operate at a high frame rate, 60 frames per second (fps) and therefore provide high flexibility in adjusting the temporal resolution to match the speed of the motion to be captured. The frame resolution is limited to 640x480 pixels but it is sufficient for the purpose of volumetric reconstruction. Furthermore, cameras possess internal color calibration functionalities and several pre-processing functionalities to enhance the quality of the acquired video. Finally, the IEEE 1394b (FireWire b-type) bus speed allows for multiple cameras to be connected to a single acquisition node at high frame rate thus helping to reduce the global system cost and to increase the system's mobility.

Table 3.1. Factors of influence in the selection of the camera hardware for motion capture

Factors	Description
Global Shutter	Cameras equipped with a global shutter are clearly advantageous to cameras equipped with a rolling shutter especially for the purpose of accurate motion capture. Indeed, a global shutter allows all pixels in a frame to be exposed simultaneously in contrast with progressive and interlaced scan rolling shutters where the pixels are exposed line by line sequentially leading to artifacts in the presence of high motion. Interlaced rolling shutters are totally unacceptable because even and odd scan lines are evaluated in two fields separated by half of the frame duration.
High frame rate	The targeted working frame rate for our human motion capture system is estimated at 30 fps. However, the capability of extending the frame rate to 60 fps is desired for future work, especially to detect very fast and subtle (low amplitude) movements.
Resolution	A resolution of 320x240 is used in most motion capture applications because it significantly reduces the computational cost of extracting the human silhouette in each view. Furthermore, 3D voxel data is limited to coarse resolutions because incorporating a third dimension imposes very high memory and computational requirements and, thus, it becomes irrelevant to use higher image resolution.

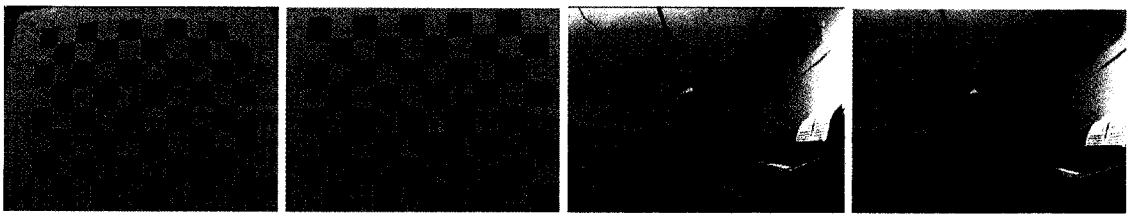
Synchronization	Frames of video need to be precisely synchronized in order to ensure consistency in time across all cameras. This is typically achieved using either an asynchronous software or hardware trigger, or using special synchronization tools provided by the camera manufacturer.
Color calibration	Cameras need to be color-calibrated with respect to the lighting characteristics of the working environment in order to improve the quality of the input video and, thus, to facilitate subsequent motion capture tasks such as silhouette extraction and volumetric reconstruction. This can be done in software [25] although, nowadays, many camera manufacturers offer advanced functionalities to automatically adjust many of those parameters (exposure, gain, white balance, etc) with minimal programming required.
Bus Speed	The use of a faster bus speed, such as IEEE 1394b (800Mb/s), allows multiple color cameras to operate on a single computer at high frame rate. The reader is referred to Appendix A for detailed bandwidth requirement calculations.

3.1.3 Camera Lenses

For this project, we use fixed focal length lenses. The use of fixed focal length lenses is advantageous because it allows the intrinsic camera parameters to be estimated only once. However, fixed focal length lenses are not flexible and thus, they need to be replaced if a wider or narrower field of view is desired. Lenses are selected individually for each camera. They are chosen to best-fit the targeted human subject in each view.

Lateral cameras (c2, c3, c6, c7) as well as one top-view camera (c4) are equipped with wide angle lenses (focal length $\cong$ 3.5 mm) because they are located very close to the position of the human subject. This type of lens allows a wide field of view even for subjects close to the cameras. Wide angle lenses are useful for workspaces that are restrained in size by physical constraints (wall, roof, etc.) which can considerably limit the maximal distance that a camera can have from the subject. However, those lenses have the disadvantage of introducing a significant amount of distortion. Figure 3.2 demonstrates the importance of distortion compensation especially when working with wide angle lenses. The amount of distortion is clearly noticeable on the checkerboard image of Figure 3.2a since lines on the pattern are not straight, especially for those far from the image center. In Figure 3.2b, radial and tangential distortions have been compensated [28] and we can notice that these lines are now perfectly straight. For the case of natural images, the difference between the distorted (Figure 3.2c) and undistorted

(Figure 3.2d) images is visually more subtle. However, from a computer vision perspective, lens distortion is critical for the quality of reconstructed 3D data.



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

Figure 3.2. Lens distortion compensation applied on a checkerboard image and a natural image: (a) and (c) are the original images, (b) and (d) are the undistorted images

The back-view cameras (c0 and c1) are located further away from the tracked subject and therefore can be equipped with 6mm lenses which provide a narrower field of view but almost no distortion is introduced. Finally, the second top-view camera (c5) also contains a 6mm lens and focuses on the pianist hands and forearms.

3.2 Architectural Design for a Multi-Camera/Multi-Computer System

The previous section individually listed the multiple components required by our multi-camera acquisition system. The purpose of this section is to analyze how these components interconnect and interact. The diagram of Figure 3.3 shows the full system architecture. One or more cameras are expected to be connected to each acquisition node (PC1-PC3). The main purpose of these nodes is to acquire and, if desired, pre-process each stream of video separately. All acquisition nodes are daisy-chained by an IEEE1394a link. This link is called the sync link and serves at synchronizing all IEEE1394a and IEEE1394b buses across all nodes as will be detailed in section 3.3. All nodes are also connected to a local area network such that they can freely communicate. An additional computer (Main PC) is shown in this diagram and serves the purpose of grouping together frames of video that occurred simultaneously and of merging the data from multiple synchronized views to achieve 3D human gesture analysis. In the context of an offline application, where the motion capture is performed only after the

completion of the video acquisition, it is not necessary to use a separate computer for this task.

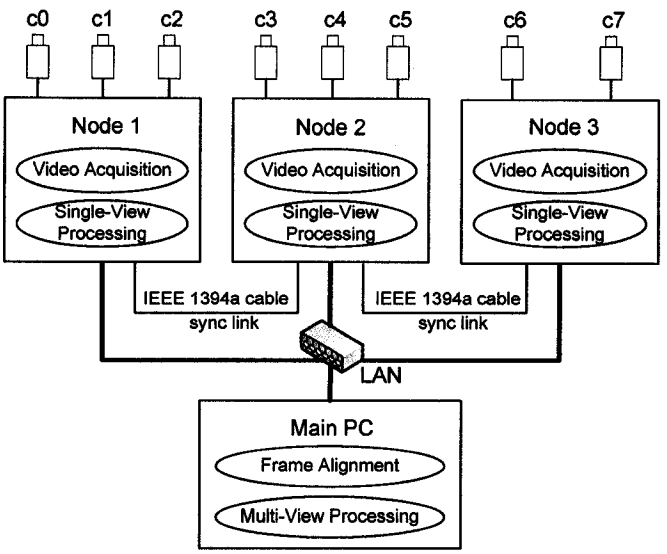


Figure 3.3. Architecture of the multi-camera/multi-computer system

3.3 Multi-Camera Frame Synchronization

In section 3.1.2, it has been highlighted that global shutter exposure and frame synchronization are two important requirements to achieve precise multi-camera human gesture analysis. Accurate 3D gesture analysis not only requires all pixels within an image to be spatially synchronized, by the use of a global shutter, but also across all views, using inter-camera synchronization. Otherwise frames of video from multiple sources may be offset in time, as demonstrated in the example of Figure 3.4, resulting in inaccurate location of various body parts in the 3D space. Thus, the purpose of frame synchronization is, by analogy, to ensure that all pixels, across all cameras, are exposed simultaneously. Our multi-camera system is synchronized in two distinct stages. The first stage is to ensure exposure synchronization across all cameras. The second stage consists of grouping frames that occurred at the same instant of time such that they can be processed together.

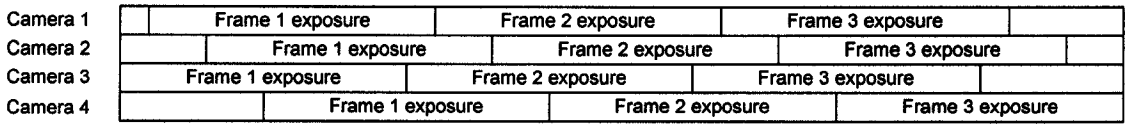


Figure 3.4. Inter-camera delays that can occur in an unsynchronized camera network

3.3.1 Inter-Camera Frame Exposure Synchronization

Multiple methods exist to ensure adequate pixel exposure synchronization across all cameras of a network. In particular, the Flea2 cameras can be synchronized using either an asynchronous software or hardware trigger. Alternatively, Point Grey Research® provides the MultiSync software which has the sole purpose of synchronizing multiple 1394a and 1394b Firewire buses such that the cameras can operate synchronously in free-run video mode.

The asynchronous trigger approach

A common approach to achieve frame synchronization is via an asynchronous trigger. This approach is supported by a wide range of camera models. In this mode, cameras will measure and output a frame of video only upon reception of a trigger signal. Such trigger signal can either be a software signal or a hardware signal. The software trigger approach is simple, low-cost and does not require any extra wiring to the cameras. This approach is accurate when all cameras are connected to a single computer but becomes inaccurate when cameras are distributed among many computers because of inter-PC communication latency. Indeed, significant synchronization inaccuracies (10-20 ms) are reported in the ViRoom system [24]. Furthermore, this approach cannot guarantee a constant frame rate because of unknown and variable overheads in both the camera driver and communication layers. To account for these drawbacks, it is possible to synchronize all cameras using a hardware trigger. With this approach, a hardware pulse is sent simultaneously to all cameras at a desired frequency (the desired frame rate). The synchronization using this approach is very accurate but requires the generation of a pulse and additional cabling.

An additional limitation related to asynchronous triggering is that the cameras are forced to operate in an asynchronous mode (snapshot) rather than in free-run video mode. In asynchronous mode, many camera models are not capable to operate at the maximum frame rate. While in free-run video mode, many camera models are optimized to overlap the shutter exposure of the upcoming frame with the data transmission of the previous frame as illustrated with the timing diagram of Figure 3.5. A penalty of up to half of the maximal frame rate may occur when a camera operates in an asynchronous mode, depending on the camera model.

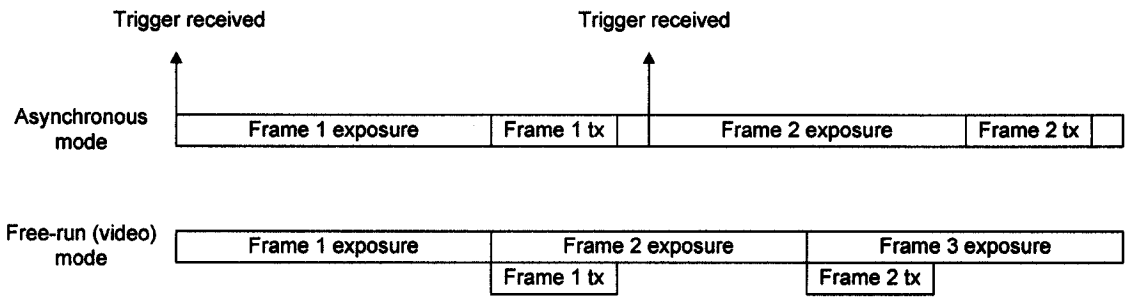


Figure 3.5. Typical timing diagram of a camera operating in asynchronous and free-run video mode

The Point Grey Research® MultiSync solution

Because there is a clear advantage in achieving synchronization in free-run video mode, Point Grey Research® provides their MultiSync application. This small application can be used to synchronize multiple IEEE1394a and IEEE1394b buses across multiple computers. The only constraint with this approach is that the MultiSync application is only compatible with products that are manufactured by Point Grey Research®. This constraint is however surpassed by many advantages which justifies the use of the MultiSync software for our multi-camera system:

1. *Complexity*. This method does not require any additional and cumbersome wiring to the cameras. Only the acquisition computers need to be linked together, with a Firewire connection.

2. *High frame rate.* Since the camera operates in free-run mode, the cameras can operate at their maximal frame rate.

3. *Synchronization precision.* The MultiSync software assigns a timestamp to each received frame of video to assess the accuracy of the synchronization. Experimentally, we observed that the timestamp value of two or more synchronized frames admits a difference of at most 1 timestamp tick. It was also observed that two consecutive frames of video differ by 267 ticks when the working frame rate is 30 fps. With this knowledge, the duration of a single tick can be calculated as follows:

$$tick\_duration = \frac{1 frame}{267 tick} x \frac{1 second}{30 frame} = \frac{0.12 ms}{tick} \quad (3.1)$$

From these computations, we estimate the precision of the frame synchronization to be ±0.24 ms (twice the tick duration) although there is no clear way to verify the exactitude of the timestamp values assigned to each frame.

To verify the frame synchronization we performed a simple test, shown in Figure 3.6, inspired from Doubek *et al.*'s work [24], where two synchronized cameras monitor a clock running on a computer screen. In this test, a very low frame rate (1.875 fps) was used because of the low timer resolution (10 ms) and because the refresh rate of the monitor influences the results. Nevertheless, when the cameras are not synchronized, the offset between two frames can be of at most 0.26 seconds (half of the frame duration). In the example of Figure 3.6, the offset is of 0.10 seconds (100 ms). With synchronization, the timer indicates the exact same time even if the cameras operate on two separate computers. It would be interesting to test the synchronization with a finer timer resolution but it would require costly equipment.

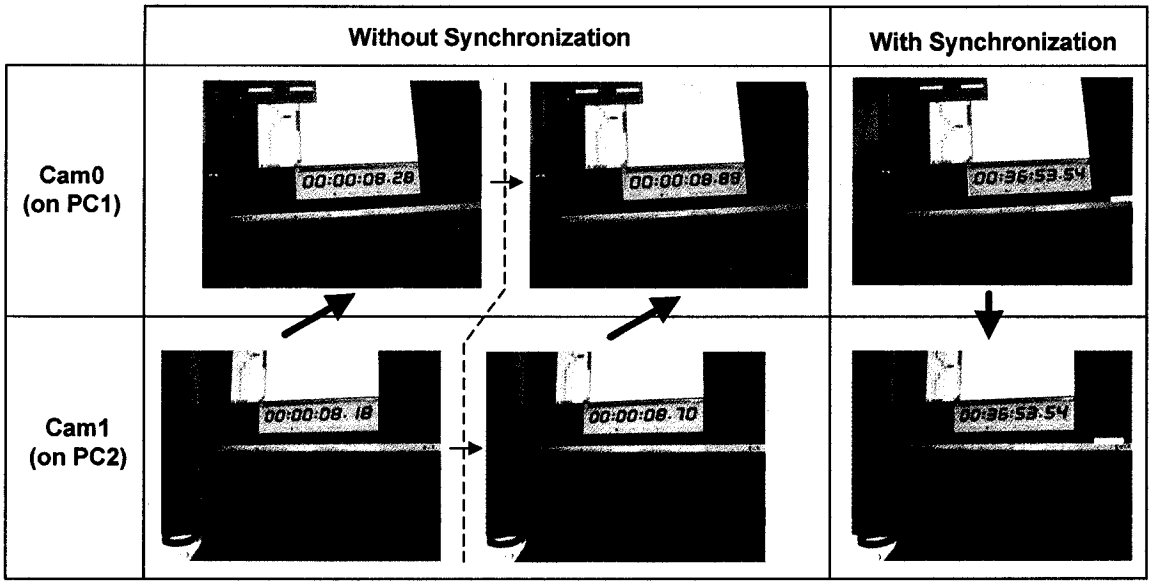


Figure 3.6. Importance of frame synchronization in multi-camera applications

3.3.2 Frame Buffering and Alignment

Having ensured that all pixels in all cameras are exposed in a synchronous manner, an additional step is required to group together synchronous frames of video. This is achieved by comparing the timestamp values embedded in every received frame of video. Our proposed strategy relies on the use of multiple queues where received frames of video can be buffered. There is one queue per camera. The alignment process evaluates and compares the timestamp values of the front element of every queue. The frame with the oldest timestamp value is dequeued and processed along with all other frames which have a timestamp value equal to the oldest frame. If a frame has a timestamp value that is more recent that the frame with the oldest timestamp value, then this frame remains in the queue and is delayed until the next operation.

An illustration of the proposed frame alignment procedure is shown in Figure 3.7. In this example, timestamp values are virtually represented using simple sequence numbering (1, 2, 3, ...). In the first iteration, the oldest timestamp is contained in Cam3 queue and is processed right-away, while the queues for Cam1 and Cam2 remain unmodified. Iterations 2 and 3 illustrate the standard scenario where the frames are

matched in all camera views. In Iteration 4, the oldest timestamp (#4) is contained in Cam1 and Cam3 queues, but Cam2 queue only contains timestamp #5. Therefore, we conclude that frame #4 for Cam2 has been lost. Iteration 5 shows another special case where the oldest timestamp (#5) is contained in both Cam1 and Cam2 queues but Cam3 queue is empty. We cannot conclude that timestamp #5 for Cam3 has been lost because it could simply not have been received yet. The process therefore needs to wait until Cam3 queue receives a new frame to continue. In iteration 6, new data has been received and timestamp #5 is matched in all cameras.

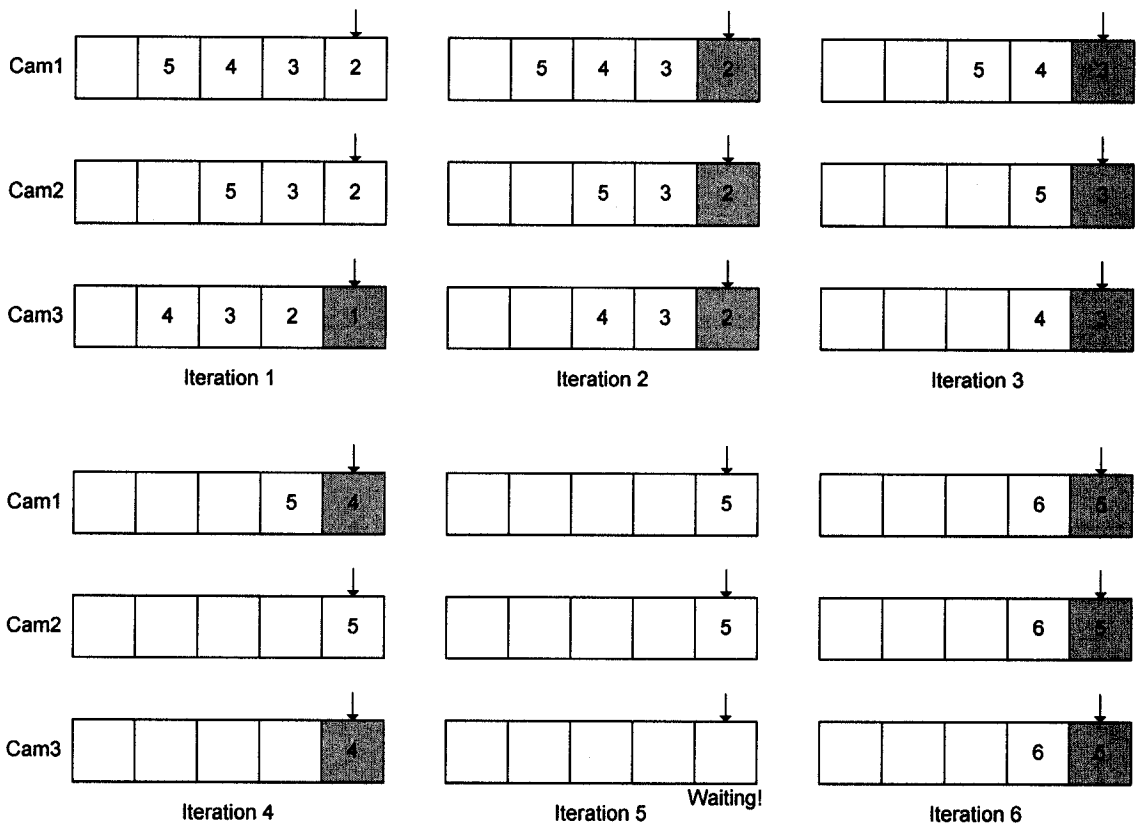


Figure 3.7. Illustration of the frame alignment process

This method assumes that the frames are enqueued in the same order that they were received. When all cameras are directly connected to a same acquisition node, frames of video are guaranteed to be received in order. Otherwise, when frames are acquired and then transmitted to a main computer, a reliable communication protocol, like TCP, would

normally be required. In a non-real-time motion capture system however, this alignment procedure can be performed offline, once the video acquisition is completed.

3.4 The Design of a Generic Multi-Camera Software Framework

The previous sections elaborated on the architectural design and the synchronization of a multi-camera/multi-computer acquisition system. Having resolved those issues, the next logical step is to encapsulate low-level interactions with cameras, into a flexible higher-level software framework. Thus, this section discusses the design and implementation of a generic multi-camera software framework.

3.4.1 The Global Scheme

Figure 3.8 provides an overview of the designed software framework that is targeted to execute on the hardware architecture presented earlier in Figure 3.3. This framework is divided in two threads and can potentially be distributed across several computers. A first thread (the acquisition thread) is responsible for acquiring raw video directly from the cameras. This thread executes on every acquisition node (PC1...PCn). Each received raw frame of video is converted, from YUV to RGB, and pre-processed (single-view processing) if necessary before being written into its corresponding queue, which is implemented using a circular buffer. There is exactly one circular buffer per camera connected and they are all maintained on the main computer (the main computer can be one of the acquisition nodes). A second thread is used to read buffered frames of video out of the circular buffers according to the frame alignment procedure discussed in section 3.3.2. Then a user supplied *multi-view processing function* is called, in real-time, against each synchronized set of video frames. Throughout the remaining of this discussion, the single and multi-view processing function will be simply referred as *callback functions*. The length of the circular buffers can be parameterized and should be large enough to account for variations in the execution time of the callback functions.

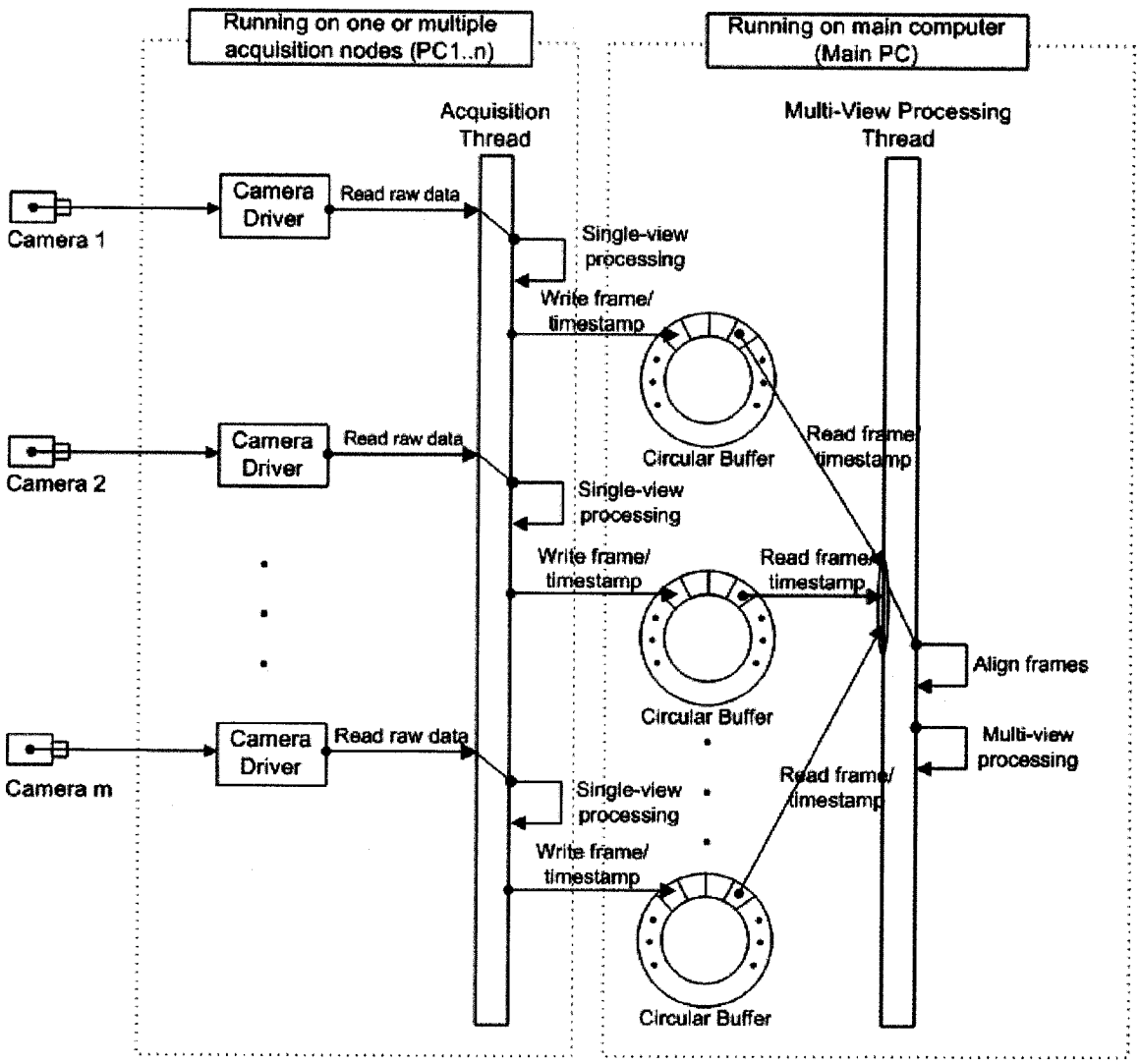


Figure 3.8. A multi-purpose software framework for real-time processing of multiple synchronized video streams

For the example of human motion capture, it would be natural to implement the silhouette extraction procedure as a single-view callback function executed locally on the acquisition nodes (PC1…PCn) because this operation is independent from other views and, therefore, parallelizable. Volumetric reconstruction and human posture estimation would be executed as a multi-view callback function running on the main computer.

3.4.2 Practical Implementation of the Software Framework

The implemented motion capture application is designed to operate offline mainly because of high computational cost involved in extracting the human silhouettes in complex scenes [9]. This eliminates the need for live inter-computer communication. To validate the concepts proposed in the previous section, a stripped-down version of this framework was implemented, for the specific case of a single acquisition node, to which multiple cameras are connected, in order to avoid inter-computer communication because it is outside the scope of this work. The only difference with respect to the software design resides in the absence of a reliable communication protocol (like TCP) to transfer the acquired images across multiple computers. This reduced implementation remains advantageous because working at such a high-level of abstraction, using simple callback functions to interact with cameras, eliminates the need to duplicate code that pertains to low-level camera operations. On a side note, this implementation provides good foundations for many real-time single-computer stereo vision applications.

In the achieved implementation, multiple cameras are attached to a single computer which collects, in its main thread, frames from all cameras which are being written to a circular queue along with their timestamp information. As per the designed framework, frames are aligned in the separate thread, before calling the user-supplied multi-view callback function. Missing frames are detected according to the mechanism introduced earlier in section 3.3.2. In normal conditions, there should never be any missing data. However, this is not true in practice. Indeed, few frames of video may be dropped during the acquisition process for several reasons:

1) Frames can be dropped because the execution time of the user-specified callback functions is too long for the frame rate at which new frames of video are received. This case needs to be handled by the programmer, therefore ensuring that real-time deadlines are respected (execution time < frame rate).
2) Frames of video can be missing because they were lost somewhere in the path linking the camera to the IEEE1394 bus and finally to the computer's internal

memory. To avoid as much as possible this scenario, bandwidth limitations everywhere along this path need to be taken into account (see Appendix A).

When a situation of missing data occurs, it is up to the programmer to decide how these extraordinary situations will be handled:

1) The callback function can simply skip this incomplete group of frames.
2) The callback function can process this incomplete group of frames by ignoring missing frames.
3) The callback function may decide to skip or process this group of frames based on the number of cameras for which data is missing.

For the purpose of volumetric reconstruction with a limited number of views (8 cameras), each viewpoint provides an important contribution to the reconstructed volume. For this reason, we decided not to partially process incomplete groups of frames (option 1). It should however be mentioned that cases of missing data are very rare (see section 3.5.2) and that the implemented framework is versatile in being able to support any of the three aforementioned alternatives.

3.5 Example of Application: Implementing a Video Recorder

The simplest use of this software framework is to record video sequences that can be post-processed afterward. Throughout this project, the FleaMultiCamViewer application (see Figure 3.9) was developed for that purpose. Frames of video, from multiple cameras, are acquired, compressed and saved to a general-purpose hard-disk in real-time on each computer separately. Table 3.2 lists a few requirements satisfied by our application. The video recorder runs independently on all 3 acquisition nodes, each one hosting at most 3 cameras. Frames that are collected among cameras attached to a same node are automatically aligned according to the implemented framework of section 3.4. At the end of the acquisition, concurrent video sequences are grouped to a single

computer which performs a final frame alignment step to ensure time-consistency in video sequences incoming from the multiple acquisition nodes.

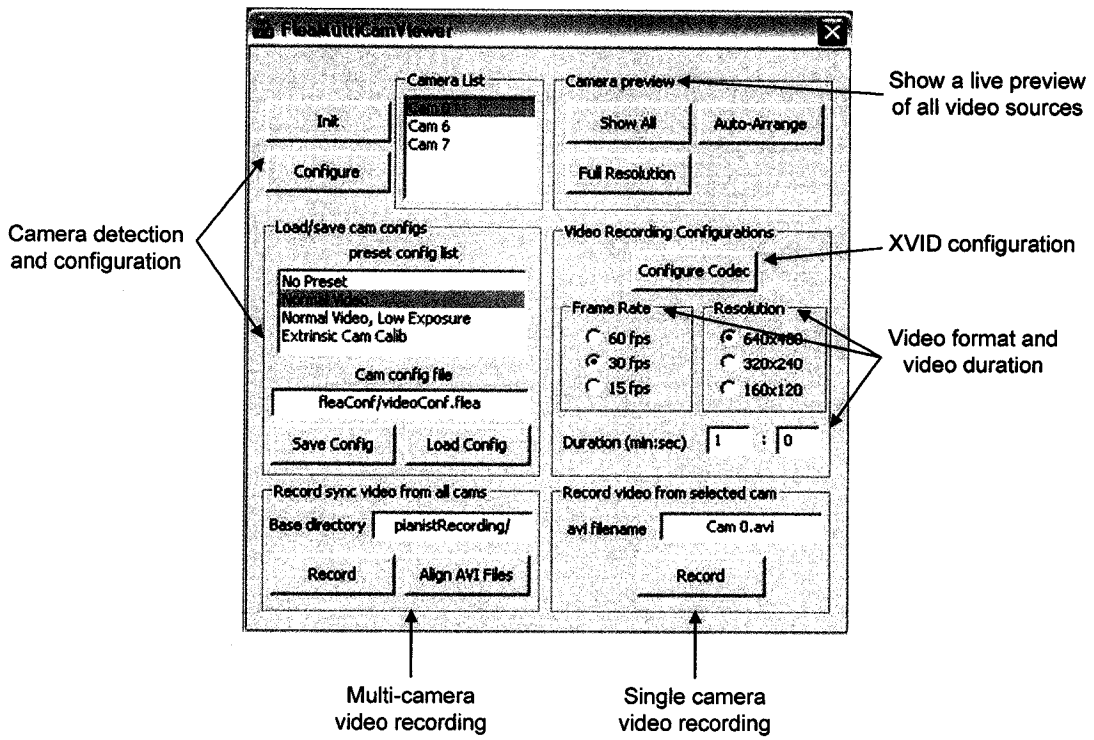


Figure 3.9. The FleaMultiCamViewer application is designed to display and record video from multiple cameras concurrently

Table 3.2. Requirements for a multi-camera video recorder

Requirements	Description
Global throughput	In order to reduce the overall system cost, the FleaMultiCamViewer must be able to record on a same FireWire adaptor from at least 3 sources of video at a targeted resolution of 640x480 and a frame rate of 30fps. This is equivalent to a global throughput of 90 fps at 640x480. Alternatively, it is possible to increase the number of cameras attached to a single acquisition node by reducing the image resolution or the frame rate.
Video quality	It is impossible to write uncompress frames of video to a hard-disk at the specified throughput (640x480@90fps). Video compression must therefore be used. Interestingly, video compression will manage to reduce the amount of data that needs to be written to disk to the expense of increasing the amount of computations in achieving the actual data compression. Another expense associated with video compression is the loss of quality. Thus, it becomes important that compressed video remains of high quality both for perceptual reasons and because high compression can compromise many image processing algorithms (including image segmentation).
Frame drops	Frame drops should be kept minimal. Sufficient buffering needs to be used to account for exceptional hard-disk access latency that can occur unexpectedly due to other background tasks performed by the operating system. Overall, it is mandatory that the average execution time of the callback function is below the operating frame duration (33 ms when cameras operate at 30 fps).

3.5.1 Implementation of a Real-Time Video Recorder

Since the multi-camera video recording functionality is built according to the framework discussed in section 3.4, a callback function needs to be developed which sole task is to compress and write each received frame to its corresponding video file (one per camera). Additionally, timestamp information for each received frame is written to a text file. Video compression is achieved using the XVID encoder [43] with real-time settings. Because real-time settings are used, only low compression can be achieved without altering the quality of the video. Experimentally, it was found that a quantization ratio of 4.0 is suitable to provide high quality video with reasonable size of data written to the hard-disk. Approximately 3 seconds (100 frames) of video buffering (size of the circular buffers) is used to account for sporadic hard-disk preemption that can occur by the operating system.

3.5.2 Examples of Video Recording Sequences

To validate the implemented video recorder, several sequences were acquired on two distinct acquisition nodes, each hosting 3 cameras. Both static and dynamic sequences were tested because it can affect the amount of data to be written to disk. Static video recording was achieved by recording live video without any active performer in the scene. Dynamic video recording was achieved by recording live video featuring a performer (i.e. a pianist).

Recording statistics for the case where all cameras operate at a resolution of 640x480 and a frame rate of 30 fps are shown in Table 3.3 and Table 3.4. Based on these results, we can observe that frames are never dropped because of a circular buffer overrun. The average execution time for each test case is always below the real-time deadline of 33 ms. Even if the peak processing time is significantly higher than the real-time deadline, the 3 seconds of video buffering is generally sufficient and could even be increased if needed. Overall, the only lost frames were dropped by the hardware layer

and thus did not reach the actual computer's memory. For 30 minutes (54000 frames) video sequences, we observed that only 1 or 2 frames were dropped for that reason. This is negligible for our purposes. In terms of video compression, a quantization factor of 4.0 is preferable because it yields to smaller file size (less data to be written to disk) and slightly faster average frame processing time especially for the recording in a dynamic scene. It should also be noted that file size on the second computer is always smaller that the file size on the first computer. This depends on the actual video content which varies with the positioning of the cameras. Recording statistics for respectively a lower resolution and a lower frame rate, on a dynamic scene, are shown in Table 3.5 and Table 3.6. At a lower resolution, it can be advantageous to use a quantization factor of 2.0 since the resulting video would be of very high quality while the file size remains acceptable.

Table 3.3. Real-time recording statistics for a static scene at high resolution

	Video format	**3 cameras @ 640x480, YUV411, 30fps**	
	Recording duration	**30 minutes (54000 frames)**	
	XVID quantization factor	**2.0**	**4.0**
PC1	Frames not written to the circular buffer	0	0
	Frame sets containing missing data	0	2
	Average frame processing time	33.04 ms	28.05 ms
	Peak frame processing time	129.05 ms	71.56 ms
	Maximum size of video file	368 061 KB	43 212 KB
PC2	Frames not written to the circular buffer	0	0
	Frame sets containing missing data	1	2
	Average frame processing time	30.28 ms	28.95 ms
	Peak frame processing time	57.46 ms	59.99 ms
	Maximum size of video file	169 243 KB	21 230 KB

Table 3.4. Real-time recording statistics for a dynamic scene at high resolution

	Video format	**3 cameras @ 640x480, YUV411, 30fps**	
	Recording duration	**30 minutes (54000 frames)**	
	XVID quantization factor	**2.0**	**4.0**
PC1	Frames not written to the circular buffer	0	0
	Frame sets containing missing data	2	1
	Average frame processing time	32.90 ms	28.60 ms
	Peak frame processing time	92.13 ms	71.61 ms
	Maximum size of video file	480 319 KB	124 685 KB
PC2	Frames not written to the circular buffer	0	0
	Frame sets containing missing data	0	2
	Average frame processing time	28.61 ms	27.92 ms
	Peak frame processing time	91.95 ms	56.84 ms
	Maximum size of video file	253 031 KB	73 074 KB

Table 3.5. Real-time recording statistics for a dynamic scene at low resolution

	Video format	3 cameras @ 320x240, YUV422, 30fps	
	Recording duration	30 minutes (54000 frames)	
	XVID quantization factor	2.0	4.0
PC1	Frames not written to the circular buffer	0	0
	Frame sets containing missing data	0	0
	Average frame processing time	6.43 ms	6.25 ms
	Peak frame processing time	51.68 ms	17.56 ms
	Maximum size of video file	81 306 KB	32 945 KB
PC2	Frames not written to the circular buffer	0	0
	Frame sets containing missing data	0	0
	Average frame processing time	6.40 ms	6.27 ms
	Peak frame processing time	13.12 ms	16.44 ms
	Maximum size of video file	46 136 KB	21 829 KB

Table 3.6. Real-time recording statistics for a dynamic scene at low framerate

	Video format	3 cameras @ 640x480, YUV411, 15fps	
	Recording duration	30 minutes (27000 frames)	
	XVID quantization factor	2.0	4.0
PC1	Frames not written to the circular buffer	0	0
	Frame sets containing missing data	0	0
	Average frame processing time	28.91 ms	25.35 ms
	Peak frame processing time	114.39 ms	53.36 ms
	Maximum size of video file	281 308 KB	73 441 KB
PC2	Frames not written to the circular buffer	0	0
	Frame sets containing missing data	0	0
	Average frame processing time	28.52 ms	26.64 ms
	Peak frame processing time	46.35 ms	48.90 ms
	Maximum size of video file	208 290 KB	45 021 KB

3.6 Chapter Summary

This chapter examined several design considerations to achieve a flexible multi-camera and multi-computer human motion capture system. Specific camera hardware requirements were first established. Then a multi-camera system architecture that ensures adequate frame synchronization across multiple views has been presented. Finally a software framework was developed to support the proposed system including the acquisition and recording of multiple synchronized streams of video.

Chapter 4. Synchronized Multi-Camera Network Calibration

A premise to almost all multi-camera computer vision applications is the accurate calibration of all camera sensors. Indeed, as introduced in Chapter 2, the reconstruction of 3D data, using shape-from-silhouette, requires the knowledge of internal camera characteristics and, most importantly, a complete knowledge about the relative positioning of all cameras. To achieve this task, two distinct classes of methods were presented: classical approaches that rely on complex calibration targets and self-calibration approaches that rely on the creation of a cloud of calibration points. Unfortunately, as per section 2.4.4, methods based uniquely on the use of complex and precisely manufactured targets yield to less accurate and highly restrictive calibration when applied to multi-camera systems. On the other hand, methods based on the creation of a cloud of virtual[1] calibration points typically yield to more accurate and less restrictive calibration, but require a more complex implementation.

This chapter proposes a systematic method [44] to calibrate generic multi-camera systems and, more particularly, the system designed in the previous chapter. The proposed method utilizes both the strength of classical 2D calibration targets to model the internal behaviour of each camera sensor, and the strength of single marker targets to generate a cloud of virtual calibration points, to achieve precise and complete inter-camera registration. The designed method aims at meeting a set of predefined requirements enumerated in Table 4.1.

[1] Throughout this discussion, we utilize the term "virtual" to emphasis on the fact that the calibration points are randomly created and do not correspond, in shape, to any physical 3D object.

Table 4.1. List of requirements for a flexible and robust multi-camera calibration method

Requirement	Description
Scalability	The method must be scalable in terms of the number of cameras in the network as well as the dimensions of the working volume.
Straightforward	The global procedure should be easy to repeat since extrinsic parameters need to be recomputed every time there is a change in the positioning of one or more camera. In addition, this procedure may need to be performed by non-expert users.
Ease-of-use	The proposed method should be fully automated and should require minimal human manipulation, especially regarding manual measurements in the working volume.
Cumbersomeness	The calibration procedure should not require any cumbersome modifications to the working environment. In particular, it should not require to move any equipment already present in the working environment.
Lighting controllability	The proposed calibration procedure is designed for indoor setups and to be performed offline, as a configuration step prior to human motion capture. It is therefore assumed that lighting can be adjusted during the calibration procedure (basically, it can be turned off).
Accuracy and precision	The outputed calibration parameters must admit sub-pixel precision in order to be competitive with other multi-camera calibration techniques [38] proposed in the literature.
Free camera positioning	No restriction should be imposed on the camera positioning apart from the requirement that cameras need to share sufficient overlap between viewpoints. This is not at all a limitation since 3D reconstruction algorithms require very high overlap among views.
Cost of the calibration equipment	The cost of the calibration equipment should remain as low as possible. The calibration equipment (targets) should not require any re-manufacturing under an increase of the working volume dimensions.

4.1 Camera Calibration Scheme

A high-level view of the proposed calibration scheme is illustrated in Figure 4.1. As for many existing multi-camera calibration procedures, the proposed method is executed in two stages. In the first stage, the intrinsic parameters and lens distortion are estimated for every camera. Since these parameters are completely independent from the positioning of cameras, it is convenient to estimate these parameters separately for each camera. In our application, a classical 2D checkerboard calibration pattern is used for this purpose. This stage is performed only at the time of the initial configuration of the acquisition setup because intrinsic parameters remain constant as long as there are no changes in the lens (or focal length), or the image resolution. Our camera setup utilizes fixed focal length lenses and the working resolution is fixed at either 320x240 or

640x480. Once the intrinsic calibration stage is completed, all cameras can be positioned in their final configuration (in the working environment). A second calibration step needs to be performed in order to estimate the relative positioning and orientation of all cameras in the camera network. This second stage is referred as the extrinsic camera calibration. This stage has to be repeated every time there is a change in the positioning of one or more camera. Practically, in the context of human gesture monitoring, this stage is repeated significantly more often because it is regularly needed to perform slight adjustments to the camera setup to best-fit the performer.

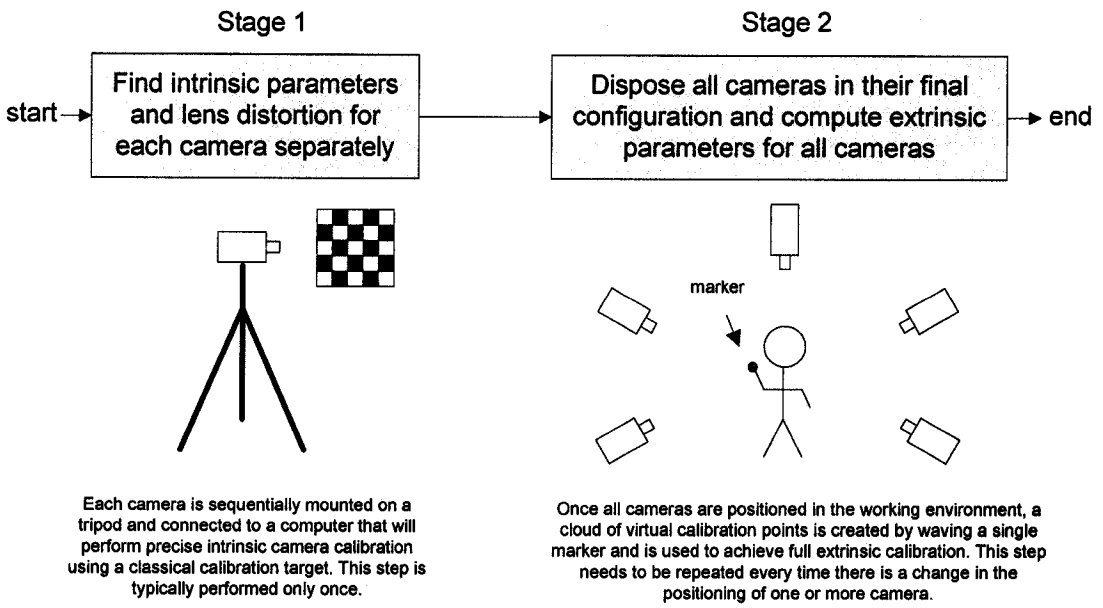


Figure 4.1. High-level view of the proposed calibration scheme

4.2 Stage 1: Intrinsic Camera Calibration

Our intrinsic camera calibration procedure utilizes an 8x10 checkerboard pattern with 2cm x 2cm cell dimensions. For each camera, multiple views of this pattern are acquired and the OpenCV [28] implementation of Zhang's method [30] is used to estimate the intrinsic camera parameters. An alternate solution would have been the use of Bouguet's Calibration Toolbox [29]. However, this toolbox is implemented in Matlab™, which makes it more difficult to integrate with our application. Furthermore,

the corner extraction process, in this toolbox, is only semi-automatic as the user is required to hand-select the four extreme corners in each view which is not convenient.

To facilitate the intrinsic calibration of all 8 of our Flea2 cameras, we developed an application (see Figure 4.2) that fully integrates with existing functionalities provided by OpenCV [28]. Multiple views of the checkerboard pattern are captured from streaming video, and corners are extracted automatically. Visual feedback is provided to assist the user such that the checkerboard can be adequately positioned within the camera's image plane. Audible feedback is also provided to the user every time a new image is processed. Our experimentation revealed that 20 views can be acquired in a timely manner and are sufficient to obtain a good and stable estimate of the intrinsic camera parameters. In addition, both radial and tangential distortions are modeled by our calibration tool to prevent destructive effect in the camera registration and 3D reconstruction, especially since many cameras in our setup are equipped with wide angle lenses.

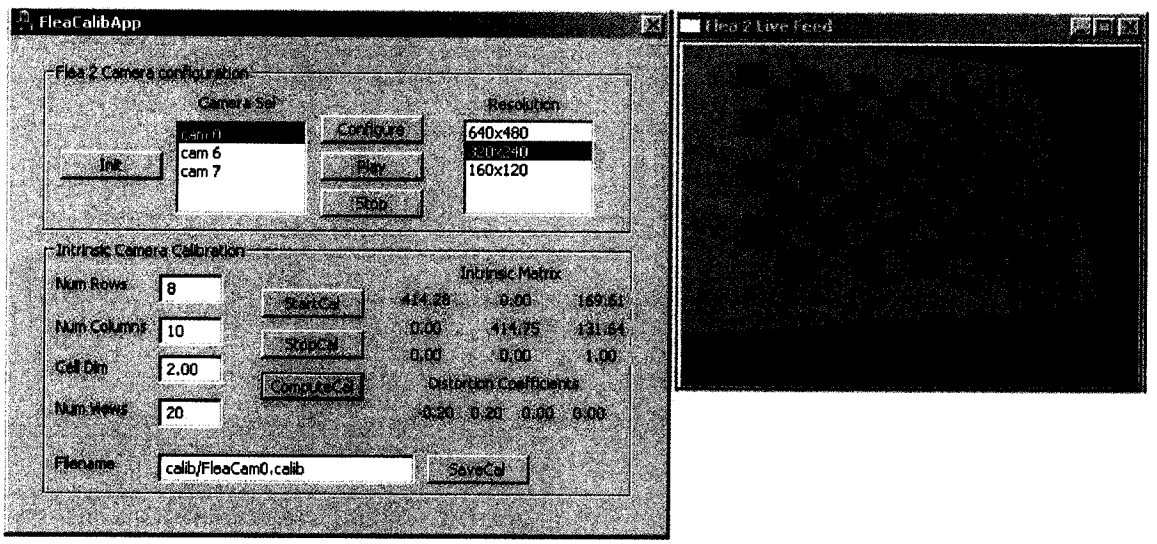


Figure 4.2. Application for fast intrinsic camera calibration

4.3 Stage 2: Extrinsic Camera calibration

Once all cameras in the network are intrinsically calibrated and positioned in their final configuration, they need to be registered to a common global coordinate system. Classical approaches that rely on the correspondences between multiple known 3D feature points and their corresponding 2D pixel position in the image plane, acquired through the use of complex 2D or 3D calibration targets, are not suitable for multi-camera calibration. As highlighted in section 2.4.4, these approaches tend to lack flexibility and, thus, it is preferable to use an approach that relies on the creation of a cloud of virtual calibration points.

In this work, a flexible framework for the extrinsic calibration of a generic multi-camera setup is proposed. The method utilizes a moving feature point to create many calibration points and to determine an initial estimate of the camera's position. This estimate is refined iteratively using a bundle adjustment technique [39] which is well known to output very accurate calibration given an initial estimate which is close to the actual solution. Therefore, the core of the proposed procedure is to determine such an initial estimate, in a robust and automated manner, to ensure adequate bundle adjustment convergence. To do so, the proposed framework counts on seven major steps, illustrated in Figure 4.3, which will be detailed in the upcoming sections.

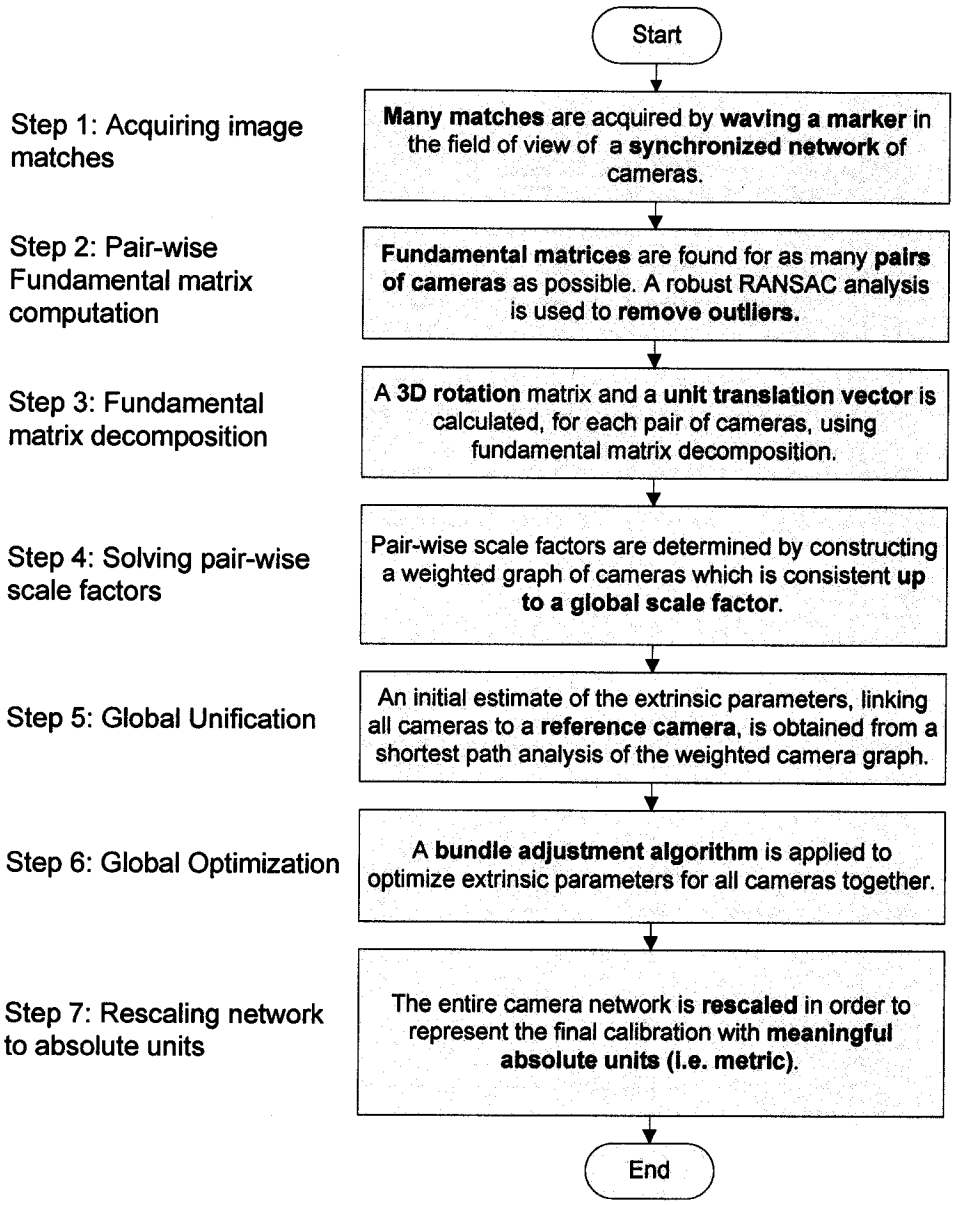


Figure 4.3. Seven-step approach to solve the problem of extrinsic multi-camera calibration

4.3.1 Step 1: Acquiring Image Matches

From a classical point of view, we recall that a calibration point is defined as a correspondence between a measured point in the 3D working volume and the pixel coordinates of that point projected on an image plane. Acquiring a sufficient number of calibration points requires either complex calibration structures or many manual

measurements which is unacceptable due to the frequency at which a multi-camera network needs to be extrinsically recalibrated.

The proposed implementation exploits the use of a virtually created cloud of calibration points (sometimes referred as virtual calibration object in the literature), which offers numerous benefits [36-38]. The strategy is to wave an easily identifiable marker over the full workspace, therefore creating such a cloud of 3D points. In our application, we use a simple light emitting device (LED) mounted at the extremity of a stick. Such a calibration instrument is easy to build (see Figure 4.4) and is very easy to identify in images when the room's lighting is turned off (see Figure 4.5). The simplicity and compactness of the calibration instrument allows excellent coverage of the working volume without any interference with the presence of other equipments in the room. While the marker is being waved, the marker's pixel position is recorded for each frame in every camera view when available.

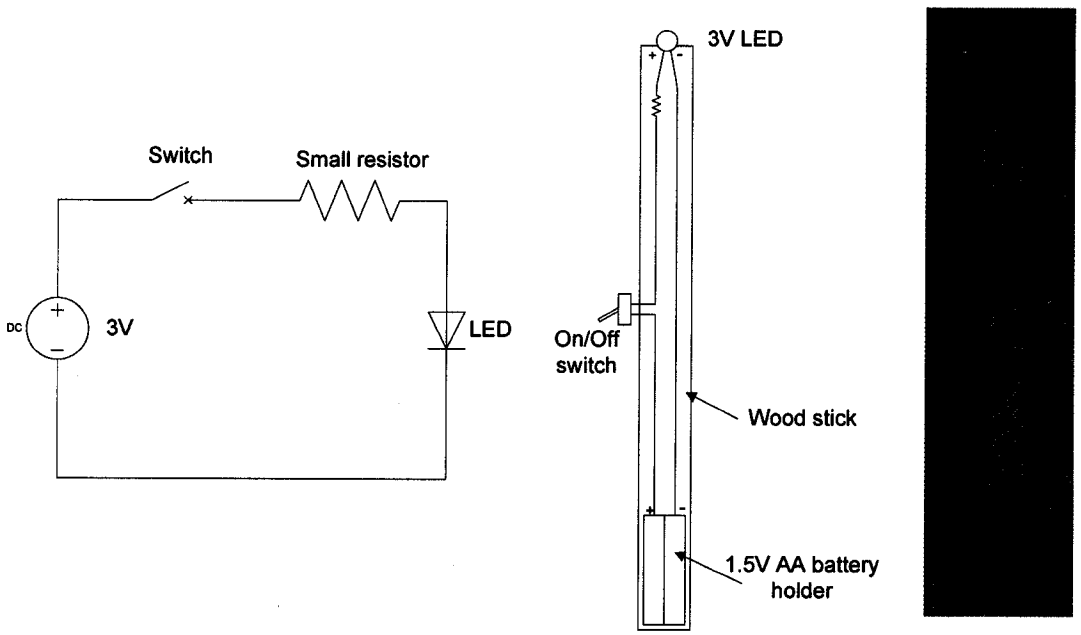


Figure 4.4. Construction of a simple calibration target for extrinsic multi-camera calibration

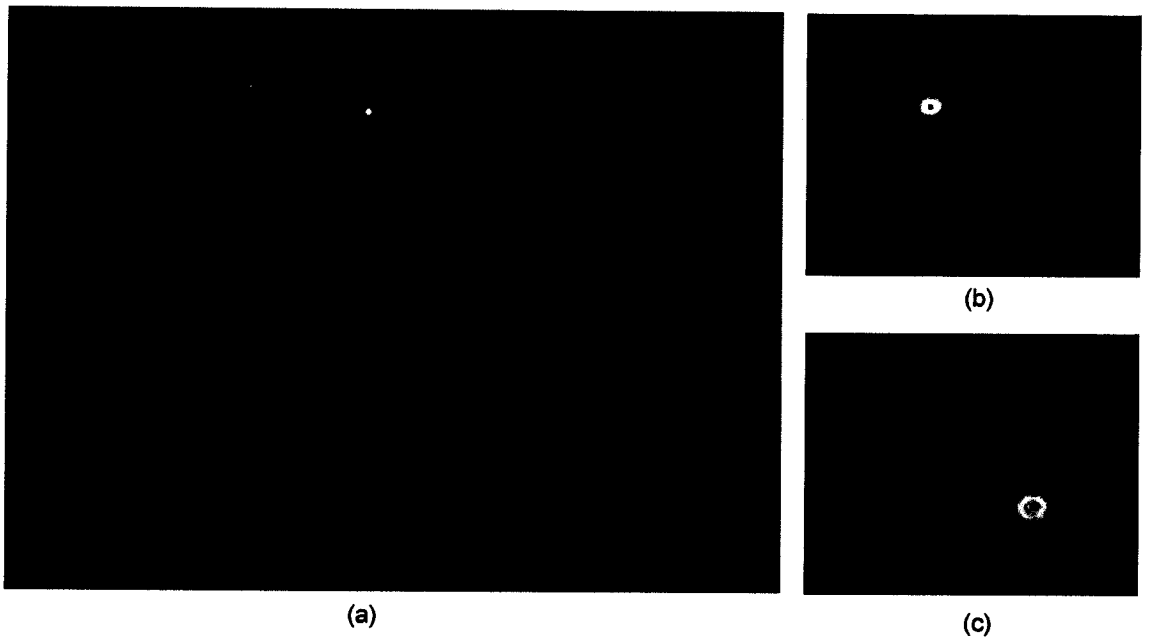


Figure 4.5. Creation of a cloud of virtual calibration points: (a) A cloud of calibration points is created by waving a red LED in the entire working environment. (b) and (c) Target identification in two images at a given instant in time

The automatic identification of the LED in images is achieved, with high confidence, using simple thresholds in RGB color space along with basic analysis on the size and shape of segmented blobs. However, the 3D world position of the target at every instant in time remains unknown and cannot be measured since the target is being waved randomly. Instead of considering absolute correspondences with 3D feature points, our procedure is based strictly on image matches. An image match is recorded whenever the target is simultaneously seen in at least 2 camera views. Synchronized multi-camera video acquisition is ensured using the infrastructure developed in Chapter 3. In order to minimize motion blur effects, cameras are configured with very short exposure time (shutter) for the tracking of the LED marker.

Following this procedure, over a thousand image matches can be obtained in less than 3 minutes of video recording. It should be noted that the nature of our marker, a red LED, will only be easily identifiable in a dark room. This can be problematic if this multi-calibration procedure is intended for outdoor use. This restriction can be avoided

by using an alternative marker such as, for example, the marker proposed in the work of Ihrke *et al.* [37].

4.3.2 Step 2: Pair-wise Fundamental Matrix Computation

Once a sufficient number of matches are found across the entire network, matches are grouped by pair of cameras such that the fundamental matrix, between each camera pair, is computed. The fundamental matrix, for a given pair, is however estimated only if at least 30 matches were found. This number was chosen arbitrarily based on the observation that, even if only 8 points are theoretically necessary, much more points are needed, in practice, for precise fundamental matrix estimation. Experimentally, we determined that using 30 matches or more is sufficient and lead to a good estimation of the stereo relation linking two cameras (in step 3).

The fundamental matrix is computed using an 8-point RANSAC implementation provided by the OpenCV library [28]. The purpose of the RANSAC analysis [45] is to eliminate outliers. An outlier is defined as a false or an imprecisely measured image match. When an outlier is found in one pair, it is removed from the global database of image matches as well (in step 1). Unlike Svoboda *et al.*'s procedure [38], our image matches are recorded in the undistorted image plane, using the lens distortion coefficients pre-computed as per section 4.2, which enables the use of a stricter outlier rejection threshold in the RANSAC analysis. Thus, we decided to classify a match as an outlier if it has a distance from point to epipolar line above 2 pixels (as opposed to 10 pixels for Svoboda *et al.* [38]). It potentially allows a better estimate of the fundamental matrix since inaccurate matches are rejected from the computation.

When computing the fundamental matrix from a set of image matches, difficulties may occur from the use of degenerative configurations [33]. A degenerative configuration occurs when two cameras share the same optical center (pure rotation, no translation) or when a set of 3D points results in a numerically unstable computation of the fundamental matrix. This latter case occurs if the 3D calibration points are all co-

planar or follows a ruled-quadratic distribution. Fortunately, those situations are easy to avoid or almost unlikely to occur [38]. Indeed, in a human motion capture system, cameras need to be separated by large baselines, discarding the problem of cameras sharing the same optical center. Furthermore, the fact that the full working volume is covered in a random manner, with the LED calibration stick, significantly reduces the risk to find all points over a co-planar or a ruled-quadratic distribution.

4.3.3 Step 3: Fundamental Matrix Decomposition

Each fundamental matrix needs to be decomposed into a rotation matrix and a unit translation vector. This is done using the method proposed by Hartley and Zisserman [33] which was earlier introduced in section 2.4.3. An essential matrix, E, is first derived from the fundamental matrix (computed in step 2) and from the intrinsic matrices of both cameras forming the pair (obtained using the one-time intrinsic calibration of section 4.2). An important property about the E matrix is that its singular value decomposition must yield to two non-zero singular values which are equals. Therefore, the numerically obtained E matrix is refined by computing the average of the first two singular values of E and setting the third singular value to zero [34]. The E matrix can then be decomposed into a rotation matrix and a unity translation vector as per equations (2.8) and (2.9).

Four mathematical solutions are found: two possible rotation matrices with an offset of 180° about the baseline and two possible translation vectors (positive or negative). To determine which solution is correct, we triangulate [46] one match and we verify which solution provides a triangulated 3D point which is in front of both cameras (positive z-axis). Multiple methods exist to triangulate, in 3D, an image match and are compared in [46]. Mathematical derivations for triangulation methods used in this work are shown in Appendix B. Ihrke *et al.* [37] reported that two out of four solutions may remain valid after this verification if the two cameras are close to a 180° rotation (two cameras facing each other). In the proposed implementation, we retain the solution which yields to the smallest triangulation error. We quantify the triangulation error as the half-

distance of the segment of intersection between the two 3D rays obtained with middle-point triangulation.

4.3.4 Step 4: Solving Pair-wise Scale Factors

In the previous step, stereo relations for each pair were found up to a pair-related scale factor. During the present phase, the magnitude of all translation vectors is estimated such that the entire camera structure becomes consistent up to a unique, global, scale factor. To do so, the method introduced by Chen *et al.* [36], which attempts to scale the links incrementally, is used and enhanced. The basis of this technique is to incrementally scale new links based on a scaled link as shown by the camera triplet of Figure 4.6.

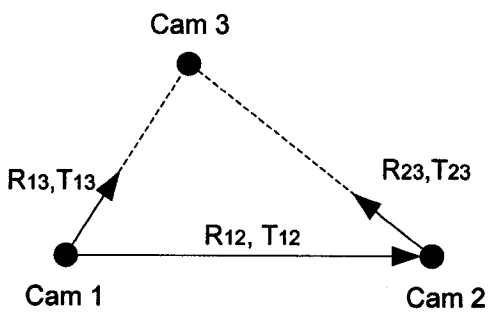


Figure 4.6. A camera triplet that shows how two links are intersected and scaled from a base link with a known scale factor

In Figure 4.6, the translation T_{12} is fully scaled but T_{13} and T_{23} are yet to be scaled. To scale T_{13} and T_{23}, the camera positions, P_{cam2} and P_{cam3}, are computed with respect to the camera 1 reference frame. P_{cam2} is obtained directly since it is equal to T_{12}. P_{cam3} is obtained by computing the vector intersection of T_{13} and T_{23}. Since T_{23} is expressed with respect to the camera 2 reference frame, it first needs to be transformed to be expressed with respect to the camera 1 frame using: $\hat{T}_{23} = R_{12}T_{23}$. With P_{cam3} known, the scaling factor of T_{13} is the Euclidian distance between P_{cam3} and P_{cam1} (0,0,0) and the scale factor of T_{23} is the Euclidian distance between P_{cam3} and P_{cam2}. Once T_{13} and T_{23} are scaled, they can be used to find scale factors of other links. This is shown in Figure 4.7.

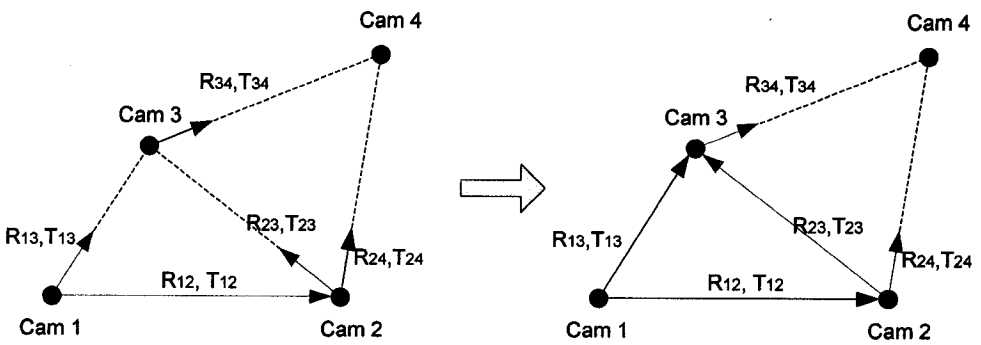


Figure 4.7. Incremental link scaling procedure:
After the scaling of T_{13} and T_{23}, T_{23} is used to scale T_{24} and T_{34}

As more links get scaled, additional links can be created by the concatenation of two or more solved links. This is shown in the example of Figure 4.8. In this example, the links l_{15} and l_{45} cannot be scaled because link l_{14} does not exist. However, l_{14} can be computed from the concatenation of l_{13} and l_{34} such that:

$$R_{14} = R_{13} \cdot R_{34}$$
$$T_{14} = R_{13} \cdot T_{34} + T_{13} \tag{4.1}$$

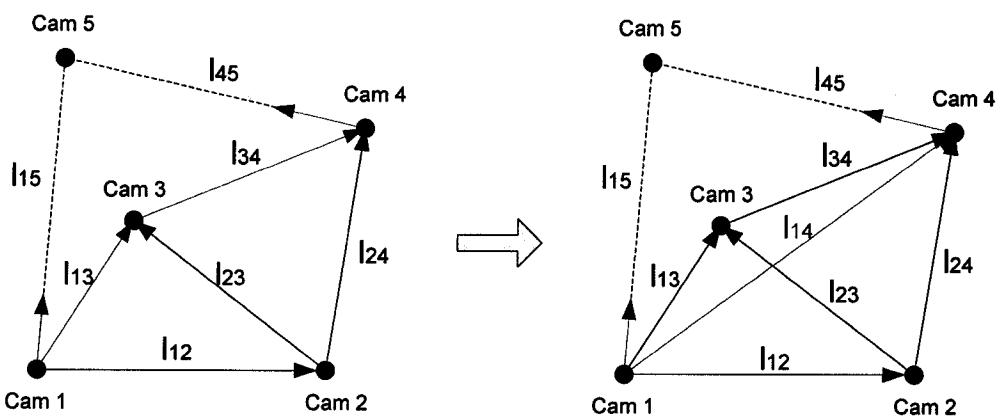


Figure 4.8. Diagrams showing a link solved by 3D links concatenation

This procedure is repeated until all links are scaled or until a path linking all cameras to the reference camera is found. Obviously, to start this algorithm, one link needs to be scaled to an arbitrary value (i.e. a scale factor of 1). Hence, the structure is fully determined up to a global scale factor. Section 4.3.7 will propose a few strategies in

order to rescale the entire camera graph to units that are meaningful (example: metric units).

In the original approach of Chen *et al.* [36], no formal mechanism is stated to designate the ordering in which links, composing the graph of cameras, should be scaled. However, pair-wise relations are not necessarily known with the same level of accuracy. When a less accurate link is used to scale another link, its error is propagated to other links thus yielding to a less accurate global estimate of the camera structure. The idea of eliminating unneeded, low quality, links is briefly suggested by Chen *et al.* but no metrics is defined to quantify the links quality. Thus, we propose a formal weighted camera graph analysis that allows the links to be scaled in a preferential order with the objective of improving the accuracy of the initial estimate obtained in step 5 (section 4.3.5) and thus ensuring adequate bundle adjustment convergence in step 6 (section 4.3.6). This particular enhancement will be discussed in details in section 4.4.

Another outstanding issue, not mentioned in Chen *et al.* [36], remains in some specific camera triplet configurations which lead to inaccurate vector intersection. Figure 4.9a shows a special case where the magnitude $\|T_{12}\|$, that is the baseline between cameras 1 and 2, is very small relatively to the distance of camera 3. In such a case, vectors l_{13} and l_{23} are almost parallel and therefore cannot be accurately intersected. Figure 4.9b shows another special case where all three cameras composing the triplet are almost co-linear. In this case as well, the vectors l_{13} and l_{23} are almost parallel. These special cases can be detected by a vector cross-product and should be avoided unless they are absolutely required by the global unification procedure. The detection and avoidance of these special scenarios are automated in the proposed framework (as will be detailed section 4.4).

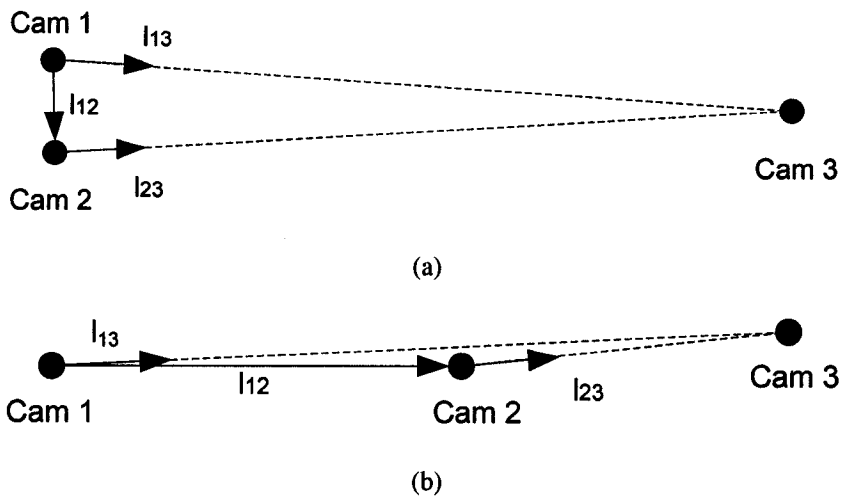


Figure 4.9. Two special cases of camera triplets that lead to inaccurate triangulation

4.3.5 Step 5: Global Unification

The purpose of the global unification step is to determine a rotation matrix and a translation vector that link all cameras in the network to the reference camera using only links that were successfully scaled in the previous step. These relations correspond to the extrinsic camera parameters. This step consists of finding a path that links all cameras to the reference camera. In a complex multi-camera setup, there will be various paths to link a camera to the reference camera and a conventional Dijkstra algorithm [47] is used to determine the shortest, thus preferential, path using the resulting weighted graph obtained from the previous step. Again, the specific details about link weights assignment are discussed in section 4.4. The global unification stage results in the computation of an initial (coarse) estimate of the extrinsic camera parameters for all cameras.

An example of global unification is shown in Figure 4.10 where camera 1 is considered as the reference camera. The extrinsic parameters for camera 1 correspond to an identity rotation matrix and no translation. The extrinsic parameters for cameras 2 and 3 are determined directly from the inversion of links l_{12} and l_{13} respectively. These links need to be inverted recalling, from Figure 2.2, that the extrinsic camera parameters

encode the relations linking all cameras to a global reference frame (here camera 1). To obtain the extrinsic parameters for camera 4, a link between camera 1 and 4 is first created from the concatenation of links l_{12} and l_{24}. This newly created link is then inverted. A similar approach is used to calculate the extrinsic parameters for camera 5. A link is inverted using:

$$\hat{R} = R^T \mid \hat{T} = -R^T T \tag{4.2}$$

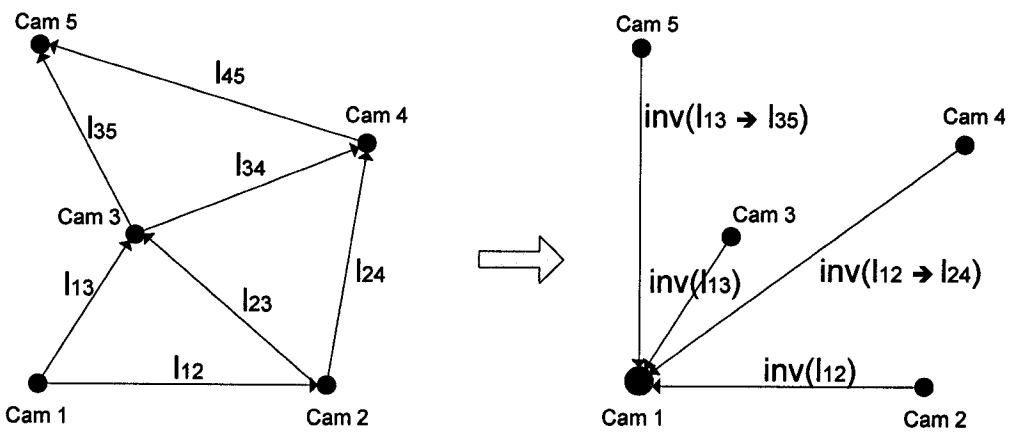


Figure 4.10. Example of global unification

4.3.6 Step 6: Bundle Adjustment Optimization

Upon successful global unification, a coarse estimate of the extrinsic parameters for all cameras is obtained. To reach a more precise calibration, these extrinsic parameters are optimized using a sparse bundle adjustment implementation which integrates with the framework proposed by Lourakis *et al.* [48, 49]. The bundle adjustment process is illustrated in Figure 4.11. This procedure takes as inputs the intrinsic and the estimate of the extrinsic parameters for each camera to calibrate. A second input to the bundle adjustment consists of a database of measured 2D image points from each view. This database is in fact the collection of image matches from step 1 (section 4.3.1) minus the outliers. Similarly, a database of 3D world points is provided. This second database contains the 3D world position of all calibration points acquired in step 1. Since the position of these points is completely unknown, as established in step 1, these 3D points

need to be calculated. The 3D location of every calibration point is calculated from the triangulation of multiple corresponding image points using the intrinsic and the coarse estimate of the extrinsic camera parameters. A standard linear least-square triangulation algorithm [33, 46] was implemented for this purpose. The reader is referred to Appendix B for further details about this implementation.

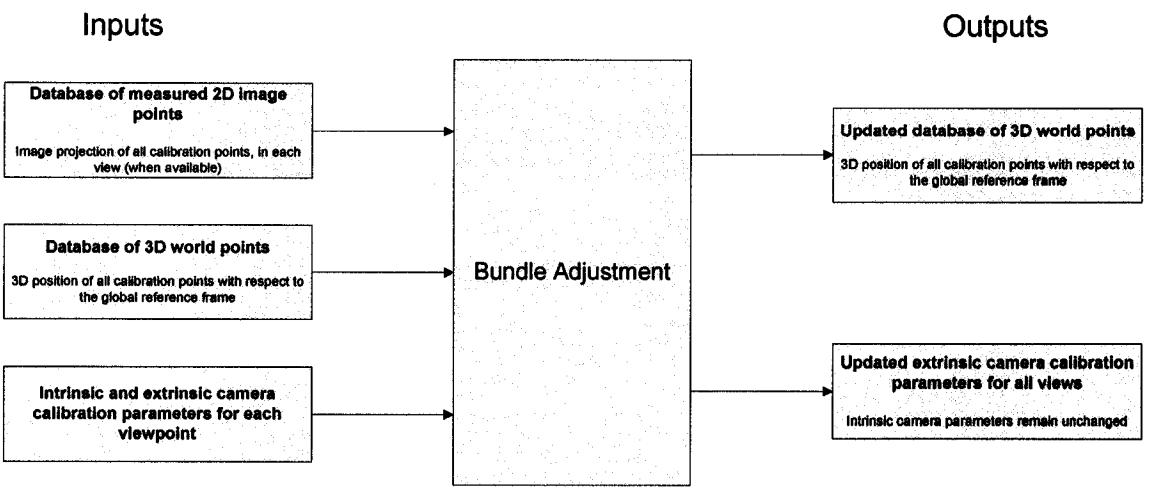


Figure 4.11. High level view of the Bundle Adjustment optimization procedure

The bundle adjustment can serve multiple purposes. A bundle adjustment can be used to iteratively refine the extrinsic camera parameters, or the database of reconstructed 3D points, or both. In situations where the 3D world points are known exactly (ground truth), for example because they were manually measured, then only the extrinsic parameters should be refined. On the other hand, if the camera calibration is considered to be exact, then only the database of 3D world points should be modified by the bundle adjustment. In our case both the extrinsic camera parameters and the database of 3D points need to be optimized by the bundle adjustment since none of them constitutes ground truth data. The extrinsic camera parameters need to be refined because it is the overall goal of the procedure. The database of 3D world points also needs to be refined because those points where calculated off of a coarse, inexact, estimate of the camera calibration. At the end of the bundle adjustment, only the updated extrinsic parameters are retained and the updated database of 3D world points is discarded.

4.3.7 Step 7: Rescaling Network to Absolute Units

In step 4, pair-wise scale factors were calculated in order to obtain a consistent camera network. However, to begin this procedure, one link (the first link), had to be scaled to an arbitrary factor of 1.0. As a result, the extrinsic parameters found in step 5 and refined in step 6 are only defined up to a global and unknown scale factor. In terms of calibration accuracy, there are absolutely no benefits in determining this absolute scale factor. Indeed, if the extrinsic translation vector for every camera is multiplied by an equal scalar value, there is absolutely no gain nor loss in the precision of the calibration solution. However, one advantage of finding this absolute scale factor is that the calibration data describing the camera structure and, consequently, the working volume would be expressed using meaningful units, such as metric units. Thus, from a human operator perspective, finding that absolute scale factor would become advantageous. Two main strategies are proposed to determine the absolute scale factor.

1) An approximate solution consists of manually measuring the baseline distance between the reference camera and any other camera in the network. The absolute scale factor becomes the ratio between the measured baseline and the baseline extracted from the camera model. This approach is easy to implement but is not accurate since the position of the camera's optical center is not precisely known and manipulations are necessary.

2) Another approach builds upon the identification of two features in the working volume seen by two or more cameras and with a known (measured) absolute distance. The absolute scale factor becomes the ratio between the absolute (metric) distance and the distance calculated from triangulation of the two features. This approach requires the capability of identifying and reliably matching the features in multiple camera views.

In our implementation, the problem of determining the absolute scale factor is achieved by substituting the single point calibration target, presented earlier in Figure

4.4, with a dual point calibration target, as shown in Figure 4.12. Markers are robustly distinguished using two different colors (red and blue). The distance between the two markers is reconfigurable (from approximately 10 cm to 70 cm) in order to accommodate working volumes of various dimensions. This enhanced, dual-point target, achieves both the extrinsic calibration and the estimation of the absolute scale factor in a single manipulation. This scale factor is estimated, frame-by-frame, and averaged at the end.

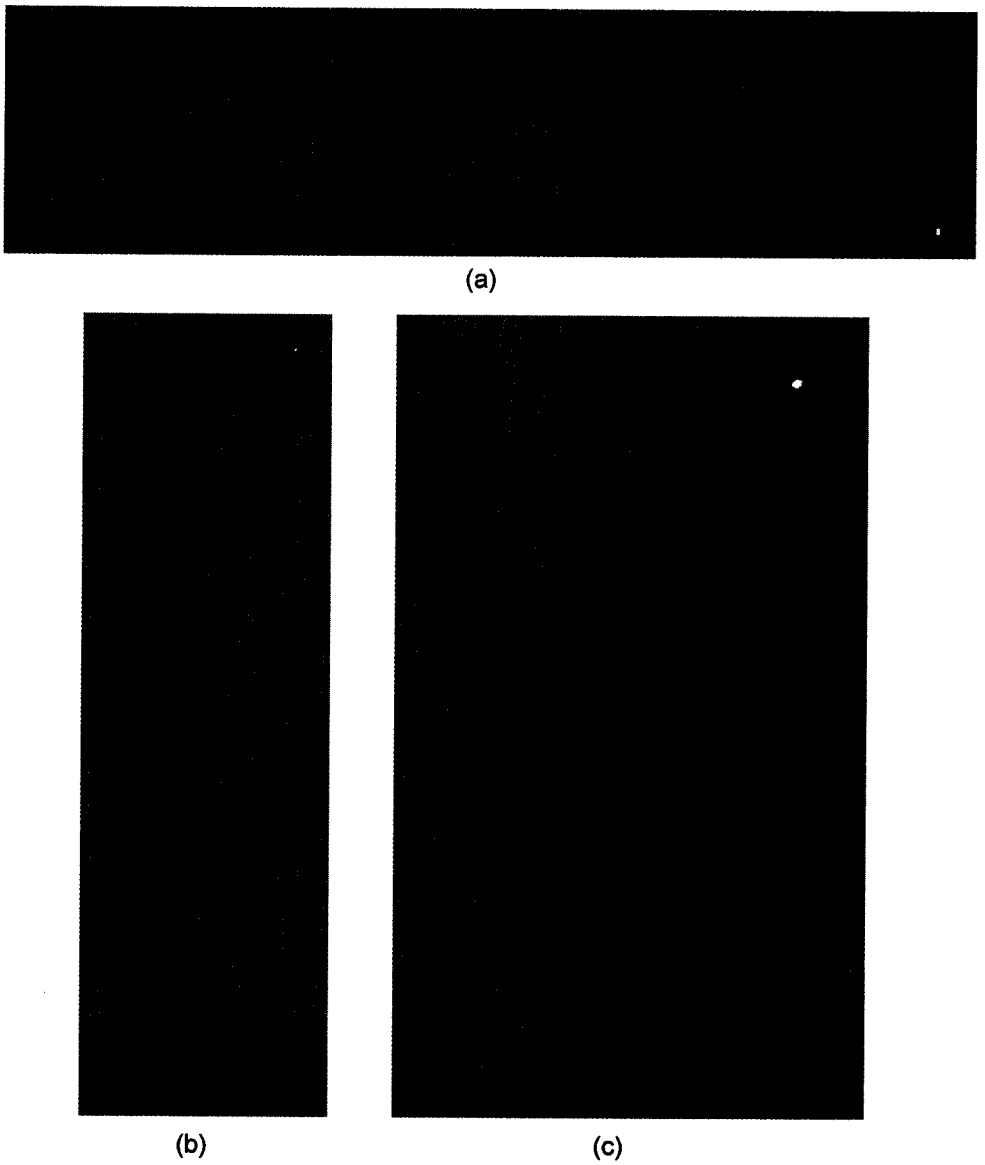


Figure 4.12. Reconfigurable dual point calibration target which achieves extrinsic multi-camera calibration with no scale factor ambiguity; distance between markers: (a) 76 cm, (b) 46 cm, (c) 13 cm

4.4 Improving the Initial Estimate of the Extrinsic Parameters

In order to ensure adequate bundle adjustment convergence to a valid solution, and not to a sub-optimal solution (a local minimum), a reliable initial estimate of the extrinsic camera parameters is required. To improve the reliability of that initial estimate, it is proposed that links are scaled in a preferential order such that links with a smaller error are always preferred over links with higher error in both solving un-scaled links (in step 4) and determining the resulting extrinsic parameters of each camera (in step 5). As an enhancement to the method proposed by Chen *et al.* [36], we propose a formal methodology to assess the quality of the links and, consequently, to achieve a preferential order. Assessing the quality of a link is not a trivial task and it depends on many factors:

1) The number and the location of the matches used in the fundamental matrix computation for a specific link can sometimes impact the quality of the link. This is why the fundamental matrix for a given pair of cameras is only computed if a sufficient number of matches (at least 30 matches) were found in that pair.

2) The pair-wise average and maximal re-projection error can partially assess the quality of a link. These two values are determined by the triangulation and re-projection of all image matches for a camera pair.

3) The problem of error accumulation is probably the factor that influences the most the quality of a scaled link. Indeed, the number of intermediate scaling operations for a specific link can serve as an indicator of the quality of this link. A similar observation can be made for links found by the concatenation of multiple intermediate links.

4) Special degenerative configurations of a camera triplet critically impact the quality of a scaled link and need to be avoided whenever possible.

Thus, to improve the overall accuracy of the initial extrinsic calibration estimate, we proposed to create a weighted camera graph that enforces a special ordering in solving the camera graph. Such preferential ordering is controlled by a set of heuristic rules:

Rule 1) A startup link is assigned a default scale factor of 1.0 to kick start the link solving procedure of section 4.3.4. This startup link is assigned a weight of 1 and should be any link that connects to the reference camera. Doing so will allow the graph to be optimized such that lower weighted links will be located near the reference camera.

Rule 2) When links are scaled by vector intersection, they take the weight of the base link + 1, where the base link is the link between the first two cameras of a triplet. This weighting indicates directly the number of intermediate scaling operations that separate a link from the startup link.

Rule 3) Links found by the concatenation of multiple scaled links will be weighted as the sum of weights of all intermediate links.

Rule 4) A queue is used to regulate the order in which the links are evaluated. At the beginning of each iteration, all links that have already been scaled, including links that can be solved by concatenation of existing links, are added to an ordered queue where lower weights get dequeued first. When a link is dequeued, an attempt is made to scale as many pairs as possible using this sole link by testing all possible cameras as the third camera of a triplet. Then a next link is dequeued until the queue is empty. A new iteration begins until no new link can or needs to be scaled.

Rule 5) Unscaled links are added incrementally in the camera graph, such that poor quality links are not part of the network if they are not required by the global unification. The quality of an unscaled link is computed as a weighted average between the mean (weight=70%) and the maximal (weight=30%) reprojection error[1] of all image matches for this pair. This quality factor gets computed after the fundamental matrix decomposition step (step 3). The steps of solving pair-wise scale factors and global unification are performed iteratively starting only with high quality links and introducing lower quality links later in the process, and only if they are absolutely needed. This rule may seem to contradict with previous rules in the sense that the link scaling should be done in a breath-first manner. Thus, removing too many links from the graph may result in requiring additional levels of intermediate scaling. Therefore, only the links which admit an abnormally high error, in comparison to other links, need to be withdrawn.

Rule 6) When a degenerative configuration is detected for a camera triplet, as illustrated earlier in Figure 4.9, the two links to be intersected are placed in a temporary buffer. Once the scaling of all links is completed, this temporary buffer is revisited. If the delayed links have not been scaled using an alternate route, then we have no other choice but to intersect the two vectors even if they are almost parallel. In such case, a special weight of twice the number of cameras in the network is assigned to those links because they are expected to be very imprecise.

Figure 4.13 demonstrates an example of the aforementioned enhanced weighted graph analysis. In Figure 4.13a, the link l_{01} is chosen as the startup link and is assigned a scale factor of 1.0 and a weight of 1. In Figure 4.13b, links (l_{02}, l_{12}), (l_{03}, l_{13}) and (l_{04}, l_{14})

[1] The reprojection error for an image match is computed from the triangulation of this match in 3D and the reprojection of this 3D point back into both image planes. The reader is referred to Appendix C for a mathematical derivation.

are all scaled using the base link l_{01} and therefore take a weight of 2. In Figure 4.13c, the base link l_{04}, with weight 2, is used to scale links (l_{07}, l_{47}). Similarly, link l_{24} is scaled using the base link l_{12} and the already known link l_{14}. All new links are assigned a weight of 3. It should be noted that links (l_{15}, l_{45}) remain unscaled even if the base link l_{14} is known because cameras c1, c4 and c5 are nearly co-linear. In Figure 4.13d, the base link l_{47} is used to scale links (l_{45}, l_{57}) with a weight of 4 and link l_{57} itself is used to scale links (l_{56}, l_{67}) with a weight of 5. A new link is created between cameras c1 and c7 from the concatenation of links inv(l_{01}) and l_{07}. The weight of this new link is the sum of the intermediate links (1+3 = 4) and serves as a base link to scale the remaining link l_{15} with a weight of 5.

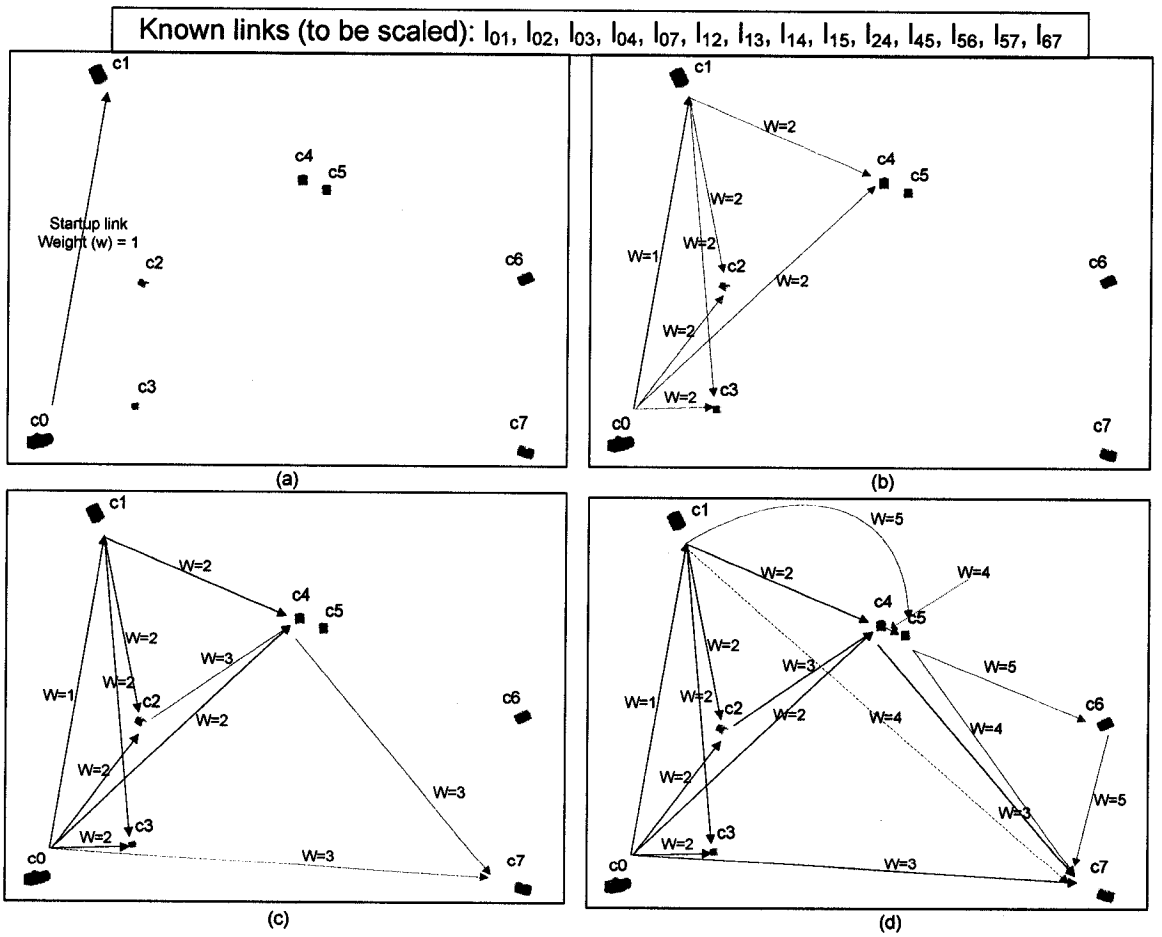


Figure 4.13. Example of the enhanced link scaling procedure

In the global unification step, a search for the shortest path to camera c0 is performed to find the extrinsic parameters of all other cameras. It results in:

Camera c1: $inv(l_{01})$

Camera c2: $inv(l_{02})$

Camera c3: $inv(l_{03})$

Camera c4: $inv(l_{04})$

Camera c5: $inv(l_{01} \rightarrow l_{15})$

Camera c6: $inv(l_{07} \rightarrow inv(l_{67}))$

Camera c7: $inv(l_{07})$

4.5 Results and Analysis

The purpose of this final section is to validate the enhanced multi-camera calibration procedure elaborated throughout this entire chapter. Our validation methodology was performed as followed:

- Section 4.5.1 analyses the overall (final) calibration accuracy obtained by applying the proposed method to a variety of multi-camera networks.
- Section 4.5.2 performs a qualitative analysis of the proposed calibration method which relates to the pre-defined requirements established earlier in Table 4.1.
- In section 4.5.3 a comparison with respect to similar multi-camera calibration methods is achieved in support to the proposed method.
- Sections 4.5.4 and 4.5.5 concentrate on analyzing the behaviour of the proposed method in the presence of critical scenarios.
- Finally, section 4.5.6 provides numerical results of absolute scale factor estimation using the dual marker calibration target.

4.5.1 Assessing the Calibration Accuracy

The average reprojection error (see Appendix C) is used as a basis to evaluate the accuracy of the achieved calibration. Any point seen by two or more cameras can be triangulated in the 3D space and reprojected back into the image plane of each camera using the final calibration. The Euclidian distance between each reprojected point and their respective original (measured) position is computed and averaged. In order to validate adequately the calibration accuracy, it is important to use a new set of virtual 3D points, rather than the set used during the calibration procedure, to ensure that the camera parameters were not overfitted with respect to a particular dataset [36].

To assess the versatility and stability of the proposed method, many camera networks were tested. In total, 14 different camera networks were tested. Six of these camera networks were composed of 3 cameras (see Figure 4.14) and were used to test the vector intersection accuracy for many configurations of camera triplets. In particular, test case 5 shows a camera triplet where two cameras have a very short baseline with respect to the reference camera (in red). Test case 6 shows a critical configuration where cameras are close to being co-linear. In addition, 4 networks composed of 4 cameras were also tested (see Figure 4.15). Test case 9 contains 3 co-linear cameras but we recall that the proposed algorithm should utilize the remaining camera to avoid intersecting parallel vectors. Test case 10 shows a special case where all cameras are co-planar and facing each other. Finally networks composed of 5 to 8 cameras were tested (see Figure 4.16) to ensure that the proposed calibration procedure remains accurate when the number of camera increases.

The calibration accuracy is expressed in pixels. To be fair in comparing our achieved calibration accuracy with results reported by other techniques, the tests were performed at a standard resolution of 640x480 because it is the most commonly used

resolution in the literature[1]. Furthermore, 5 out of the 8 cameras used for the testing were equipped with highly distorting wide angle lenses.

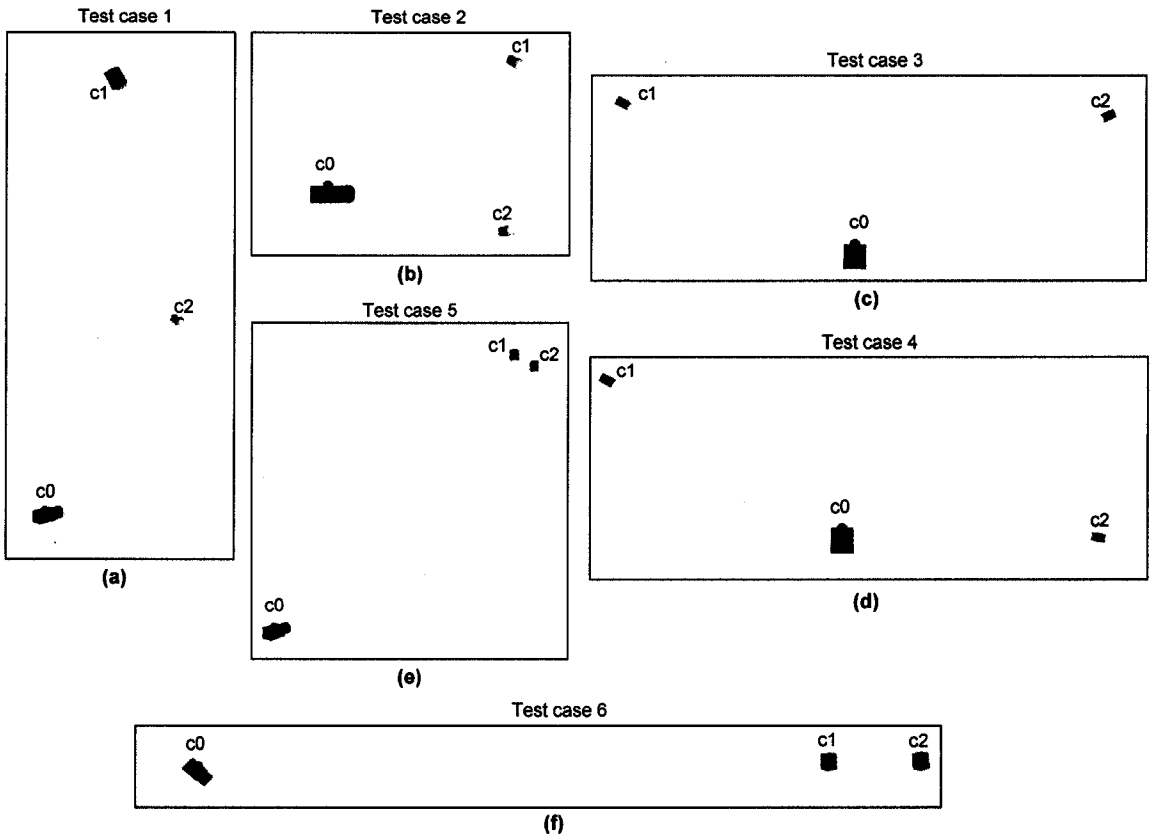


Figure 4.14. Test cases for networks of 3 cameras

[1] It should be noted that our actual motion capture experiments were performed at a reduced image resolution of 320x240. During real experiments, the multi-camera network was simply re-calibrated for this lower resolution.

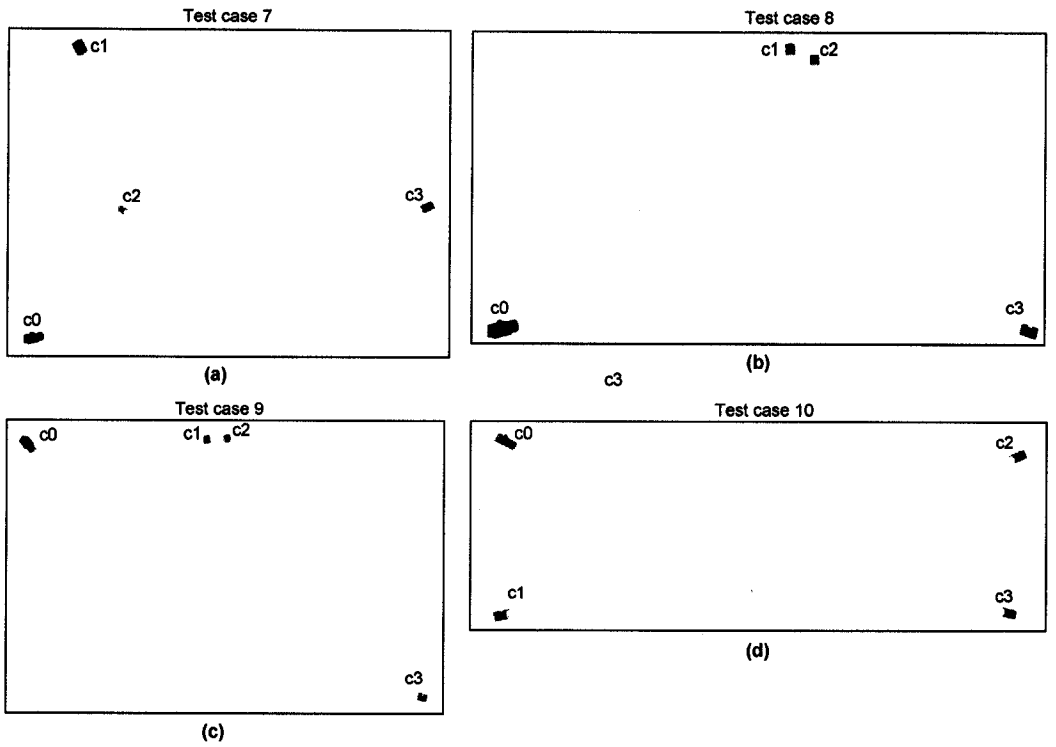


Figure 4.15. Test cases for networks of 4 cameras

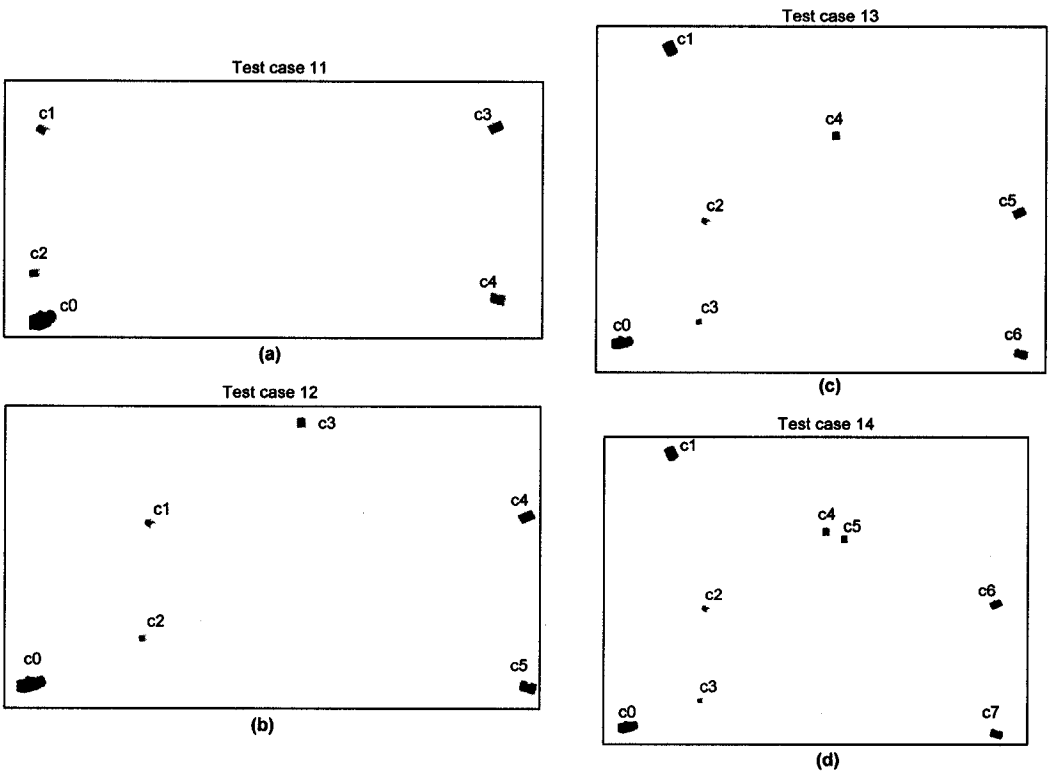


Figure 4.16. Test cases for networks of 5 to 8 cameras

Table 4.2 shows the results for all enumerated test cases. For every test case, over a thousand image matches were identified in only 3 minutes of video recording featuring our LED-based target. From these results, we notice that all camera triplets tested (test cases 1 to 6) are registered with very high precision. Even in the test case 6, where cameras are quasi co-linear, the network is successfully calibrated. This particular scenario is analyzed in more details in section 4.5.4. Overall, for networks of 3 to 8 cameras, the average reprojection error before the bundle adjustment step is always small enough to guaranty bundle adjustment convergence but remains too high for accurate 3D reconstruction. After the bundle adjustment step, this error is reduced to less than ½ pixel and increases very slowly with the introduction of additional cameras. It has never been observed that our bundle adjustment would not converge for any of the enumerated test cases. To verify that the proposed method does not suffer from data overfitting, all calibrations were validated using a new set of virtual 3D points (shown in the last column). No degradation in the calibration accuracy could be noticed with the new set of points.

Table 4.2. Calibration statistics for networks of 3 to 8 cameras

Test case	Number of cameras	Number of 3D points	Average reprojection error [pixels]		
			before bundle adjustment	after bundle adjustment	with new set of 3D points
1	3	1452	1.4125	0.2949	0.2680
2	3	1267	0.6637	0.2637	0.2438
3	3	1609	1.4125	0.2742	0.2511
4	3	1441	3.3140	0.2848	0.2868
5	3	1088	0.8138	0.2424	0.2314
6	3	1238	1.7347	0.2370	0.2150
7	4	1798	1.6186	0.3518	0.3197
8	4	1436	4.6608	0.3294	0.3302
9	4	1672	2.5794	0.2787	0.3032
10	4	1847	0.8910	0.2894	0.3023
11	5	1869	3.4704	0.3476	0.3429
12	6	1906	3.7134	0.3895	0.3934
13	7	1911	3.9670	0.4152	0.4090
14	8	1885	3.9365	0.4174	0.4126

Table 4.3 provides detailed statistics, per camera, about test case 14 which utilized all 8 cameras. From this table, it is possible to notice that before the bundle adjustment,

the error is relatively high for the purpose of 3D motion capture, but, overall, remains near the desired local minimum thus ensuring adequate bundle adjustment convergence to very accurate final calibration for all cameras. Prior to the bundle adjustment, the error fluctuates because of error propagation in links located far from the reference cameras, but remains low in comparison with the results reported in the original Chen *et al.*'s implementation [36], as further discussed in section 4.5.3.

Table 4.3. Detailed calibration statistics for test case 14 – 8 cameras

Camera id	Num Proj.	Reprojection error before bundle adjustment [pixels]			Reprojection error after bundle adjustment [pixels]		
		average	std. dev	max	average	std. dev.	max
c0	961	1.6383	0.7935	3.6186	0.4255	0.2674	2.1811
c1	1373	1.8790	0.7423	4.9278	0.3047	0.1978	1.3248
c2	1398	2.0240	1.9631	17.0612	0.4893	0.2799	2.0468
c3	1077	5.2249	1.5802	9.7173	0.4810	0.2811	1.5533
c4	1390	5.7618	2.7352	22.7446	0.4699	0.2906	3.1634
c5	520	4.0560	2.3513	17.0315	0.4124	0.2190	1.3888
c6	1563	3.0211	1.4204	8.1933	0.3720	0.2325	1.7669
c7	1296	7.9117	2.1979	13.9244	0.4007	0.2312	1.6123
All Cameras	9578	3.9367	1.8315	22.7446	0.4174	0.2430	3.1634

4.5.2 Qualitative Evaluation

The proposed calibration method meets the predefined requirements of Table 4.1. This method is easy-to-use because it requires very few human manipulations. The only mandatory manipulation is to wave a LED calibration stick randomly over the working volume. Absolutely no cumbersome or time-consuming absolute measurements of 3D points are required. This extrinsic calibration method is straightforward and can be repeated conveniently whenever there is a change in the camera positioning. The calibration procedure is scalable because: 1) the calibration results remain very accurate with an increase in the number of cameras and because 2) this procedure may be applied to smaller or larger working volumes with minimal modifications to the calibration target. Complete coverage of the working volume is easy to achieve with no coplanarity concerns, as shown in Figure 4.17. The black dots correspond to the cloud of calibration points. The compactness of the calibration target allows the calibration to be performed without moving any equipment already present in the workspace and is therefore suitable

for non-empty working volumes. Finally, the procedure imposes very minimal constraints to the camera positioning (wide baseline and large orientation changes are easily accommodated). Only sufficient overlapping between cameras' field of view is required.

Figure 4.17. A model that shows the positioning of all cameras along with all virtual 3D calibration points

4.5.3 Comparison with Other Methods for Multi-Camera Calibration

It is clear, from the accuracy assessments of section 4.5.1 and the qualitative analysis of the above section, that the proposed solution surpasses classical methods that require complex calibration targets [22, 32, 35]. The proposed solution naturally lends itself for comparison with alternative self-calibration methods of Chen *et al.* [36], Ihrke *et al.* [37] and Svoboda *et al.* [38] that all utilize a similar cloud of virtual calibration points.

The technique used to determine the initial estimate of the extrinsic camera parameters in the method of Chen *et al.* [36] is similar to the proposed method. Unfortunately, cameras are not precisely synchronized and no systematic rules are enounced regarding the order in which links are scaled. Consequently, very high average reprojection errors, from 16 to 38 pixels, are reported in [36]. In contrast, our approach yields to an initial estimate which admits an average reprojection error never above 5 pixels in all tests executed as reported in Table 4.2. Another major distinction resides in the fact that Chen *et al.* applied iteratively an extended Kalman filter as the final optimization step, instead of a bundle adjustment. The final average reprojection error reported in [36] is always above 1 pixel of precision, while our method admits a final accuracy under ½ pixel.

In the method of Ihrke *et al.* [37], an initial estimate of the extrinsic parameters is also obtained from a camera graph analysis, except that closed cycles are considered, rather than vector intersections. The first part of their validation procedure was performed using synthetic data rather than real-data which, in a sense, is not meaningful to assess the registration accuracy of real-world camera hardware that admits several forms of distortions not present in simulated data. When applying their procedure to the calibration of a real multi-camera system, they report a final accuracy which is always above 1 pixel and sometimes even above 5 pixels, which is 2 to 10 times less accurate than the results obtained with the method proposed in this thesis. This higher error can be attributed to multiple factors. Their camera graph analysis does not take into account the fact that some links are known with less accuracy than others. They utilize a bundle adjustment as their final optimization step but their implementation is unsuccessful at achieving sub-pixel precision because it is reduced to the optimization of only the camera rotation and translation parameters and not the cloud of 3D calibration points.

Finally, the method proposed in this thesis admits competitive accuracy with respect to the results reported by Svoboda *et al.* [38] where sub-pixel accuracy is achieved for networks of up to 16 cameras. In their approach, even the intrinsic parameters and distortion coefficients are self-calibrated using solely the cloud of points

acquired from waving a red led. In a sense, this is an advantage with respect to our method except for the fact that their approach is very expensive computationally. Indeed, a wait time of 60 to 90 minutes is reported in order to calibrate a multi-camera system, not counting the computational time required to extract the image matches from the multiple video files. In our approach, the image matches identification remains relatively slow (approximately 15 minutes for 8 cameras at 640x480) but the extrinsic calibration is achieved in less than 20 seconds. This is a key advantage for multi-camera networks that frequently need to be re-calibrated.

Overall, the proposed calibration technique is competitive with existing advanced techniques for multi-camera calibration [36-38]. The proposed method is convenient, accurate and can calibrate effectively a wide variety of multi-camera networks. The remaining sub-sections provide further testing under critical camera configurations to show the robustness and the generality of the proposed solution.

4.5.4 Critical Camera Configurations

Test case 6 exhibited a very special situation where three cameras are almost co-linear but still yield to accurate calibration. This very particular case should have failed due to our incapacity of intersecting parallel vectors. However, such failure only occurs when cameras are precisely aligned. To demonstrate examples of numerical failures for this very special configuration, test case 6 has been repeated twice with precisely aligned cameras. Results are shown in Table 4.4 and Table 4.5. Prior to the bundle adjustment (Table 4.4), results are totally inaccurate. This inaccuracy prevents the bundle adjustment to converge to sub-pixel accuracy. Indeed, in Table 4.5 the bundle adjustment converges but to a final average reprojection error of 1.04 and 3.79 pixels in the two trials that were made.

Table 4.4. Examples of numerical instabilities, prior to the bundle adjustment, that occur for the case of 3 precisely co-linear cameras

Camera id	Reprojection error (1st trial) [pixels]			Reprojection error (2nd trial) [pixels]		
	average	std. dev	max	average	std. dev.	max
c0	0.7365	1.2617	6.2984	4.0598	4.2967	11.4465
c1	7.2344	13.8929	72.3223	53.7698	57.6931	150.7241
c2	13.6850	9.4925	39.3982	57.7435	17.7884	81.4626
All cameras	**5.6786**	**10.3504**	**72.3223**	**35.3670**	**37.8471**	**150.7239**

Table 4.5. Numerical instabilities, for the case of 3 precisely co-linear cameras, remain even after the bundle adjustment

Camera id	Reprojection error (1st trial) [pixels]			Reprojection error (2nd trial) [pixels]		
	average	std. dev	max	average	std. dev.	max
c0	0.1159	0.0896	0.5569	0.1852	0.1434	0.6026
c1	0.3816	0.4082	2.9197	3.2659	1.7331	6.9529
c2	5.9279	1.0934	11.5288	11.6418	2.4528	21.9629
All cameras	**1.0402**	**0.4978**	**11.5288**	**3.7895**	**1.5707**	**21.9629**

This fabricated example shows the importance of detecting numerically unstable cases of co-linearity regarding vector intersections of critical camera triplet configurations. In larger camera networks, it is almost unlikely that all cameras will be co-linear although sub-groups of cameras may be. Test case 9, of Figure 4.15c, exhibits a situation where 3 cameras out of 4 were co-linear. Table 4.6 shows the calibration results for a camera network similar to test case 9. On the left side of Table 4.6, no detection is made to detect co-linear vectors and an unacceptable calibration is obtained with an average reprojection error of 42 pixels. On the right side of Table 4.6, it is shown that avoiding the intersection of co-linear vectors improves, almost 15 times, the calibration such that an average reprojection error of 2.9 pixels can be achieved, for the initial estimate, under the exact same camera network.

Table 4.6. Test case 9 revisited to show the importance of detecting and avoiding cases of co-linear camera triplets (results shown prior to the bundle adjustment optimization)

Camera id	Reprojection error without avoidance of co-linear camera triplets [pixels]			Reprojection error with avoidance of co-linear camera triplets [pixels]		
	average	std. dev	max	average	std. dev.	max
c0	2.5953	3.6594	21.9626	0.6578	0.3734	1.6634
c1	50.5562	56.8650	209.2420	4.8254	2.3489	9.5265
c2	147.3603	27.7315	288.1373	6.2320	0.8893	7.4704
c3	32.1386	39.1203	116.4889	2.2075	1.0148	6.6232
All cameras	**42.1650**	**37.7475**	**288.1373**	2.9191	1.3815	9.5265

This decisive difference in the accuracy of the initial estimate significantly impacts the ability of the bundle adjustment to converge. Indeed, on the left side of Table 4.7, the bundle adjustment fails to converge to sub-pixel accuracy while on the right side of Table 4.7, the bundle adjustment converges to an average reprojection error of 0.29 pixel. Therefore, by the sole detection of cases of co-linear camera triplets, the final camera calibration accuracy is almost 5 times better.

Table 4.7. Test case 9 revisited to show the importance of detecting and avoiding cases of co-linear camera triplets (results shown after the bundle adjustment optimization)

Camera id	Reprojection error without avoidance of co-linear camera triplets [pixels]			Reprojection error with avoidance of co-linear camera triplets [pixels]		
	average	std. dev	max	average	std. dev.	max
c0	0.1529	0.0948	0.5221	0.1488	0.0956	0.6123
c1	1.9363	0.6978	3.6609	0.4242	0.2452	1.8700
c2	2.5869	0.4080	4.2910	0.3589	0.2382	1.7976
c3	1.6729	0.5975	3.7172	0.2796	0.1599	1.1055
All cameras	**1.3942**	**0.5123**	**4.2910**	**0.2877**	**0.1830**	**1.8700**

4.5.5 Importance of the Enhanced Weighted Graph Analysis

Besides the detection of co-linear camera triplets, other enhancements were implemented to increase the overall robustness of the proposed method. In particular, our hybrid approach, which relies on the scaling of links using both vector intersections and link concatenations, allows the calibration of networks that contain many missing

(unknown) links. This enhancement was earlier studied, theoretically, in the examples of Figure 4.8 and Figure 4.13 and is completely transparent from a user perspective. The sole purpose of this enhancement is to allow the calibration of multi-camera networks which admits fewer overlaps among cameras. Furthermore, eliminating links with abnormal errors (high average reprojection error) helps minimizing the error propagation over other links. This was never really observed in our tests because a link is solved only if a sufficient number of matches were found for a particular pair.

The fact that links are scaled in a breath-first manner, using an ordered queue, can also attenuate slightly the average reprojection error of the global network. Indeed, if links were scaled in an arbitrary order or, critically, in a depth-first manner, the error propagation would typically be higher. An example of such a situation was fabricated in the example of Figure 4.18. To emulate a depth-first search, all redundant links were manually disabled to force links to be scaled in depth. Table 4.8 shows, in this particular example, that issuing a search in a breath-first manner yields to an average reprojection error of 3.94 rather than 5.92 if the search was issued in a depth-first manner. Of course, in this example, the improvement is not significant and thus the bundle adjustment converges, in both cases, to sub-pixel accuracy without any difficulty. However, for very large camera networks this enhancement can certainly be more predominant.

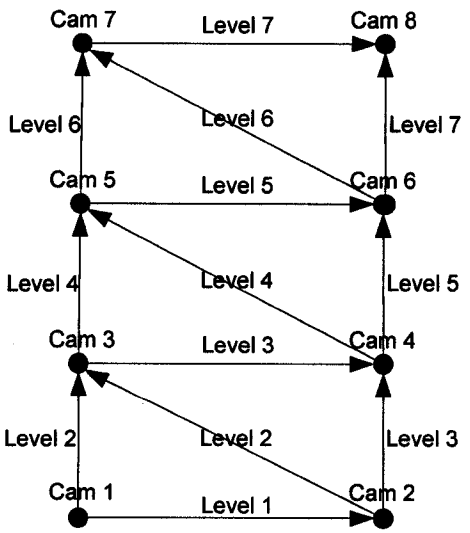


Figure 4.18. Calibration of a camera network with minimal number of links found

Table 4.8. Calibration statistics when, respectively, all links, and a minimal number of links are enabled

	Reprojection error before bundle adjustment [pixels]		
	average	**std. dev**	**max**
Calibration with minimal number of links (depth-first)	5.9244	2.5843	27.1858
Calibration with all links enabled (breath-first)	3.9367	1.8315	22.7446

4.5.6 Estimation of the Absolute Global Scale Factor

Representing the final calibration using meaningful units is a clear asset because it provides realistic and absolute dimensions about reconstructed objects. Section 4.3.7 introduced an efficient method to estimate the absolute scale factor using a dual LED, rather than a single LED, calibration target. The purpose of this section is to evaluate the use of this new calibration target to estimate the absolute scale factor of any given calibrated multi-camera network.

For this test, a network composed of 8 cameras, similar to Test case 14 of Figure 4.16d, has been used. Three different video sequences have been recorded using three different configurations of the dual LED calibration target. Video sequence 1 was recorded using the target configuration of Figure 4.12c with a distance between LEDs measured at 13 cm. Video sequence 2 was recorded using the configuration of Figure 4.12b with a distance between LEDs measured at 46 cm. Finally, video sequence 3 was recorded using the configuration of Figure 4.12a with a distance between LEDs measured at 76 cm.

In Table 4.9, video sequence 1 (LED distance = 13 cm) was used to calibrate the camera network. The fully scaled calibration network was then cross-validated using video sequences 2 and 3. In Table 4.10 and Table 4.11, a similar experiment was performed using, respectively, video sequences 2 (LED distance = 46 cm) and 3 (LED distance = 76 cm) to calibrate the network. Clearly, the use of a LED distance of 13 cm

is too small, relatively to the working volume dimensions (2.5m x 2.5m x 2.5m), to achieve an accurate estimation of the effective scale factor. Indeed, with the calibration of Table 4.9, a measured distance of 46 cm is modeled at 48.29 cm and a measured distance of 76 cm is modeled at 79.22 cm. Alternatively, when the camera network is scaled using video sequence 2 or 3 (Table 4.10 and Table 4.11), the scale factor estimate is much more accurate. In Table 4.10, a distance of 13 cm is modeled at 12.68 cm and a distance of 76 cm is modeled at 75.34 cm. Similarly, in Table 4.11, a distance of 13 cm is modeled at 12.86 cm and a distance of 46 cm is modeled at 46.53 cm. Both the experiments of Table 4.10 and Table 4.11 led to a stable scale factor as the ratio between the measured and modeled distances, during the validation phase, is always very close to 1.00. The scaling error remains within 1cm of the exact distance which represents less than 0.5% of the working volume dimension.

Table 4.9. Scale factor estimation using video sequence 1 (distance b/w LEDs = 13 cm)

	Video sequence 2	Video sequence 3
Measured distance b/w LEDs	46.0 cm	76.0 cm
Modeled distance b/w LEDs (avg)	48.29 cm	79.22 cm
Modeled distance b/w LEDs (std. dev)	±0.36 cm	±0.79 cm
Number of samples	1451	1302
Scale factor ratio	0.9525	0.9594
Scale factor error	-0.0475	-0.0406

Table 4.10. Scale factor estimation using video sequence 2 (distance b/w LEDs =46 cm)

	Video sequence 1	Video sequence 3
Measured distance b/w LEDs	13.0 cm	76.0 cm
Modeled distance b/w LEDs (avg)	12.68 cm	75.34 cm
Modeled distance b/w LEDs (std. dev)	±0.20 cm	±0.75 cm
Number of samples	1351	1302
Scale factor ratio	1.0254	1.0087
Scale factor error	+0.0254	+0.0087

Table 4.11. Scale factor estimation using video sequence 3 (distance b/w LEDs = 76 cm)

	Video Sequence 1	Video Sequence 2
Measured distance b/w LEDs	13.0 cm	46.0 cm
Modeled distance b/w LEDs (avg)	12.86 cm	46.53 cm
Modeled distance b/w LEDs (std. dev)	±0.20 cm	±0.40 cm
Number of samples	1351	1449
Scale factor ratio	1.0106	0.9885
Scale factor error	+0.0106	-0.0115

Table 4.12 and Table 4.13 demonstrate scale factor estimation results achieved by manually measuring the metric baseline between a pair of cameras chosen arbitrarily, as alternatively proposed in section 4.3.7. Results were then validated using the same three recorded video sequences. In terms of accuracy, this manual technique performs better than the experiment of Table 4.9 (LED distance = 13.0 cm) but not as well as the experiments of Table 4.10 and Table 4.11. Overall the scale factor estimate using this manual technique remains acceptable but only because cameras are separated by large baselines respectively to the working volume dimensions (this is a requirement imposed by the shape-from-silhouette algorithm).

Table 4.12. Scale factor estimation using manual camera baseline measurement (distance b/w cam0 and cam1 = 136 cm)

	Video Sequence 1	Video Sequence 2	Video Sequence 3
Measured distance b/w LEDs	13.0 cm	46.0 cm	76.0 cm
Modeled distance b/w LEDs (avg)	13.05 cm	47.23 cm	77.53 cm
Modeled distance b/w LEDs (std. dev)	±0.20	±0.38	±0.78
Number of samples	1351	1450	1302
Scale factor ratio	0.9964	0.9739	0.9802
Scale factor error	-0.0036	-0.0261	-0.0198

Table 4.13. Scale factor estimation using manual camera baseline measurement (distance b/w cam0 and cam7 = 221 cm)

	Video Sequence 1	Video Sequence 2	Video Sequence 3
Measured distance b/w LEDs	13.0 cm	46.0 cm	76.0 cm
Modeled distance b/w LEDs (avg)	13.13 cm	47.54 cm	78.04 cm
Modeled distance b/w LEDs (std. dev)	±0.20 cm	±0.38 cm	±0.78 cm
Number of samples	1351	1450	1302
Scale factor ratio	0.9899	0.9676	0.9738
Scale factor error	-0.0101	-0.0324	-0.0261

Two methods to estimate the absolute scale factor of a calibrated multi-camera network have been evaluated. Measuring the baseline between a pair of cameras that admits a large baseline provides an acceptable estimate of the absolute scale factor. However, from a user perspective, this method requires the calibration to be performed in two stages: 1) calibrate the network using a single marker and 2) measure the scale factor. The use of a dual LED calibration target eliminates this second step and, additionally, achieves a better scale factor estimate provided that the distance between

the LEDs is reasonably large in proportion to the working volume dimensions. Furthermore, since two points per frame can be recorded (two LEDs), the video acquisition routine can be shortened almost by half (i.e. 90 seconds of video recording instead of 3 minutes), therefore accelerating the overall calibration procedure.

4.6 Chapter Summary

In this chapter, a convenient procedure was developed to achieve full calibration of a multi-camera network. The intrinsic parameters were computed once using a classical method that requires multiple frames, directly acquired from streaming video, of a small 2D checkerboard pattern. Extrinsic parameters were solved using a custom method that relies on the creation of a cloud of virtual calibration points. Experimental results demonstrated that the use of a weighted camera graph analysis is very robust and provides accurate calibration estimates before and after the bundle adjustment. Overall, a reprojection accuracy of up to ½ pixel is achieved for networks containing as much as 8 cameras. This approach also meets all pre-established requirements of scalability, ease-of-use and non-cumbersomeness, required by non-expert users. Finally, our innovative dual-marker target allows extrinsic calibration and estimation of the global scale factor in a single step.

In summary, this work resulted in the development of a versatile software framework for precise multi-camera calibration. This framework was primarily designed for the accurate calibration of a markerless motion capture system. However, this solid approach also finds applications in a variety of multi-camera systems ranging from stereo to large scale multi-camera networks.

Chapter 5. Volumetric Reconstruction and Coloring

The previous two chapters achieved the design of a synchronized and calibrated multi-camera setup. This chapter focuses on volumetric reconstruction from multi-camera video data using shape-from-silhouette. In Chapter 2, it has been established that a volumetric representation, rather than multiple 2D images, of a human performer results in a better cue for human motion capture. Kehl *et al.* [21] even mention that multi-view 2D images, solely, provide weak cues, thus resulting in complex and computationally demanding post-processing analysis to extract human posture information. The process of volumetric reconstruction, using shape-from-silhouette, was previously introduced in section 2.5. Synchronized video sequences of a performer are acquired and pre-processed to extract the human silhouette in each view. With the accurate knowledge of the calibration parameters for all cameras, synchronized silhouettes are intersected in 3D to obtain a voxel representation of the performer, which can later be used to extract high-level kinematics information about the human posture.

This chapter is subdivided as follows: In the first section, various silhouette extraction methods are evaluated. The second section elaborates on a practical shape-from-silhouette implementation which determines the binary occupancy information of all voxels that subdivide the working environment. In the third section, a voxel coloring scheme is proposed and provides a supplementary cue to higher-level gesture analysis modules. Voxel reconstruction and coloring results are presented in the final section, with particular interest to the piano pedagogy application.

5.1 Silhouette Extraction

The extraction of human silhouettes from images with a complex background is a research area on its own. In the current project, the silhouette extraction activity was pursued by a colleague [8-10]. In many vision-based motion capture systems, techniques

that rely on the statistical modeling of an empty background are very popular although not optimal. Indeed, background subtraction techniques require the background to remain static throughout the entire acquisition [8]. Under this constraint, it is never permitted, for a performer, to move any object within the working environment. Furthermore, a simple change in the working environment's lighting may compromise many of these algorithms. The problem of shadow pixels being classified as foreground is also difficult to handle. Nevertheless, an example of background subtraction is shown in Figure 5.1 to demonstrate that acceptable results can be obtained using this technique provided that the foreground object sufficiently contrasts with respect to the background.

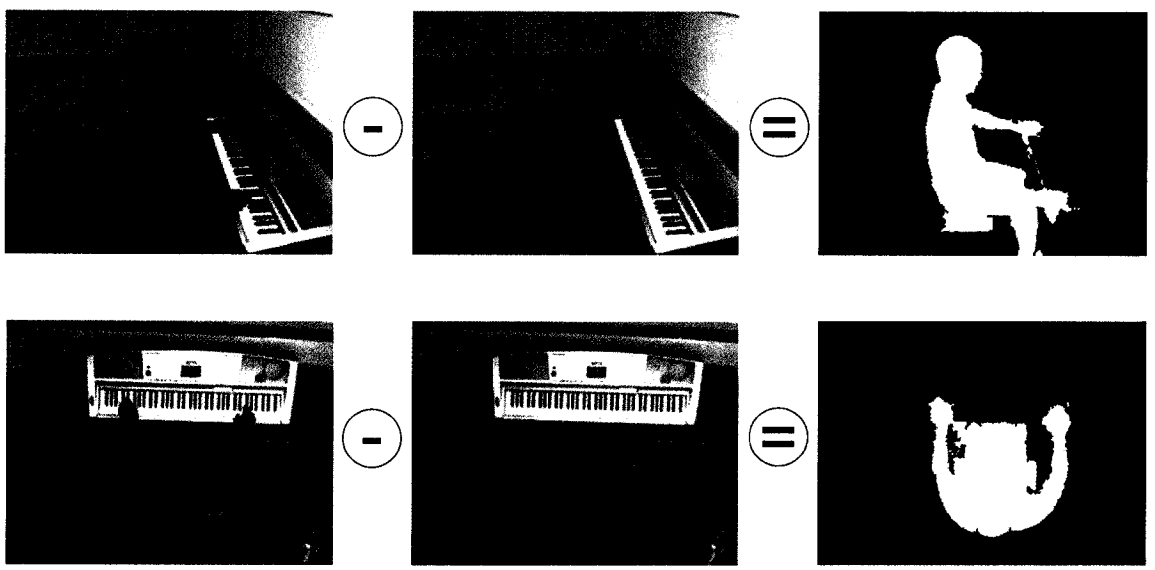


Figure 5.1. Process of background subtraction for silhouette extraction

As an alternative to background subtraction, region-based segmentation allows the decomposition of an image into several regions based on color and texture similarities between neighboring pixels. Recent advances related to region-based segmentation are very promising [9, 10]. This segmentation scheme allows the pianist silhouette to be extracted without any prior knowledge of the background. Evolving backgrounds are therefore permitted. Examples of region-based segmentation are shown in Figure 5.2 and were generated using the implementation described in [9, 10]. Regions of interests are manually chosen for the first frame of video but the tracking is fully automated over all of the remaining frames. While this segmentation scheme provides many clear

advantages, it admits the drawback of being very demanding computationally; therefore resulting in long wait times in order to generate silhouette data especially over long video sequences from multiple cameras.

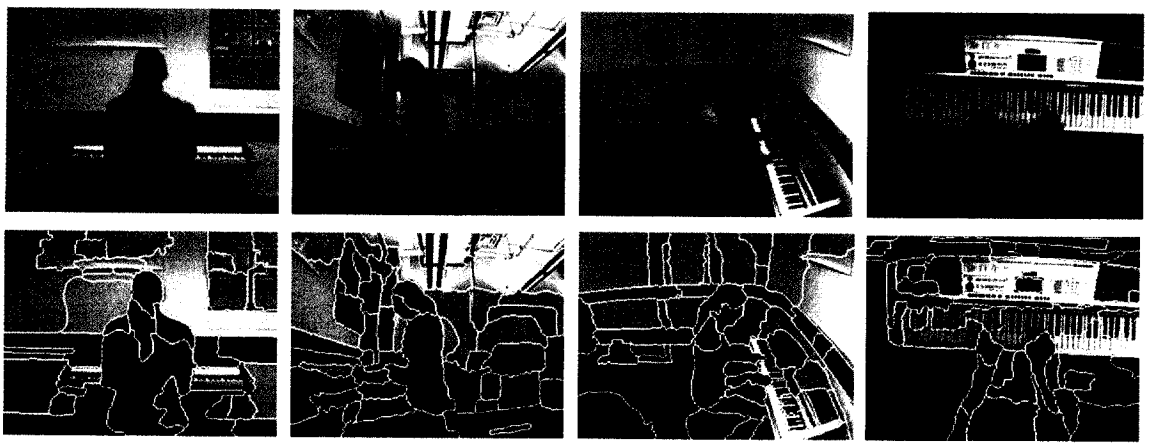


Figure 5.2. Example of region-based human body segmentation without any prior knowledge of the background

In this work, two segmentation schemes were used for volumetric reconstruction. A statistical segmentation implementation [8], based on Mixture of Gaussians, was primarily used because of faster execution time. In this implementation, the background is learnt during the first 10 seconds of video and then remains static for the rest of the acquisition to avoid the problem of non-moving foreground body parts being gradually incorporated to the background model [22]. Some reconstruction examples featuring a novel region-based segmentation implementation of [9, 10] were also produced with the objective of reducing, to some extend, the constraints imposed on the working environment.

5.2 Voxel Occupancy Classification

To compute a binary (occupancy) voxel model of the performer, the working volume is subdivided into many equal-size voxels at a desired resolution (i.e. 64^3, 128^3 or even 256^3). Our implementation [50] utilizes single resolution voxels because multi-

resolution (octree-based) approaches are difficult to post-process. The 3D position of each voxel is calculated with respect to a reference camera acting as the world reference frame. This allows all voxels to be projected over all silhouette images thus leading to the binary classification of each voxel as either "background" or "foreground".

An exact shape-from-silhouette algorithm was proposed in Algorithm 2.1. This algorithm assumes perfect calibration and perfect silhouette images. In practice, the accuracy of shape-from-silhouette depends on both:

- The accurate projection of each voxel over each image plane using the multi-camera calibration parameters.
- The robust occupancy evaluation of each voxel projected over all silhouette images.

The accuracy of the calibration was assessed in the previous chapter. However, silhouette images are never perfect and can sometimes be significantly noisy. This section highlights specific details about the implemented shape-from-silhouette algorithm that takes into account possible inaccuracies regarding silhouette data.

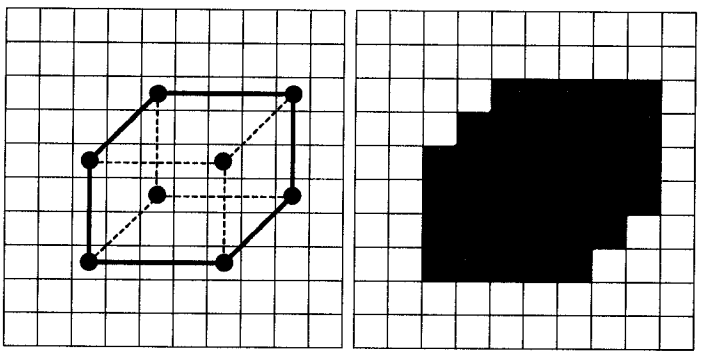
5.2.1 Fast Computation of Voxel Projections

In order to determine the occupancy state of a voxel, it first needs to be projected over each silhouette image. In this work, we analyzed the use of a few distinct methods to achieve this task.

Method 1: Computing the Convex Hull of a Projected Voxel

An exact method to calculate a voxel projection is shown in Figure 5.3. All eight vertices of a voxel are first projected over the image plane, as shown in Figure 5.3a. The convex hull of the projected vertices is then computed, as per Figure 5.3b. Available

implementations exist for convex-hull calculation and can be found, for example, in the OpenCV library [28].

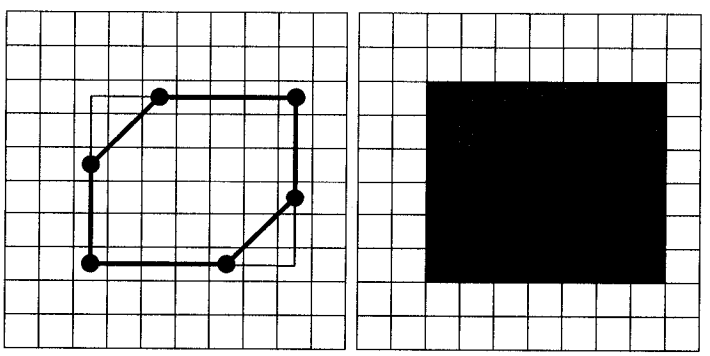


(a) (b)

Figure 5.3. Projection of a voxel in an image

Method 2: Computing the Bounding Box of a Projected Voxel

A significant problem regarding convex hull calculation resides in very high computational requirements. As an alternative, a voxel projection can be approximated by calculating its bounding box, as shown in Figure 5.4a, using the min and max values in both X and Y coordinates. With this approximation, few pixels outside the convex hull but inside the bounding box will be mistakenly included within the projection, as shown in Figure 5.4b. However, it has minimal incidence on the final classification of a voxel projection because of the relatively small size of the projections in the images when the voxel model has a sufficiently high resolution.



(a) (b)

Figure 5.4. Bounding box approximation of a voxel projection

Method 3: The Sampling a Voxel Projection

Cheung *et al.* [18] raised an issue related to the high number of pixels that need to be visited, with both the convex hull and the bounding box approaches, to evaluate the occupancy state of each projection. To reduce the number of pixels to be tested, they proposed the Sparse Pixel Occupancy Test (SPOT) where each voxel projection is uniformly sampled. Thus, the voxel occupancy evaluation is accelerated by visiting only a subset of the pixels pertaining to each voxel projection. Cheung *et al.* estimated that for a working environment configuration where a voxel projects in an image region of, in average, 10 pixels, only 2 samples are sufficient to evaluate the voxel occupancy with a misclassification rate below 1%. Kehl *et al.* [20] chose to utilize only the projection of voxel centers along with the 4 nearest neighboring pixels.

Accelerated Voxel Projection Using Pre-Computed Lookup Tables

In all of the above-mentioned methods, it is advantageous to encode information about all voxel projections into pre-computed lookup tables such that these computations are not repeated continuously for each frame of video. Table 5.1 shows the memory requirements involved in using lookup tables (LUT), for each of the aforementioned methods, using 8 cameras and a standard voxel resolution of 128^3. The convex hull approach requires the recording of 6 image points delimiting the boundaries of each voxel projection, leading to a significantly large LUT of 384 MB. Unfortunately, even with the knowledge of those 6 boundary points, it is not obvious, from a computational perspective, to determine which pixels are contained inside the closed contour formed by those 6 points. On the other hand, the bounding box approach leads to a smaller-size LUT (128 MB) and a simple scan of the bounding box ranges, defined by the min and max values in both X and Y directions, is required to determine the pertinent pixels to be evaluated. The SPOT implementation of Cheung *et al.* [18] is advantageous because fewer pixels need to be evaluated per voxel projection. However, very few samples per voxel projection can realistically be kept without exceeding any reasonable memory

requirements and, thus, resulting in less robustness against noisy silhouette data. Finally, the voxel center approach is probably the most memory efficient method, but is not robust against noisy silhouette data.

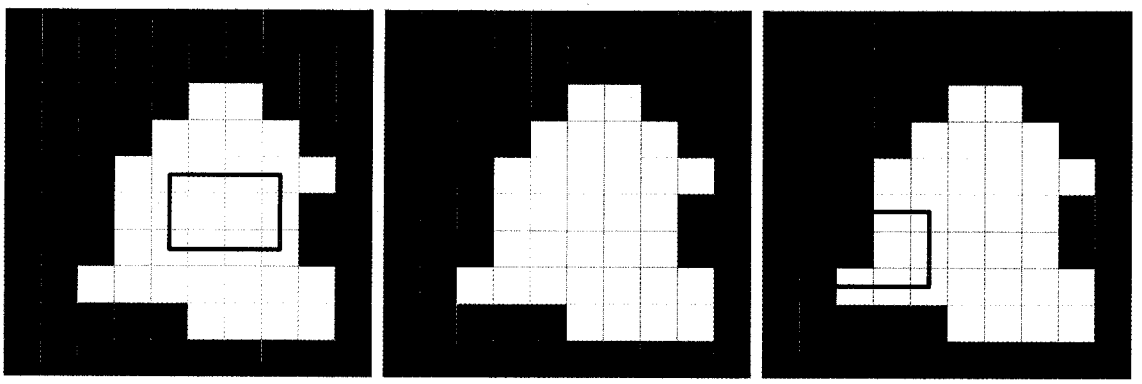
Table 5.1. Memory requirements for different voxel projection LUTs

	Memory requirements per voxel projection [Bytes/projection]	**Total memory requirements 128^3 resolution, 8 cameras [MB]**
Convex hull	$\frac{6points}{projection} * \frac{2coords}{point} * \frac{2bytes}{coords} = \frac{24bytes}{projection}$	$\frac{24bytes}{projection} * 8cams * 128^3 voxels = 384MB$
Bounding box	$\frac{4coords}{projection} * \frac{2bytes}{coords} = \frac{8bytes}{projection}$	$\frac{8bytes}{projection} * 8cams * 128^3 voxels = 128MB$
SPOT (4 samples)	$\frac{4points}{projection} * \frac{2coords}{point} * \frac{2bytes}{coords} = \frac{16bytes}{projection}$	$\frac{16bytes}{projection} * 8cams * 128^3 voxels = 256MB$
SPOT (2 samples)	$\frac{2points}{projection} * \frac{2coords}{point} * \frac{2bytes}{coords} = \frac{8bytes}{projection}$	$\frac{8bytes}{projection} * 8cams * 128^3 voxels = 128MB$
Voxel center	$\frac{1point}{projection} * \frac{2coords}{point} * \frac{2bytes}{coords} = \frac{4bytes}{projection}$	$\frac{4bytes}{projection} * 8cams * 128^3 voxels = 64MB$

In the proposed implementation, full bounding boxes are used because they can efficiently be encoded in lookup tables and because all pixels are evaluated thus improving the robustness of the reconstruction under non-optimal silhouette data and, most importantly, because they are needed for proper occlusion detection in our voxel coloring algorithm that will be described in section 5.3.

5.2.2 Voxel Projection Evaluation

Voxel projections are evaluated against the pre-calculated silhouette images, for all camera views. A voxel projection can either lie over a foreground (Figure 5.5a), a background (Figure 5.5b), an edge image region (Figure 5.5c) or outside the image boundaries. In this latter case, this particular view cannot vote on the occupancy state of the voxel of interest.



(a) (b) (c)

Figure 5.5. Voxel projection evaluation: white pixels correspond to the silhouette of an arbitrary foreground object, black pixels correspond to the background, and the red box corresponds to a bounding-box approximation of a voxel projection.

From a theoretical point of view, in a single-resolution shape-from-silhouette algorithm, as presented earlier in Algorithm 2.1, each voxel is either classified as "foreground" or "background" based on the intersection of all voxel projection binary classifications. According to this algorithm, there is no "edge" state and, thus, "edge" projections need to be reclassified as either "background" or "foreground", based on the ratio of silhouette versus background pixels inside the projection. To the opposite, in the case of a multi-resolution shape-from-silhouette, an "edge" state is required because it determines if a voxel needs to be subdivided or not.

In our single-resolution shape-from-silhouette implementation, we found useful to keep an "edge" state at the voxel projection level which is to be resolved only at the voxel classification level (see section 5.2.3). This is done in order to increase the robustness against noisy silhouette data. Therefore, voxel projections are being evaluated as follows:

- *Foreground*: A projection is classified as "foreground" if at least 50% of the pixels that pertain to the projection, over an image, are silhouette pixels.
- *Background*: A projection is classified as "background" if less than 20% of the pixels that pertain to the projection, over an image, are silhouette pixels.
- *Edge*: Whenever the silhouette pixels count is smaller than 50% but greater than 20%, the projection is classified with a special "edge" classification value.

Any voxel projection that counts at least 50% of silhouette pixels is naturally classified as "foreground". However, when a voxel projection counts less than 50% of silhouette pixels, it is not systematically classified as "background". Instead, the range [20%-50%] is introduced and is used as temporary "edge" classification. This range was chosen subjectively in order to compensate for the high incidence of voxel projections misclassified as "background" over the final voxel occupancy classification.

5.2.3 Voxel Classification

The intersection of all voxel projections is used to determine the final voxel occupancy state. Computing such intersection implies that a voxel is classified as background whenever it projects over a region that pertains to a background region in one or more images. This condition is highly restrictive, under sub-optimal silhouette data, and can be relaxed if the number of cameras in use is sufficiently high [23]. In our setup however, only 8 cameras are used and, thus, relaxing this condition, by requiring a voxel to project in a background image region in, minimally, two or three views for it to be classified as "background", is near impossible. Instead, we propose to take advantage of the "edge" classification, obtained during the evaluation of some voxel projections, as shown in Algorithm 5.1. The final voxel classification is achieved according to the following rules:

- *Foreground:* A voxel is classified as "foreground" if all voxel projections are evaluated as "foreground". In addition, a voxel remains classified as "foreground" if only one voxel projection is evaluated as "edge" provided that all other projections are evaluated as "foreground".
- *Background:* A voxel is classified as background if at least one voxel projection is evaluated as "background" or if at least two projections are evaluated as "edge".

In other words, the main distinction with respect to the standard algorithm is the increased tolerance in accepting to classify a voxel as "foreground" even if, in one view, it projects onto an "edge" region where the silhouette pixels count is low (20% to 50%). Also, a final verification is made to ensure that a minimal number of views (3 cameras in our implementation) see the voxel as "foreground" in order for it to be classified as "foreground". This is done for the obvious purpose of avoiding a voxel to be misclassified as "foreground" when this voxel projects outside the image boundaries in too many views.

Algorithm 5.1. Final Single-resolution shape-from-silhouette algorithm

```
allVoxels[] = InitVoxelMap(desiredResolution);
    for all voxel in allVoxels[]
        edgeCnt=0;                      // count the number of "edge" projections
        numVoters=0;                    // count the total number of voters
        voxelOccupancy = foreground;  // initially assume the voxelOccupancy to be foreground
        // scan all camera views and reclassify the voxelOccupancy if one or more cameras see the voxel
        // as background or if two or more cameras see the voxel as "edge"
        for all cameras
            voxelProj = GetVoxelProjFromLUT(voxel, camera);
            projectionState = EvaluateVoxelProj(camera.GetImage(), voxelProj);
            if (projectionState == OutsideImageBoundary)
                continue;  // skip this camera
            else if (projectionState == foreground)
                numVoters = numVoters+1;
            else if (projectionState == edge)
                numVoters = numVoters+1;
                edgeCnt = edgeCnt+1;
            else if (projectionState==background)
                voxelOccupancy = background;
                break; // abandon the inner loop (camera loop)
            end if
            if (edgeCnt >= 2)
                voxelOccupancy = background; // abandon the inner loop (camera loop)
                break;
            end if
        end for
        // special case: to ensure that the voxel is not classified as foreground if the voxel projections
        // lie outside the image boundaries in too many views
        if ((voxelOccupancy == foreground) AND (numVoters < 3))
            voxelOccupancy = background;
        end
        voxel.Classify(voxelOccupancy);
    end for
```

5.3 Voxel Coloring

Binary voxel occupancy information can be used as the only cue for human gesture analysis [16, 17, 18, 23]. However, there exist numerous human postures that cannot be estimated using solely voxel occupancy information. For example, Figure 5.6a shows a scenario where the two arms cannot at all be recognized. However in Figure 5.6b, this ambiguity is unfolded using voxel coloring which can serve as a logical complementary cue to voxel occupancy information. Figure 5.7a and Figure 5.7b show another example where binary occupancy voxels are not sufficient to clearly disambiguate between two possible neck rotations but is however unfolded with colored models of Figure 5.7c and Figure 5.7d.

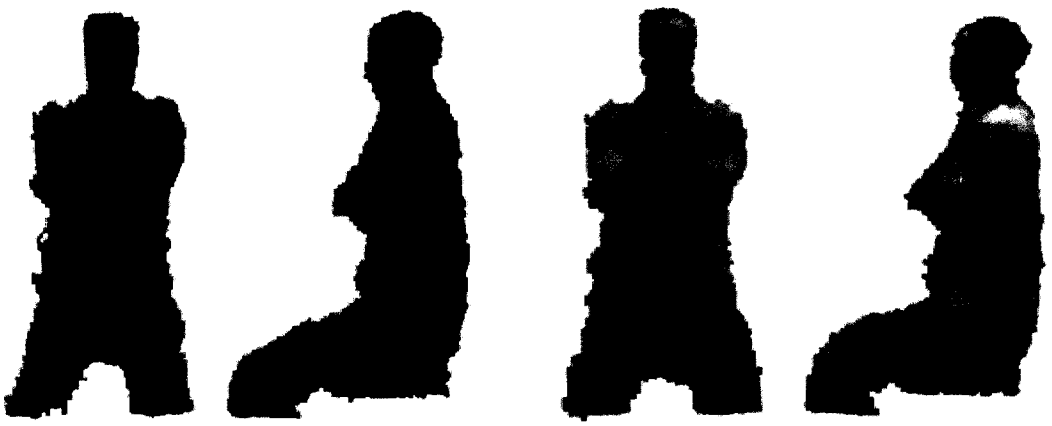


Figure 5.6. Example of voxel coloring used to disambiguate arms position

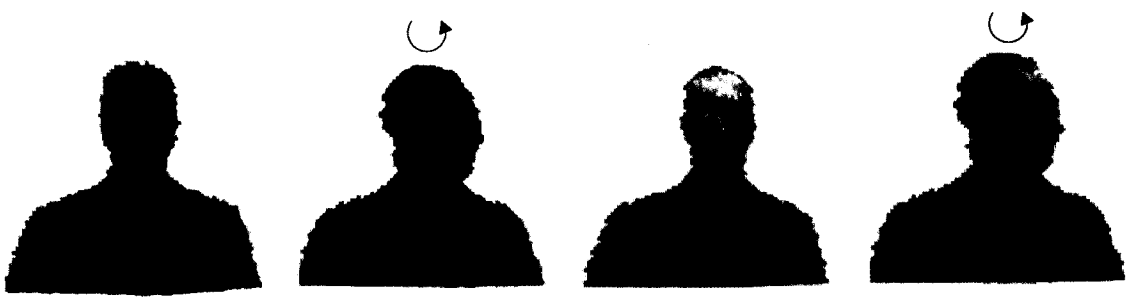


Figure 5.7. Example of voxel coloring used to disambiguate between two distinct neck rotations

5.3.1 Surface Voxel Test

The proposed voxel coloring scheme is applied mainly to the surface voxels because only surface voxels are visible and therefore rendered. A foreground voxel is a surface voxel if at least one of its six nearest neighbors is classified as "background". However, when silhouette data is not optimal or the number of viewpoints is limited, surface foreground voxels may very likely correspond to small background volumes nearby (surrounding) the actual performer. Coloring these particular voxels results in noisier texturing because color characteristics pertaining to the background are mistakenly incorporated into the coloring equation. We propose to enhance the coloring by eroding, in 3D, all surface voxels, thus re-classifying them as background. Applying this procedure once or twice, depending on the actual resolution of the voxel map and on the level of confidence about the reliability of the extracted silhouette, is sufficient and results in excellent voxel coloring.

Furthermore, it may be advantageous to propagate surface voxel color to interior voxels as well. Indeed, as Caillette [22] observed, it can facilitate post-processing kinematics analysis if all voxels, not only the surface voxels, are colored. The 3D erosion procedure can be applied in a loop to color all interior voxels layer-by-layer. In other words, all exterior voxels would be colored and then removed from the model, in order to color the underlying layer of surface voxels until all voxels have been colored.

5.3.2 Occlusion Detection using a Depth Buffer

The attribution of a color to each surface voxel needs to take into account the color information from all views to ensure uniformity and smoothness all around the reconstructed volume of interest [20]. At the same time, views that do not have a direct access to a particular voxel need to be discarded from the coloring decision process. This problem is referred to as occlusion. An occlusion occurs whenever a surface voxel obstructs another surface voxel in a particular view. Figure 5.8 shows an example where

the two back views do not have a direct access to the performer's hands (arrows in red). Therefore, only the views in front of the performer (arrows in green) may participate to the coloring equation of voxels pertaining to the performer's hands.

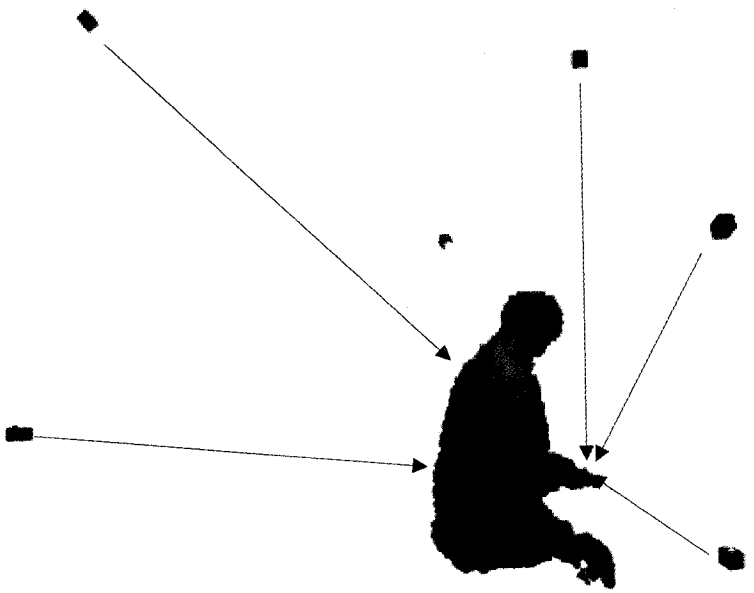


Figure 5.8. Example of a self-occluding human posture

Detecting cases of occlusion is primordial and many approaches have been suggested in the literature. Such methods to compute highly photorealistic colored voxel models are reviewed in [42]. Unfortunately, these methods are highly demanding computationally because they require many traversals of the entire voxel data or are iterative. These methods totally eliminate any real-time system aspirations. Even for offline motion capture, the wait time in generating the colored voxel model, frame-by-frame, remains an inconvenience.

Instead, we propose an effective occlusion detection mechanism based on the use of a depth buffer for all image views. A depth image is initialized for each camera and serves the purpose of maintaining the 3D distance to the closest surface voxel for every image pixel. These depth buffers are computed according to Algorithm 5.2. Initially, all pixels of a depth buffer are initialized at infinity. All surface voxels are traversed once, in no predefined order. For each view, if the voxel projection is comprised within the image

boundaries, the depth separating this voxel from the current camera is calculated. This is achieved by transforming the 3D position of a voxel center such that it is expressed with respect to the current camera frame. The depth is then evaluated as the magnitude of this transformed 3D point ($depth = \sqrt{x^2 + y^2 + z^2}$). Optionally, this information can be pre-computed using lookup tables since viewpoints are static. All depth buffer pixels comprised within the voxel projection are updated according to the newly calculated depth. An update occurs if a pixel already holds a depth value greater than the new calculated depth value.

Algorithm 5.2. Depth buffer creation

```
allDepthBuffers[] = InitDepthBuffers (numCameras);
for all voxel ∈ SurfaceVoxels
    for all cameras
        voxelProj = GetVoxelProjFromLUT(voxel, camera);
        if (IsVoxelInsideImageBoundaries(voxelProj, camera))
            depth = EvaluateVoxelDepth(voxel, camera)
            UpdateDepthBuffer(allDepthBuffers[camera], depth, voxelProj);
        end if
    end for
end for
```

Examples of computed depth buffers are shown in Figure 5.9 for four different views. The black dots represent the 3D depth position to the closest foreground voxel that projects into each pixel in a view. When no foreground voxel projects into a particular pixel, the depth is set to infinity and, therefore, no dot is displayed.

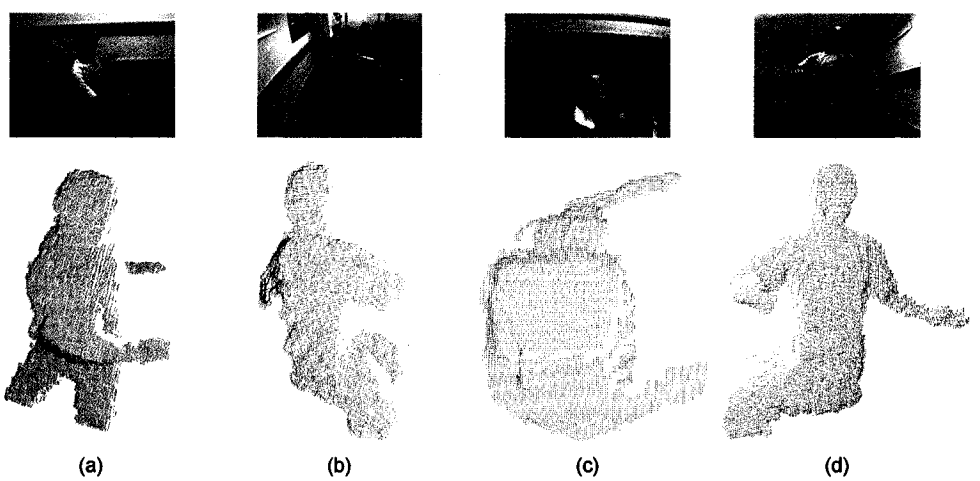


Figure 5.9. Occlusion detection using an image depth buffer for each view

5.3.3 Voxel Visibility Analysis

Once the image depth buffers have been filled, surface voxels are traversed again. The purpose of this second pass is to determine which views will participate in the final color calculation for a particular voxel. An outline of the proposed voxel coloring method is provided in Algorithm 5.3.

Algorithm 5.3. Outline of the voxel coloring algorithm

```
allDepthBuffers[] = ComputeDepthBuffers(numCameras);
allColors[] = InitColorBuffer(numCameras);
for all voxel Є SurfaceVoxels
    numColors=0;
    for all cameras
        voxelProj = GetVoxelProjFromLUT(voxel, camera);
        if (IsVoxelInsideImageBoundaries(voxelProj, camera))
            depth = EvaluateVoxelDepth(voxel, camera);
            if (TestNoOcclusion(voxelProj, depth, allDepthBuffers[camera]))
                allColors[numColors] = EvaluateColor(voxelProj, colorVideo[camera]);
                numColors = numColors+1;
            end if
        end if
    end for
    if (numColors > 0)
        finalColor = EvaluateFinalColor(allColors[], numColors); // equation (5.1)
        voxel.SetColor(finalColor);
    else
end for
```

The depth distance to each surface voxel center, in each view, is compared against the depth value of the corresponding image depth buffer. Testing only the voxel center to detect an occlusion is sufficient here because, in the first pass, calculated depths were written into all pixels pertaining to their voxel projections. Doing so in this second pass would have been redundant. If the calculated voxel depth is equal to the value of the

depth buffer[1], then no occlusion is found. In such case, the color of this voxel projection is evaluated, from the original color video, and added to the color equation. Once all views have been evaluated, the color from all non-occluding (visible) views is averaged (per color channel) to determine the final color for this voxel according to following equation.

$$FinalColor_{RGB} = \frac{\sum_{img=1}^{visible\_proj} ProjColor_{RGB}(img)}{visible\_proj} \tag{5.1}$$

5.3.4 Remarks

This voxel coloring approach is similar to the one proposed by Kehl *et al.* [20] with some key improvements:

1) In Kehl *et al.*'s method [20] only the voxel centers are maintained in the lookup tables instead of the full voxel projections. This approximation leads to an insufficient condition for occlusion detection, especially when the voxel resolution is unbalanced with respect to the working image resolution. For example, if a voxel projects at position (200, 193) and a voxel behind projects at position (201, 192), no occlusion would be detected. However, it is typical that a voxel projection occupies a surface of around 10 pixels, maybe more, depending on the image resolution.

2) For special cases where all views are disqualified, according to the voxel visibility test (of section 5.3.3), the color equation (5.1) cannot be used. Instead, the color contained in the "less occluded view" is assigned as the final color for such a voxel. The "less occluded view" is the one where the difference between

[1] Two depth distances are considered to be equal if they admit a difference of less than the dimension of a voxel (i.e. distance between two consecutive voxel). Optionally, this criterion can be relaxed to twice or thrice the dimension of a voxel for more smoothness.

the voxel's depth and the occluding voxel's depth is the smallest. This special case only occurs for few voxels that are not directly seen by any camera. For example, in Figure 5.8, voxels under the performer's torso and legs are not viewed by any camera (these few voxels are generally unimportant to the overall reconstruction). This enhancement was omitted in Algorithm 5.3 for clarity purposes.

3) In situations where multiple views are not occluded, views in which a voxel projects into an edge region (according to the silhouette occupancy data) are excluded from the voxel coloring equation. This is done in order to prevent, as much as possible, for background colors, in images, to be mistakenly incorporated into the voxel model. This enhancement was also omitted in Algorithm 5.3 for clarity reasons.

4) A mechanism for propagating color to all interior voxels is elaborated by re-applying this coloring procedure layer-by-layer.

A direct comparison of performance with Kehl *et al*'s work will be discussed in section 5.4.2.

5.4 Results

The purpose of this section is to evaluate the results of the aforementioned voxel-based volumetric reconstruction algorithm. These results were achieved by integrating all modules developed throughout this project. The multi-camera system presented in Chapter 3 was utilized to synchronously acquire real video data featuring a human performer at a frame rate of 30 fps. A frame resolution of 320x240 was used because it is sufficient for the purpose of shape-from-silhouette reconstruction. Hence, the multi-camera network was re-calibrated for this resolution. Two different silhouette extraction implementations were utilized to perform binary silhouette data extraction. Results

presented in sections 5.4.1 to 5.4.6 utilize a statistical Mixture of Gaussians segmentation algorithm [8], which requires *a priori* knowledge of the background but operates faster. Results presented in section 5.4.7 were generated using an innovative region-based segmentation algorithm [9, 10] which removes this constraint but is slower to compute. All results were computed with a working volume optimized to best fit the performer. The voxel resolution was set at 128^3 such that each voxel occupies a volume of approximately $(1.5\text{cm})^3$.

This analysis is qualitative and is based on perceptual observations of reconstructed voxel models. However, our evaluation methodology remains formal and is subdivided as followed:

- In section 5.4.1, multiple human body reconstruction results are shown to demonstrate the generality of the proposed method against various human postures.
- In section 5.4.2, our results are compared to results reported by other authors in the field of markerless motion capture.
- In section 5.4.3, some distortion is applied to the extrinsic camera parameters to demonstrate the crucial importance of accurate multi-camera calibration.
- Section 5.4.4 demonstrates the importance of using a sufficient number of viewpoints, well distributed over the full workspace, in order to obtain finer reconstruction results.
- Section 5.4.5 shows successful volumetric reconstruction results using noisy, real-world, silhouette data.
- Section 5.4.6 demonstrates reconstruction results related to the specific application of monitoring a pianist's motion.
- Finally, section 5.4.7 shows reconstruction results using a novel region-based silhouette extraction method [9, 10]. This new segmentation algorithm enables human silhouette extraction in complex and evolving scenes.

5.4.1 General Result Analysis for Various Human Body Postures

Before evaluating reconstruction results in the context of the piano pedagogy application, results generated from arbitrary human postures are first presented to demonstrate the generality of the proposed solution and for comparison purposes with other implementations suggested in the literature. These results are shown in Figure 5.10. In this figure, images from the first row correspond to a frame of original video, for each evaluated posture. The second and third rows show two reconstructed views for each posture.

Figure 5.10a shows a simple example which does not admit any apparent case of occlusion. In this posture, the performer is reconstructed almost flawlessly. In particular, the proposed voxel coloring algorithm is very powerful at reconstructing subtle details such as wrinkles on the t-shirt. The example of Figure 5.10b is more complex because the two arms are crossed and they occlude the torso. This is a clear case where voxel coloring is required to disambiguate the two hands position and, furthermore, to disambiguate that the left arm is under the right arm. Figure 5.10c is another simple example which is well reconstructed as observed in the two displayed views of the reconstruction. Figure 5.10d and Figure 5.10e show a left and a right torso rotation. In these two examples, few cases of self-occlusion can be raised. Indeed, in Figure 5.10d, the right arm occludes the torso for views facing the performer and the upper legs for views positioned on top of the performer. Figure 5.10e is a similar but reversed example. In both examples, occlusions are resolved using the remaining views such that the two hands are successfully separated from the rest of the body. The problem of occlusion is also well handled, from a texturing perspective, since the two arms are colored distinctively from the t-shirt and pants color. Figure 5.10f is the most interesting and complex example. Not only does the left leg and left arm occlude the torso but they also create a local concave region which is difficult to reconstruct using shape-from-silhouette and especially with a limited and fixed number of viewpoints. This concavity is located nearby (in front) the lower-torso. This example results in few background

voxels being misclassified as foreground. These artifacts also slightly perturb the voxel coloring in this concave area but the overall reconstruction remains reliable.

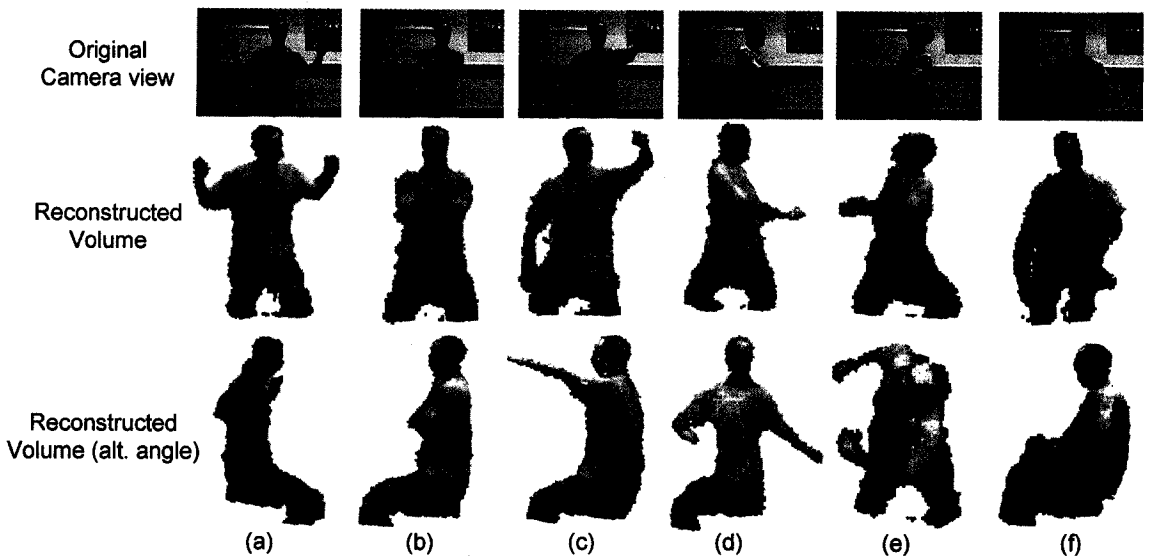


Figure 5.10. Various test cases of full-body reconstruction

Figure 5.11 shows an example where only the head is reconstructed. The working environment was defined to be smaller such that each voxel occupies a smaller volume thus resulting in a reconstruction with finer details. Figure 5.11a and Figure 5.11b show 2 (out of 8) synchronized camera views. Figure 5.11c and Figure 5.11e show two side-views of the reconstructed model. From these two images, we can observe that the shape of the head (occupancy) is reconstructed with good accuracy since the nose and the chin contours are well defined. The voxel coloring is applied with reasonable accuracy and a clear distinction between skin and hair color can be made. Figure 5.11d shows a front view of the reconstructed head. Obviously a finer resolution would be required to reconstruct the facial expression but it is not the purpose of the current project.

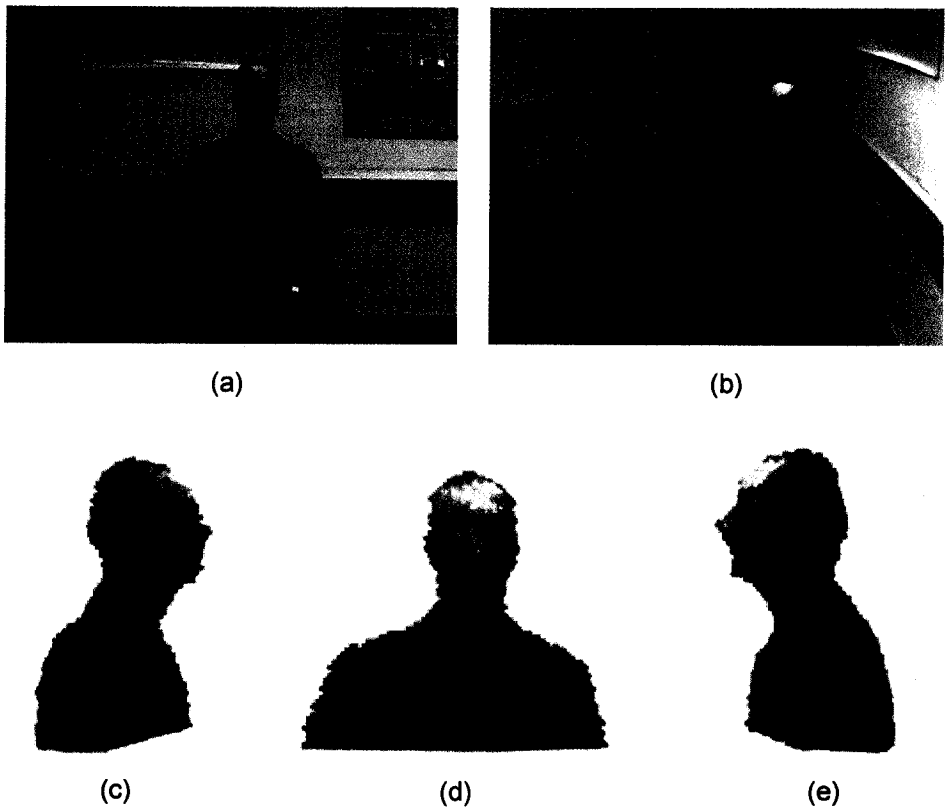


Figure 5.11. 3D head reconstruction with voxel coloring

5.4.2 Qualitative Comparison with Other Volumetric Reconstruction Methods

This section emphasizes on qualitative improvements that are achieved with the proposed implementation in comparison with results demonstrated in other markerless motion capture systems described in the literature. Figure 5.12 shows a comparison between reconstruction results reported in several of these implementations.

Figure 5.12a shows a volumetric reconstruction borrowed from Cheung *et al.*'s paper [19]. Perceptually, this result is very precisely rendered in a photorealistic manner. However, the reconstruction method of Cheung *et al.*'s is achieved by registering iteratively multiple frames of video spatially, and over time as well. Indeed, the reconstructed volume of Figure 5.12a was achieved by asking a performer to stand still on a turntable for 25 frames of video. Many viewpoints are artificially created this way.

However, this is contrary to what we are trying to achieve. Instead, we are interested in reconstructing the human shape independently, frame-by-frame, in order to monitor a moving performer over time. Cheung *et al.* did extend their temporal shape-from-silhouette framework to handle the specific case of moving articulated objects (i.e. the human body) but lacks of robustness are denoted by Sundaresan [23] for postures where there are self-contact (one or more limb touching other body parts).

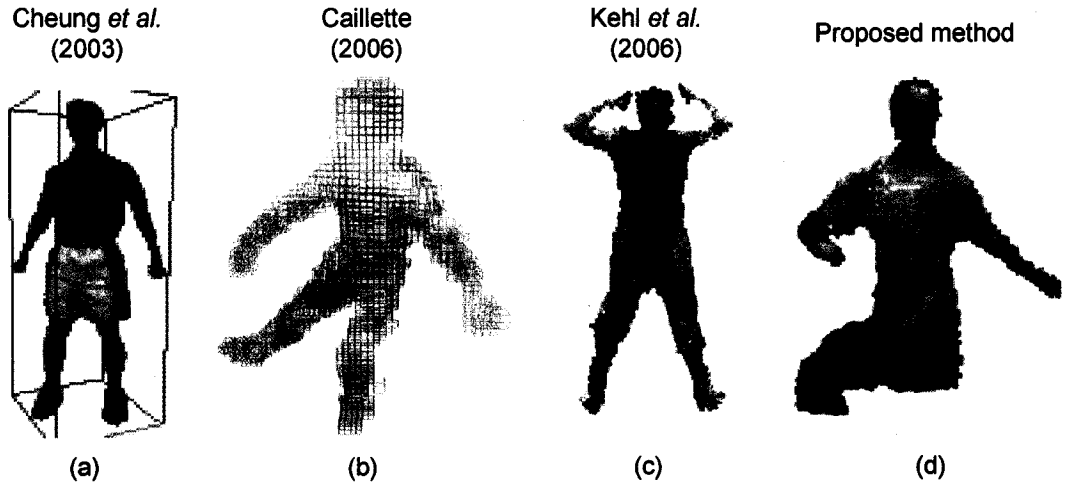


Figure 5.12. Comparison of the proposed voxel reconstruction scheme with other works presented in the literature. Source: (a), (b) and (c) reproduced from [19], [22] and [21] respectively

Figure 5.12b shows a volumetric reconstruction result borrowed from Caillette's work [22]. The voxel coloring method is very limited because no clear voxel visibility test is performed to detect cases of occlusions. In addition, the voxel occupancy information is coarse because very few camera sensors are utilized for the reconstruction, in order for the overall system to operate in real-time. Consequently, the quality of the volumetric reconstruction obtained by Caillette [22] is only acceptable for non-occlusive and simple human postures.

Figure 5.12c shows a volumetric reconstruction result borrowed from Kehl *et al.*'s work [20,21]. Their voxel coloring implementation is similar to the framework proposed in this thesis. However, this result is subjectively of a lower quality than the result achieved in our work and displayed in Figure 5.12d. Indeed, in Figure 5.12c, it is easy to notice that few voxels, belonging to head and legs regions, are incorrectly colored in red

(the color of the t-shirt). This is due to the fact that occlusions are detected only using the center of voxels and not the full voxel projection. The comparison with Kehl *et al.*'s algorithm is further extended in Figure 5.13. Figure 5.13a and Figure 5.13c show two volumes reconstructed using Kehl *et al.*'s original algorithm. In both examples, the applied voxel coloring is noisy. In Figure 5.13b and Figure 5.13d the volumetric reconstruction was achieved using the proposed algorithm. From these latter examples, it is possible to notice that the proposed enhancements perceptually result in a smoother and overally improved voxel coloring.

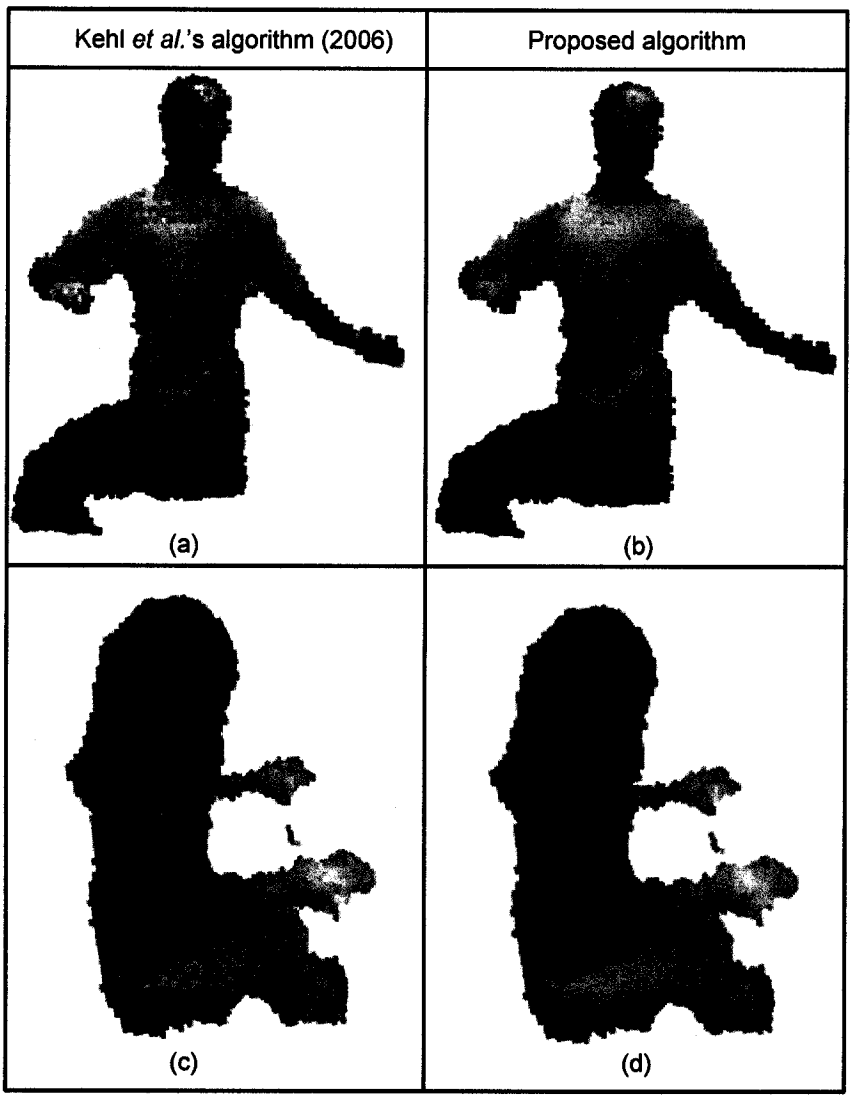


Figure 5.13. Extended comparison between Kehl *et al.* [20] and the proposed voxel coloring algorithm

5.4.3 Importance of Precise Camera Calibration

The objective of this test is to show the importance of precise camera calibration for proper fusion of multi-view images in 3D. Chapter 4 assessed numerically the accuracy of our camera calibration procedure. In this section, we verify qualitatively the effect of calibration inaccuracies to resulting voxel data of reconstructed performers. To do so, all translation parameters (T_x, T_y, T_z) and/or rotation parameters (R_x, R_y, R_z) were distorted, in all cameras, by a random value within the range ±2.5% or ±5% of each initial parameter value respectively. Applying very minimal modifications to the rotation or the translation parameters have important repercussion on the average reprojection error as shown in Table 5.2.

Table 5.2. Impact of introducing noise to the rotation (R) and translation (T) camera parameters on the average reprojection error (in pixels)

Test case [noise level]	Average reprojection error [pixel]	Std. Deviation [pixel]
No noise	0.3508	0.2562
T-Noise: ±2.5%	4.1533	2.2329
T-Noise: ±5%	8.1655	3.7776
R-Noise: ±2.5%	4.3159	2.4667
R-Noise: ±5%	8.5785	4.6116
T-Noise: ±2.5%, R-Noise: ±2.5%	9.4270	3.4750

For each test case, a shape-from-silhouette reconstruction was performed. Figure 5.14a shows a reconstruction result with very precisely calibrated cameras. Very few noisy voxels can be identified in this model and are mainly due to imperfections in the computed silhouette data. In the model of Figure 5.14b, translation parameters were distorted by up to ±2.5% in all cameras. The global shape of the human body remains recognizable and no clear perturbation is heavily noticeable. In the model of Figure 5.14c, the distortion to translation parameters was accentuated to ±5% and we can notice that the right arm starts disappearing and that the head is incorrectly reconstructed. In Figure 5.14d and Figure 5.14e the noise was applied to the rotation parameters instead. Reconstruction imprecision is already noticeable with a perturbation of ±2.5%. The imprecision is significantly accentuated when the perturbation is of ±5%. Finally, Figure

5.14f shows a case where noise (2.5%) was applied to both translation and rotation parameters. This latter case lead, subjectively, to the worst shape reconstruction (partial loss of both arms).

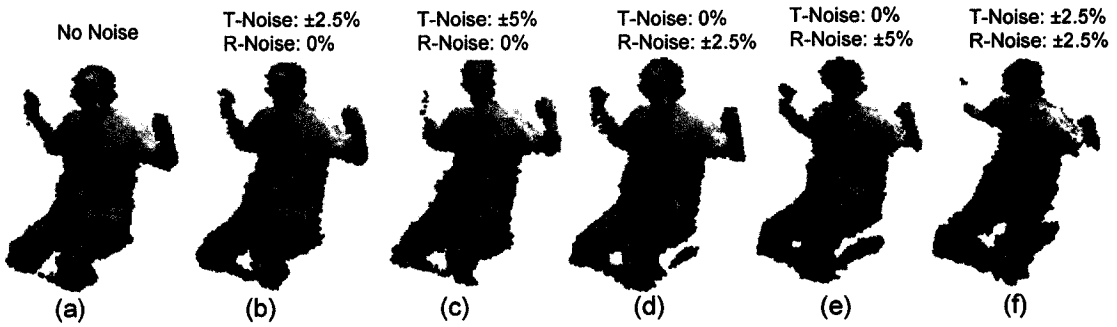


Figure 5.14. Incidence of the camera calibration precision on the reconstruction accuracy

From this analysis, it is clear that calibration accuracy is a significant requirement for adequate shape-from-silhouette reconstruction. A small perturbation of only ±2.5% on the rotation parameters, corresponding to an average reprojection error of approximately 4.32 pixels, is sufficient to introduce noticeable artifacts into the reconstructed volume. This demonstration thus shows the crucial importance of accurate multi-camera calibration with sub-pixel precision when performing shape-from-silhouette reconstruction.

5.4.4 Effect of Reducing the Number of Viewpoints

Besides precise camera calibration, another important factor for shape-from-silhouette reconstruction resides in the number of viewpoints and their positioning. The purpose of this test is to verify the behavior of the implemented shape-from-silhouette algorithm when the number of views is critically reduced. Figure 5.15 shows reconstruction results for multi-camera setups of 3 to 7 cameras. In each test, the views were positioned to surround, as uniformly as possible, the targeted performer.

From these results, it is possible to observe that the number of viewpoints has a decisive impact on the quality of the reconstructed volume. This statement comes from the theoretical definition of shape-from-silhouette. As more views are introduced to the volumetric intersection, the resulting reconstructed volume is smaller, thus yielding to a more refined model closer to the shape of the real performer. When only 3 views are used, as per Figure 5.15a, the reconstructed volume is missing many details. Significant improvements are noticed with the incorporation of only one additional camera, in Figure 5.15b, but the reconstructed volume remains coarse. The volume reconstructed from 5 cameras (Figure 5.15c) is smaller but remains imprecise. Additional details about the human body are discovered by incorporating a 6^{th} and a 7^{th} camera, as per Figure 5.15d and Figure 5.15e. In particular, the use of 7 cameras provides finer details related to the arms and legs.

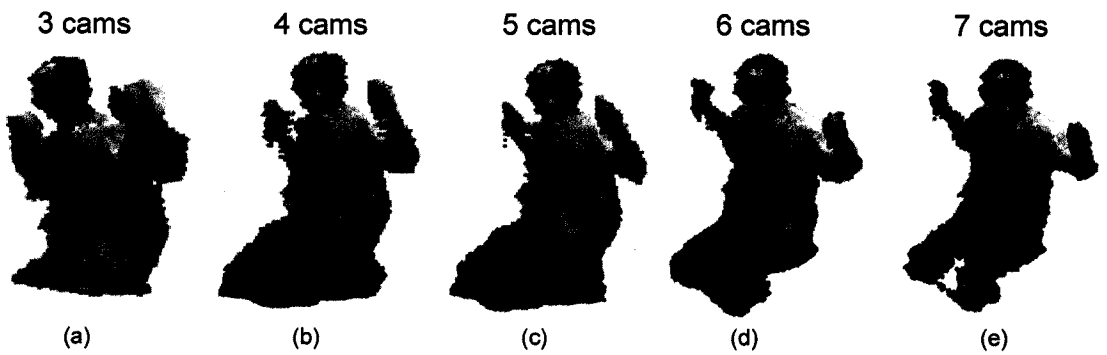


Figure 5.15. Incidence of the number of views in the reconstruction accuracy

At some point, increasing the number of viewpoints will barely improve the resulting reconstruction. The use of many viewpoints can however provide additional robustness because of inherent redundancy among views. It however increases the cost of the application as well as reduces its mobility. A compromise must therefore be made. We observe that 8 cameras are sufficient for full body reconstruction provided that silhouette data can be extracted with sufficient exactitude.

5.4.5 Robustness against Noisy Silhouette Data

In real-world applications, the working environment is typically complex making the silhouette extraction procedure, from all views, a very difficult task to achieve. Consequently, real-world silhouette data are often inaccurate. In the context of the piano pedagogy application, silhouette extraction is complex due to the high amount of shadows which can compromise the output of many statistical-based methods relying on an *a priori* model of the background [8]. In this particular application, shadows are introduced mainly because of the small distance separating the performer from the keyboard. Even with adequate compensation, many shadowed pixels remain misclassified as foreground. This effect is shown with the passage from color images, in Figure 5.16 to silhouette images, in Figure 5.17.

Fortunately, many of these imperfections are cancelled (eliminated) in the resulting voxel model. Indeed, with shape-from-silhouette, a group of misclassified foreground pixels will be carried into a 3D voxel only if all other views also agree on foreground occupancy for that voxel. This only occurs, in Figure 5.17, near the hands of the performer as a few voxels pertaining to the keyboard are incorrectly carried into the model. A more important problem is the presence of holes in silhouette data (foreground pixels misclassified as background). This situation occurs whenever the clothing of the performer is too similar to the background color for certain groups of pixels. In such case, the error is almost systematically carried into 3D, resulting in a few holes in the reconstructed model.

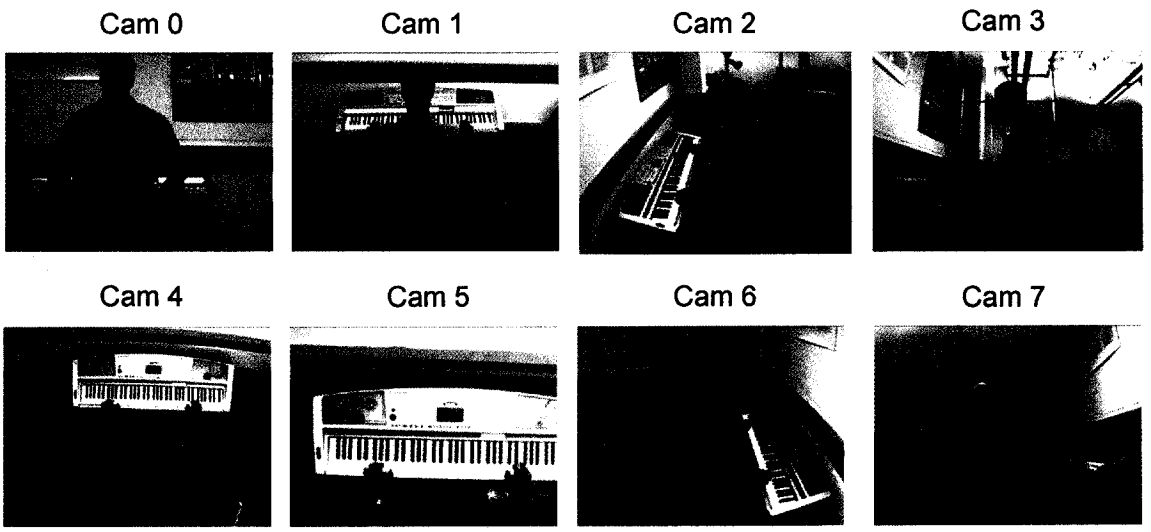


Figure 5.16. Original color videos from all 8 cameras

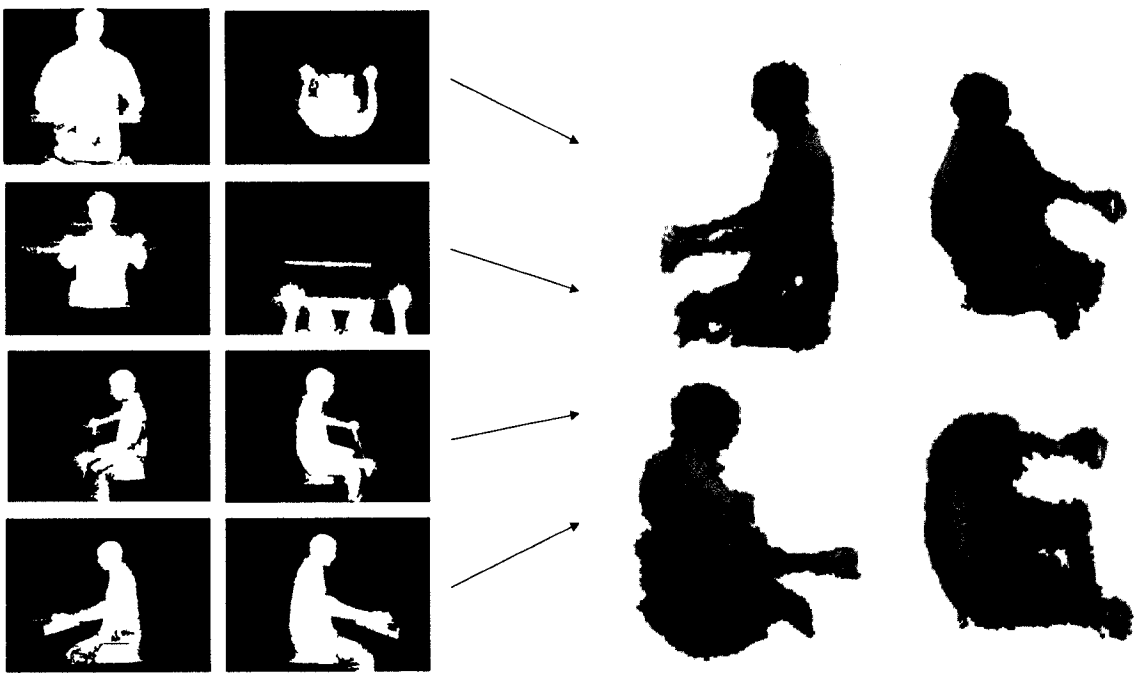


Figure 5.17. Inaccuracies in silhouette data due to the complexity inherent to real-world environments

5.4.6 Example of Application: Piano-Playing Performance Evaluation

The application of markerless motion capture in the context of piano-playing performance evaluation and prevention of injuries is particularly challenging because of the complexity of the working environment and because of the complex posture of pianists. On one side, shadows of the pianist over the keyboard make the silhouette extraction a difficult task which tempers the reliability of the volumetric reconstruction module. On the other hand, the standard pianist posture is a highly occlusive posture that admits a large concave region. Indeed, the two arms and the torso are self-occluding in the front/back views (i.e. Figure 5.18d). The left arm also occludes the right arm, or vice-versa, in side views (i.e. Figure 5.18c). In addition, a large concave region is present and is delimited as the region between the two arms and legs making difficult the separation of the two arms (i.e. Figure 5.18a, and Figure 5.18b).

Nevertheless, reliable volumetric reconstruction is obtained with different subjects and clothing. Figure 5.18 shows 4 (out of 8) views of a pianist with long hair and long sleeves. Figure 5.19 demonstrates that reliable reconstruction is achieved. In particular, both arms are reasonably well separated. Figure 5.20 shows reconstruction results for a second pianist with different clothing and hair style.

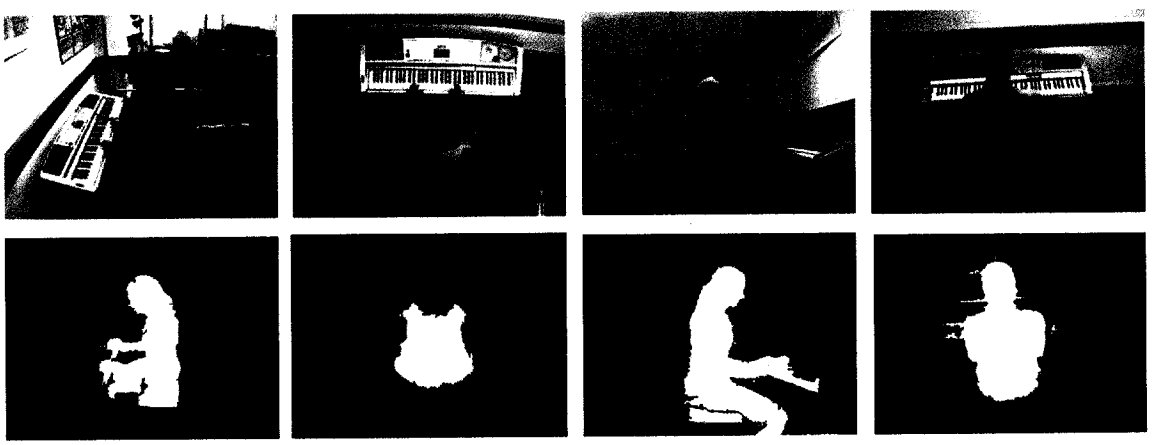


Figure 5.18. Four different color and silhouette views of a pianist

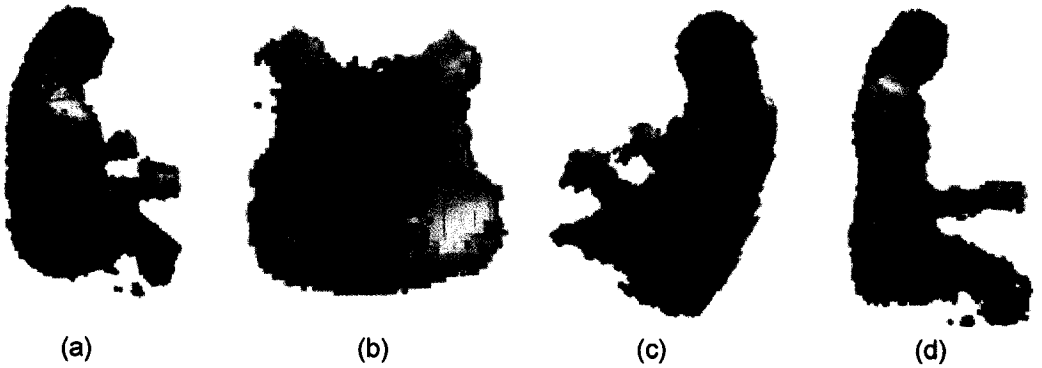


Figure 5.19. Four different views of a reconstructed pianist

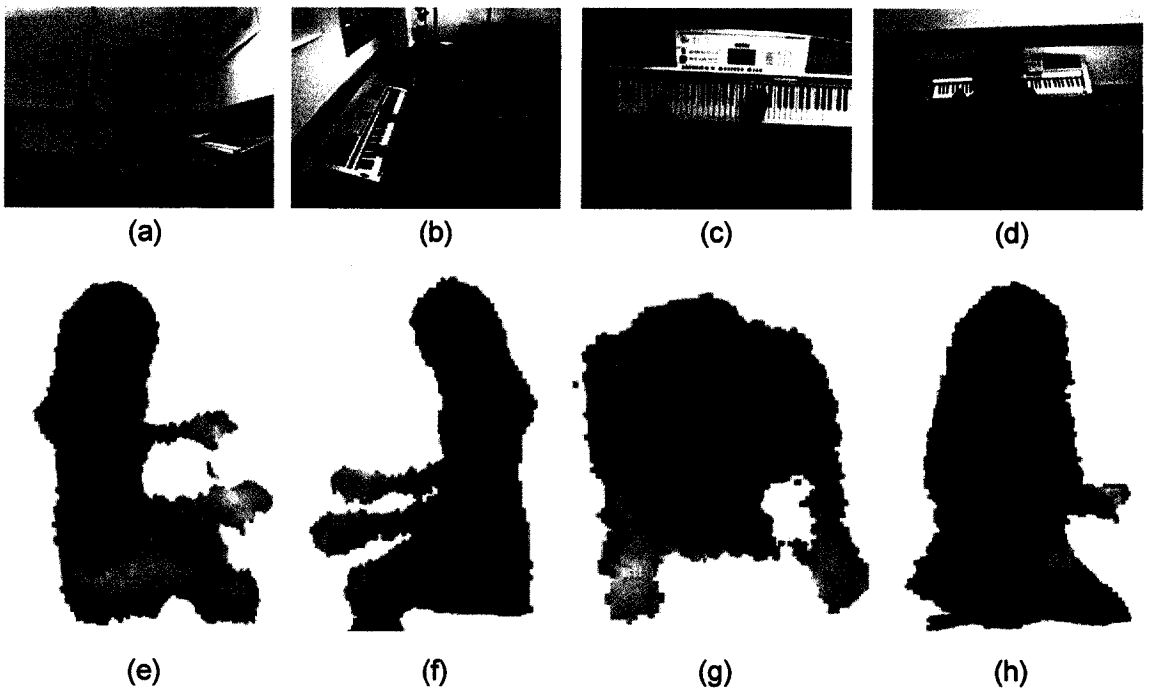


Figure 5.20. Four camera and reconstructed views of a second pianist performer

Figure 5.21 shows the reconstruction of multiple frames of video to show the robustness of the proposed algorithm over time. Since the amplitude of motion between consecutive frames of video is very small and difficult to perceive, displayed images admit a gap of at least 10 frames. The 3 frames displayed in the first row are each separated by 10 frames (0.33 second) and show a posture where the two hands and arms are well separated over the keyboard, and easier to reconstruct. The second and third rows show multiple frames of video where the hands are positioned closer to each other

making the posture more difficult to reconstruct because of an increase in self-occlusion and local concavities. Nevertheless, volumetric reconstruction is still achieved reliably.

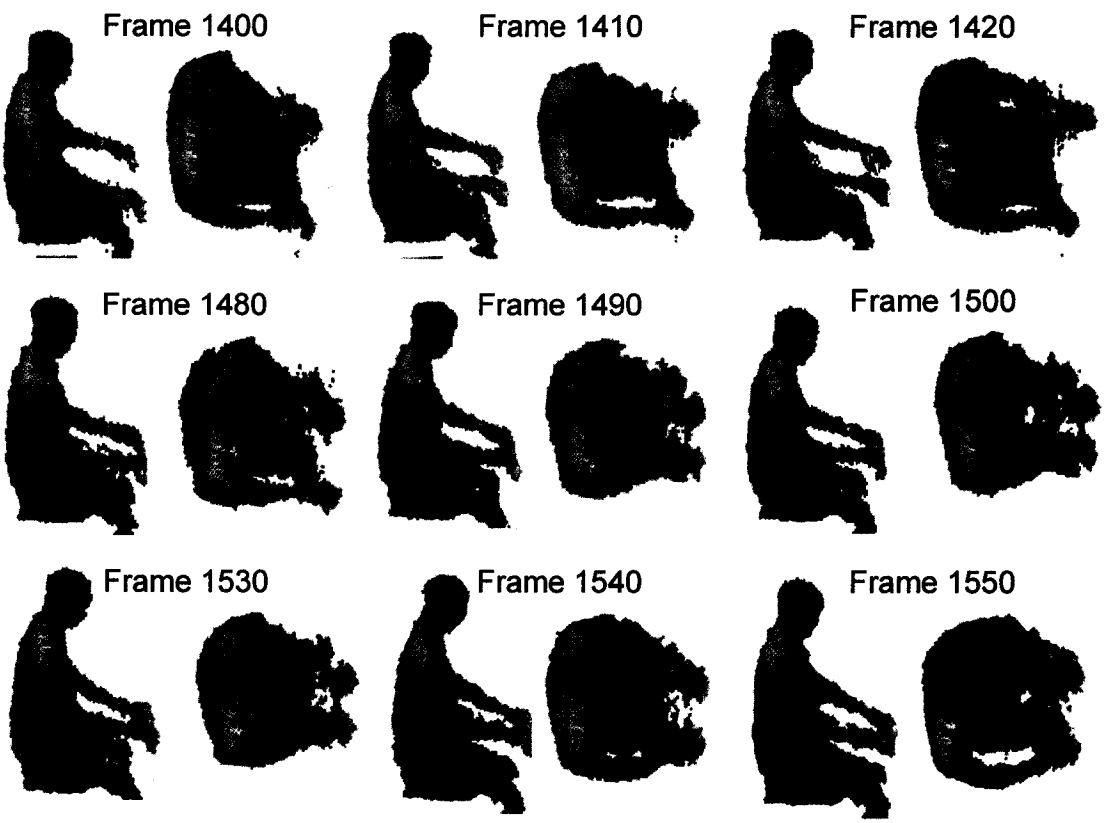


Figure 5.21. Shape-from-silhouette over time for the piano pedagogy application

5.4.7 Towards Human Posture Reconstruction in Complex Scenes

The previous section showed several reconstruction examples applied to the context of the piano-playing performance evaluation application. These results were reconstructed using a statistical silhouette extraction procedure [8] which is fast to execute but admits several drawbacks:

- An *a priori* model of the background is required and therefore, the background needs to remain static (only the targeted performer is allowed to move).
- The performer's clothing must be reasonably contrasting with respect to the background. We denoted experimental problems for the case of dark clothing

over a dark background. For example, performers with very dark hair are incorrectly segmented in the top view as the piano bench is dark as well.

- Shadows, on the keyboard, are omnipresent yielding to inaccurate reconstruction of the two hands. Many voxels pertaining to the keyboard, near the hands, are carried into the model.

For the purpose of the piano-playing performance evaluation application, we attempted to replace the use of a classical statistical segmentation scheme with a novel region-based segmentation algorithm [9, 10] developed concurrently to this project. As explained in section 5.1, region-based segmentation separates the content of an image into several groups of pixels based on color and texture similarities. Regions of interest are hand-selected and examples are shown in Figure 5.22a to Figure 5.22d. Then, selected regions of interests are tracked over time automatically in the multiple video streams. Here, the problem of shadowed regions on the keyboard is handled by the fact that these regions remain highly contrasting with respect to hand skin color. Shadowed keyboard pixels are thus clustered in a separate region than hand pixels. With an improved silhouette extraction module, the performer is potentially reconstructed with even more accuracy as demonstrated in Figure 5.22e to Figure 5.22h. In particular, in Figure 5.22f and Figure 5.22g, it is possible to notice that noisy voxels under the two hands, pertaining to the keyboard, have completely disappeared. On the counter part, a problem with region-based technique resides in the fact that large background regions may be clustered within the foreground object. This particular case occurs in the silhouette view of Figure 5.22b where a group of background pixels, located between the right arm and the right leg, is incorrectly segmented as "foreground" yielding to noisy voxels surrounding the right upper-arm, as shown in Figure 5.22g.

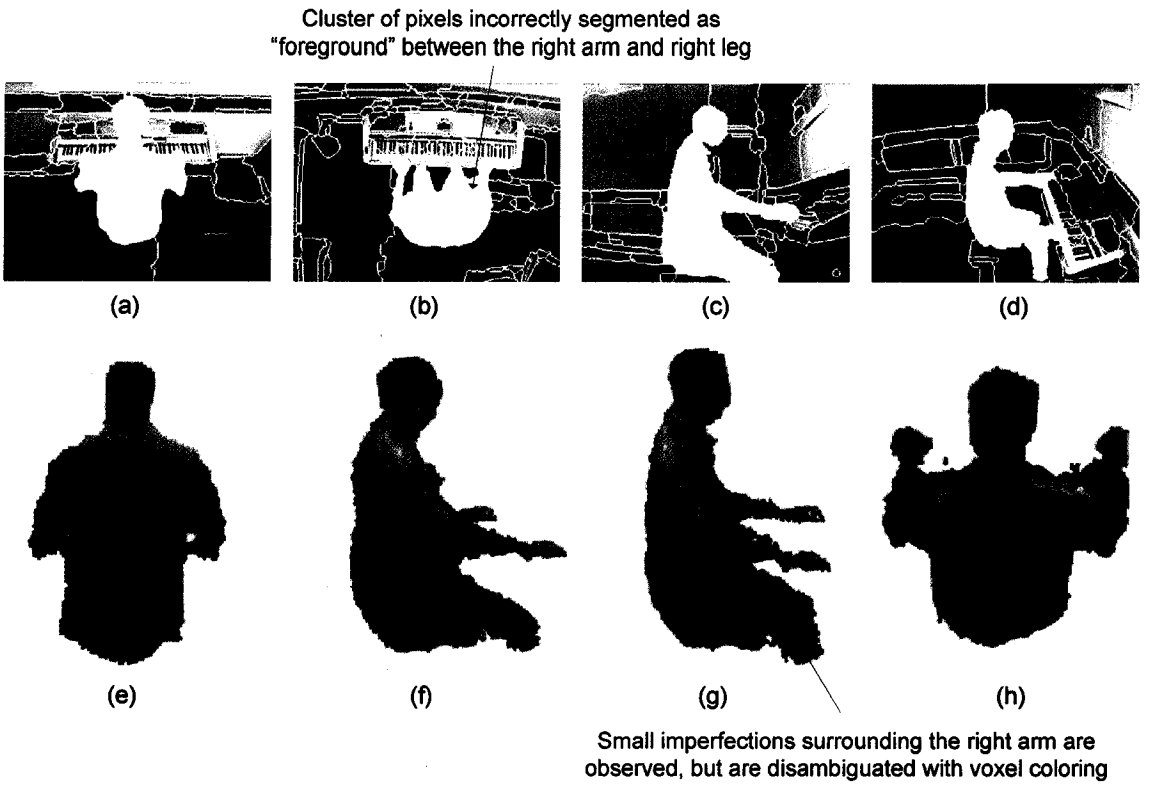


Figure 5.22. Shape-from-silhouette reconstruction using silhouette data obtained from a region-based segmentation method [9, 10]

Figure 5.23 shows another example of volumetric reconstruction with silhouette data generated from the region-based segmentation algorithm of [9, 10]. This second example also exhibits a more effective reconstruction of the performer's hands recalling that region-based segmentation is robust against the problem of shadowing. On the counter part, this segmentation scheme is highly sensible to the individual configuration of many threshold parameters that need to be manually tweaked and that depend on the actual content of the input videos. Overall, region-based segmentation has high potential to the field of motion capture in complex scenes. However, some work remains in order to increase the ease-of-use of this segmentation scheme.

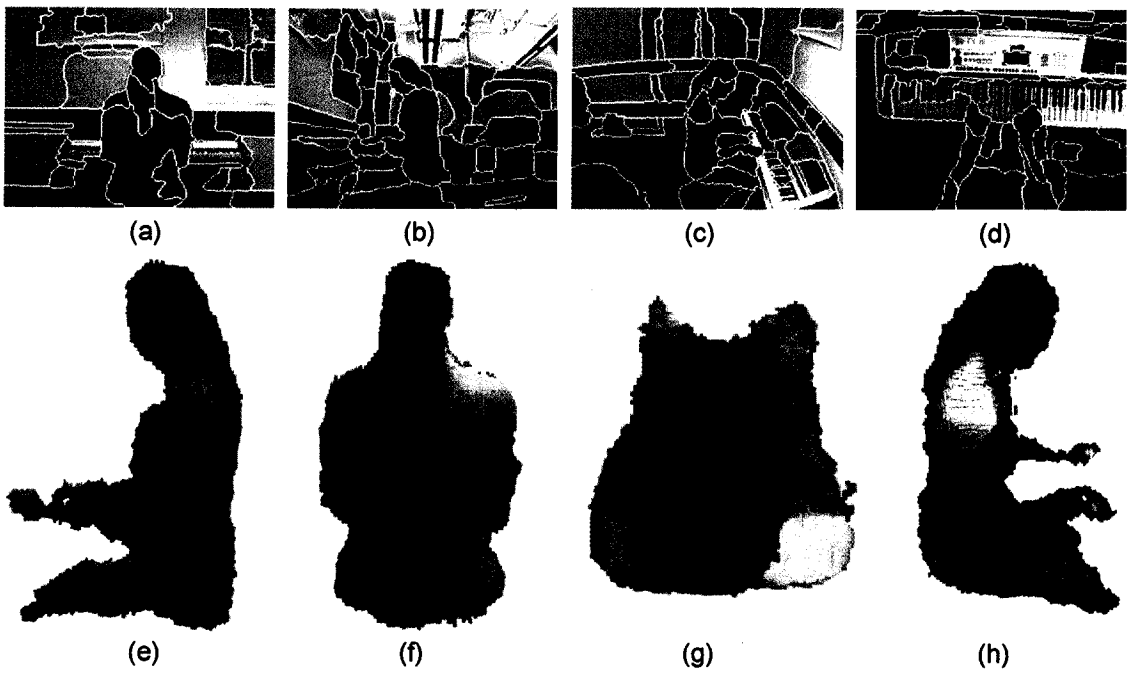


Figure 5.23. Another example of 3D reconstruction using region-based segmentation [9, 10]

5.5 Chapter Summary

This chapter presented a shape-from-silhouette reconstruction implementation which achieves adequate full body voxel reconstruction in relatively complex scenes. Difficult cases of self-occlusion and local concavities, proper to complex pianist postures, are handled in a robust manner. The enhanced voxel coloring scheme proposed in this work, using dense depth images, enables accurate voxel visibility tests and smooth, logic and meaningful color attribution for all surface voxels surrounding the reconstructed performer with the possible extension of propagating the color to interior voxels as well.

Reconstruction results presented in this chapter were proven to be of sufficient quality to provide reliable cues to subsequent modules dedicated to the analysis of human postures from a kinematic perspective. In particular, the implemented voxel coloring scheme surpasses, in perceptual accuracy, most other schemes proposed in the literature related to markerless motion capture systems. Emphasis has been put on the

necessity of accurate multi-camera calibration, hence validating, using a concrete real-world application, the calibration technique elaborated in Chapter 4. Finally, it has been demonstrated that the proposed reconstruction algorithm performs well even in the presence of noisy silhouette information and with complex, self-occluding, human body postures and, in particular, pianist postures. Furthermore, enhanced reconstruction results were obtained by using a novel region-base silhouette extraction scheme which eliminates the problem of shadow pixels.

Chapter 6. Conclusion and Future Work

This thesis focused on specific areas related to the problem of markerless human motion capture. In particular, this work addressed the issues of multi-camera system design, multi-camera calibration and volumetric reconstruction with coloring. This final chapter presents a summary of the research conducted with a special emphasis on its major contributions. The last section examines possible directions for future enhancements.

6.1 Summary

This thesis started off by introducing the reader to the problem of motion capture and, in particular, to the challenges related to gesture monitoring of pianist musicians using solely passive vision technologies. In Chapter 1, techniques for computerized motion capture were distinguished into two major categories. Marker-based solutions correspond to the state-of-the-art in motion capture, from a commercial perspective, because of their high robustness in accurately representing the kinematic posture for a wide variety of human body configurations. On the other hand, marker-based solutions are cumbersome from the fact that performers must wear multiple markers which can interfere with their natural motion. Therefore, extensive research has been performed, over the last decade, regarding markerless (vision-based) motion capture solutions. Unfortunately, lacks of robustness and generality in many areas of the whole solution currently prevents markerless systems from hitting real-world markets.

A review of current advances related to the field of motion capture has been presented in Chapter 2. Both marker-based and markerless systems have been reviewed. Special emphasis has been put on markerless systems because it relates to the objectives of this research work. From the analysis of existing markerless systems, we could identify several limitations. In particular, it has been raised that several simplifications

were introduced in early modules tempering the robustness of any subsequent modules and perhaps of the whole motion capture application. Hence, this thesis focused on increasing the overall robustness of specific aspects: multi-camera system design, multi-camera calibration, and volumetric reconstruction and coloring, in order to complement previous work of our group on video segmentation, and to achieve a fully integrated markerless human motion capture system.

Surprisingly, multi-camera system design is a topic which is discussed very briefly in most of the work reported in the literature. However sub-optimal multi-camera system design can yield to inaccuracies which are carried over throughout the rest of the application. Chapter 3 analyzed the design of a reconfigurable multi-camera system and covered the topics of hardware selection, architectural system design, and inter-camera synchronization. The outcome of this work resulted in the elaboration of a high-level software framework for generic multi-camera applications and in the construction of a fully operational synchronized multi-camera acquisition setup that was extensively used for experimental validation of all aspects of the research work.

The second major module developed within this thesis relates to the problem of multi-camera calibration. In Chapter 2, this problem was defined as the estimation of intrinsic camera parameters, to define the internal perspective projection behavior of camera and lens hardware, and the estimation of extrinsic camera parameters, which are concerned about the registration of one or many cameras to a common global reference frame. Two types of calibration techniques were evaluated: classical approaches, which are based on the use of complex calibration rigs, and self-calibration approaches which are based solely on image matches that correspond, in 3D, to a cloud of virtual calibration points. For the purpose of multi-camera calibration, classical approaches are not suitable because calibration rigs are cumbersome and they impose severe constraints on the relative positioning of the multiple cameras. In Chapter 4, a complete framework for generic multi-camera calibration has been presented. The paradigm of generating a cloud of virtual calibration points, by waving a simple marker, was utilized to achieve this task.

Finally, the third module developed throughout this project consisted of the implementation and refinement of a shape-from-silhouette volumetric reconstruction algorithm. In Chapter 2, it has been well established that voxel data, acquired from the 3D intersection of multiple silhouette images is a predominant cue for human kinematics estimation in full 3D space. In Chapter 5, a shape-from-silhouette voxel reconstruction algorithm has been proposed to estimate both the binary occupancy and the color component of voxels that subdivide a pre-defined working environment. Many results were presented and analyzed for the reconstruction of both generic human body postures as well as complex pianist postures.

6.2 Contributions

This thesis provided four major contributions to the field of markerless motion capture applications and they relate to the three activities discussed within this work.

1) Multi-camera system design and integration

Chapter 3 achieved a formal analysis about reconfigurable multi-camera system design which aimed at providing superior quality input data to subsequent modules within the whole markerless motion capture framework. Specialized high-quality camera instrumentation, optimized for motion capture and equipped with a global shutter, was used to acquire video data. All cameras were integrated with proper optics and were synchronized with very high accuracy surpassing other synchronization mechanisms proposed in other works related to motion capture [22, 24]. A software package was finally developed to facilitate the implementation of multi-camera applications and that abstracts low-level interaction with the camera hardware layer. The design of an optimal multi-camera system is further analyzed in Chapter 5 with an experimental evaluation of the minimal number of cameras to be integrated in a setup for realistic human motion capture of a single subject.

2) Multi-camera system calibration framework

The multi-camera calibration procedure proposed in Chapter 4 distinguishes itself from alternative implementations by its accuracy, ease-of-use, flexibility and repeatability. The accuracy of the proposed procedure was assessed for a variety of multi-camera networks. A final average reprojection error below ½ pixel was systematically obtained, thus leading to superior calibration in comparison to similar methods of Chen *et al.* [36] and Ihrke *et al.* [37]. The proposed method is easy to perform for non-expert users as it does not require any precise and cumbersome measurement in the 3D workspace. Furthermore, the use of a dual-marker calibration instrument allows for the extrinsic calibration and the estimation of the global scale factor to be performed in a single step. The method is flexible to calibrate a wide variety of multi-camera networks with variable number of cameras. The proposed enhancements, using a weighted graph analysis, managed to achieve very precise initial estimate of the extrinsic calibration which is refined through an efficient bundle adjustment implementation. The proposed method is easily repeatable and runs in a timely manner, in comparison with Svoboda *et al.*'s framework [38]. This is clearly beneficial since the system must be re-calibrated upon any modification on the positioning of one or more camera. Finally, the high precision of the proposed multi-camera calibration algorithm was further validated, in Chapter 5, using real shape-from-silhouette reconstruction examples.

3) 3D reconstruction with voxel coloring

The shape-from-silhouette reconstruction algorithm presented in Chapter 5 has proven its capability to provide representative 3D reconstruction of human subjects and especially pianist performers. The proposed reconstruction algorithm is robust against noisy silhouette data. Moreover the proposed voxel coloring algorithm provides sound texture mapping with subjectively superior accuracy in comparison to the schemes introduced by Caillette [22] and Kehl *et al.* [20, 21], while being deterministic (not

iterative). Many examples of reconstruction were shown, and especially within the context of piano-playing, which poses several challenges related to self-occlusion and local concavities. Also, many shadowing problems were identified, especially over the keyboard, making difficult the use of statistical segmentation methods. We resolved this shadowing problem by merging the proposed acquisition and reconstruction framework with an innovative region-based video segmentation technique [9, 10] recently developed by other members of our group.

4) The formalization of a solid framework for markerless motion capture

Combining an optimally designed and calibrated multi-camera acquisition system with innovative silhouette extraction techniques and a robust voxel reconstruction algorithm led to a solid and complete framework for full 3D shape modeling of the human body. This framework removes several constraints related to the performer and the working environment and can be successfully applied to real-world examples such as that of piano-playing performers. The end result is a complete reconfigurable system for markerless human motion capture capable of operating in complex environments found in realistic scenarios.

6.3 Future Work

This thesis presented the development and implementation of a specific set of modules related to markerless motion capture. In each of these modules, some extensions can be envisioned. The multi-computer system designed in Chapter 3 could be enhanced into a well-integrated distributed system with the implementation of a formal message-passing inter-computer communication protocol, thus allowing all video streams to be processed simultaneously upon availability without any manual transfer between computing nodes. The multi-camera calibration procedure presented in Chapter 4 could be enhanced to use natural scene features as calibration points rather than artificially created features obtained by waving a marker. This could allow the calibration to be

performed on-the-fly, thus decoupling calibration and video data, and it could even allow cameras to move during the recording of human motion to better track the performer. Unfortunately, the problem of finding a sufficient number of matches, due to very large baselines and orientation changes among the viewpoints is currently prohibiting this enhancement with the number of cameras available. The shape-from-silhouette reconstruction scheme of Chapter 5 has proven to be robust even against imperfect silhouette data. However, it would be interesting to investigate how 3D reconstruction data could be fed back to the silhouette extraction module, using depth as a supplementary cue to color and 2D texture information, to progressively improve silhouettes and, inevitably, 3D reconstruction data for subsequent frames of video. Finally, from a system-level perspective, the next logical step towards the deployment of our motion capture application would be to analyze voxel data in order to fetch higher level information about the human posture such as kinematics information. This is the purpose of subsequent activities, related to markerless motion capture (human pose estimation and human gesture analysis) and they remain open.

References

1. D. Russell, "Establishing a Biomechanical Basis for Injury Preventative Piano Pedagogy", *Recherche en Éducation Médicale*, no. 24, pp. 105-118, August 2006.

2. S. Yabukami, H. Kikuchi, M. Yamaguchi, K. I. Arai, K. Takahashi, A. Itagaki, N. Wako, "Motion Capture System of Magnetic Markers Using Three-Axial Magnetic Field Sensor", *IEEE Trans. on Magnetics*, vol. 36, no. 5, pp. 3646-3648, September 2000.

3. S. Hashi, M. Toyoda, S. Yabukami, K. Ishiyama, Y. Okazaki, K. I. Arai, "Wireless Magnetic Motion Capture System – Compensatory Tracking of Positional Error Caused by Mutual Inductance", *IEEE Trans. on Magnetics*, vol. 43, no. 6, pp. 2364-2366, June 2007.

4. Polhemus, *Liberty Latus*, Colchester, VT, 2005. *www.polhemus.com.*

5. Vicon Peak, *Vicon Motion Capture System*, Lake Forest, CA, 2005. *www.vicon.com.*

6. MetaMotion, *Motion Captor Optical Motion Capture System*, San Francisco, CA, 2006. *www.metamotion.com.*

7. PhaseSpace, *Impulse Motion Capture System*, San Leandro, CA, 2007. *www.phasespace.com.*

8. M. Côté, P. Payeur, G. Comeau, "Comparative Study of Adaptive Image Segmentation Techniques for Gesture Analysis in Unconstrained Environments", *Proc. of the IEEE International Workshop on Imaging Systems and Techniques*, pp. 28-33, Minori, Italy, 2006.

9. M. Côté, P. Payeur, G. Comeau, "Video Segmentation for Markerless Motion Capture in Unconstrained Environments", 3rd International Symposium on Visual Computing, Lake Tahoe, NV/CA, *LNCS*, vol. 4842, pp. 791-800, Springer-Verlag, Nov. 2007.

10. M. Côté, *Video Segmentation for Markerless Motion Capture in Unconstrained Environments*, M. A. Sc. Thesis, University of Ottawa, 2007.

11. Dartfish Video Software Solutions, *Dartfish ProSuite*, Fribourg, Switzerland, 2007. *www.dartfish.com.*

12. Mova, *Contour Reality Capture*, San Francisco, CA, 2007. *www.mova.com.*

13. Organic Motion, *OpenStage*, New York, NY, 2007. *www.organicmotion.com.*

14. Q. Cai, J. Aggarwal, "Tracking Human Motion using Multiple Cameras", *Proc. of the International Conference on Pattern Recognition*, vol. 3, pp. 68-72, Vienna, Austria, August 1996.

15. Q. Delamarre, O. Faugeras, "3D Articulated Models and Multi-View Tracking with Silhouettes", *Proc. of the IEEE International Conference on Computer Vision*, pp. 716-721, Corfu, Greece, September 1999.

16. I. Mikic, M. Trivedi, E. Hunter, P. Cosman, "Articulated Body Posture Estimation from Multi-Camera Voxel Data", *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 455-460, Kauai, HI, December 2001.

17. I. Mikic, M. Trivedi, E. Hunter, P. Cosman, "Human Body Model Acquisition and Tracking Using Voxel Data", *International Journal of Computer Vision*, vol. 53, no. 3, pp. 199-223, 2003.

18. G. Cheung, T. Kanade, J. Bouguet, M. Holler, "A Real Time System for Robust 3D Voxel Reconstruction of Human Motions", *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 714-720, Hilton Head Island, SC, June 2000.

19. G. Cheung, S. Baker, T. Kanade, "Shape-From-Silhouette of Articulated Objects and its Use for Human Body Kinematics Estimation and Motion Capture", *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 77-84, Madison, WI, June 2003.

20. R. Kehl, M. Bray, L. Van Gool, "Full Body Tracking from Multiple Views Using Stochastic Sampling", *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 129-136, San Diego, CA, June 2005.

21. R. Kehl, L. Van Gool, "Markerless Tracking of Complex Human Motions from Multiple Views", *Computer Vision and Image Understanding*, vol. 104, pp. 190-209, 2006.

22. F. Caillette, *Real-Time Markerless 3-D Human Body Tracking*, Ph.D Thesis, University of Manchester, 2006.

23. A. Sundaresan, *Towards Markerless Motion Capture: Model Estimation, Initialization and Tracking*, Ph.D Thesis, University of Maryland, 2007.

24. P. Doubek, T. Svoboda, L. Van Gool, "Monkeys – a Software Architecture for ViRoom – Low-Cost Multicamera System", Third International Conference on Computer Vision Systems, Springer-Verlag, *LNCS* 2626, pp. 386-395, April 2003.

25. H. Nanda, R. Cutler, "Practical Calibrations for a Real-Time Digital Omnidirectional Camera", *Technical Sketches, Computer Vision and Pattern Recognition*, 4 pages, December 2001.

26. R.Y.Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", *IEEE Journal of Robotics and Automation*, vol. 3, no. 4, pp. 323-344, August 1987.

27. J. Heikkilä, O. Silven, "A Four-Step Camera Calibration Procedure with Implicit Image Correction", *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1106-1112, San Juan, Puerto Rico, June 1997.

28. Intel, *Open Computer Vision Library*, release 1.0, November 2006, *http://sourceforge.net/projects/opencvlibrary.*

29. J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab", April 2004, *http://vision.caltech.edu/bouguetj/calib_doc.*

30. Z. Zhang, "A Flexible New Technique for Camera Calibration", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330-1334, November 2000.

31. M. Fiala, C. Shu, "Fully Automatic Camera Calibration Using Self-Identifying Calibration Targets", Technical Report, National Research Council of Canada, 26 pages, November 2005.

32. S. Drouin, R. Poulin, P. Hébert, M. Parizeau, "Monitoring Flexible Calibration of a Wide Area System of Synchronized Cameras", *Proc. of the 16th International Conference on Vision Interface*, pp. 49-56, Halifax, NS, June 2003.

33. R. Hartley, A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, UK, 2000.

34. O. Faugeras, Q-T. Luong, *The Geometry of Multiple Images*, The MIT Press, 2001.

35. P. Rander, *A Multi-Camera Method for 3D Digitization of Dynamic, Real-World Events*, PhD Thesis, Robotics Institute, Carnegie Mellon University, May 1998.

36. X. Chen, J. Davis, P. Slusallek, "Wide Area Camera Calibration Using Virtual Calibration Objects", *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 520-527, Hilton Head Island, SC, June 2000.

37. I. Ihrke, L. Ahrenberg, M. Magnor, "External Camera Calibration for Synchronized Multi-video Systems", *Proc. of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'04)*, pp. 537-544, Plzen, Czech Rep., February 2004.

38. T. Svoboda, D. Martinec, T. Pajdla, "A convenient Multi-Camera Self-Calibration for Virtual Environments", *Presence: Teleoperators and Virtual Environments*, MIT Press, vol. 14, no. 4, August 2005.

39. B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, "Bundle Adjustment: A Modern Synthesis", Vision Algorithms: Theory and Practice, *LNCS*, vol. 1883, pp. 298-372, Springer-Verlag, 2000.

40. A. Laurentini, "The Visual Hull Concept for Silhouette-Based Image Understanding", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, February 1994.

41. A. Laurentini, "Surface reconstruction accuracy for active volume intersection", *Pattern Recognition Letters*, vol. 17, issue 12, pp. 1285-1292, 1996.

42. C. R. Dyer, "Volumetric Scene Reconstruction From Multiple Views", in L.S. Davis, *Foundation of Image Understanding*, Boston, Kluwer, pp. 469-489, 2001.

43. XVID Solutions, *XVID Codec*, Hof, Germany, release 1.1.2, November 2006. *www.xvid.org*.

44. S. Bériault, P.Payeur, G. Comeau, "Flexible Multi-Camera Network Calibration for Human Gesture Monitoring", *Proc. of the IEEE Workshop on Robotic and Sensors Environments*, pp. 125-130, Ottawa, ON, October 2007.

45. M. Fischler, R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Communications of the ACM*, vol. 24, no. 6, June 1981.

46. R.I. Hartley, P.Sturm, "Triangulation", *Computer Vision and Image Understanding*, vol. 68, no. 2, pp.146-157, November 1997.

47. S. Skiena, M. Revilla, *Programming Challenges: The Programming Contest Training Manual*, Springer-Verlag, New York 2003.

48. M. Lourakis, A. Argyros, *A Generic Sparse Bundle Adjustment C/C++ Package Based on the Levenberg Marquardt Algorithm*, version 1.3, 2006. *http://www.ics.forth.gr/~lourakis/sba*.

49. M. Lourakis, A. Argyros, "The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm", Institute of Computer Science, Forth, Technical Report 340, August 2004.

50. S. Bériault, M. Côté, P Payeur, "Volumetric Modeling with Multiple Cameras for Markerless Motion Capture in Complex Scenes", submitted to *IEEE International Conference on Instrumentation and Measurement Technology*, 2008.

51. Point Grey Research, "Digital Camera Register Reference", Technical Document, version 2.1, 111 pages, Revision of August 2007.

52. E. Trucco, A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, New Jersey, US, 1998.

Appendix A. Firewire 1394b Bandwidth Allocation

Table A.1. IEEE1394b bandwidth consumption for Flea2 color cameras operating at various video formats in free-run video mode

Resolution	Frame rate	Pixel format (Number of bits per pixel)	Number of pixels per packet	Number of Bytes per packet	Maximal number of cameras per bus
320x240	15 fps	YUV422 (16 bpp)	160 pixels	320 Bpp	25[1,2]
320x240	30 fps	YUV 422 (16 bpp)	320 pixels	640 Bpp	12[2]
320x240	60 fps	YUV 422 (16 bpp)	640 pixels	1280 Bpp	6[2]
640x480	15 fps	YUV 411 (12 bpp)	640 pixels	960 Bpp	8[2]
640x480	15 fps	YUV422 (16 bpp)	640 pixels	1280 Bpp	6[2]
640x480	15 fps	RGB24 (24 bpp)	640 pixels	1920 Bpp	4
640x480	30 fps	YUV 411 (12 bpp)	1280 pixels	1920 Bpp	4
640x480	30 fps	YUV422 (16 bpp)	1280 pixels	2460 Bpp	3
640x480	30 fps	RGB24 (24 bpp)	1280 pixels	3840 Bpp	2
640x480	60 fps	YUV411 (12 bpp)	2560 pixels	3840 Bpp	2

[1] The maximal number of simultaneous isochronous channel is 16.
[2] Current 1394b chipset only supports up to 4 simultaneous DMA channels. Some 1394a chipset will support up to 8 simultaneous DMA channels.

Table A.1 shows the bandwidth allocation for various video formats in free-run (isochronous) video mode. In free-run video mode, video data is transmitted from the camera to the Firewire bus using a pre-allocated guaranteed bandwidth. The bandwidth requirement for each camera is determined by the pixel format (number of bit per pixel) and the number of pixels per packet. The total packet size is calculated using these two values and is expressed in terms of Byte per packet. The maximal amount of video data per isochronous packet for the IEEE1394b bus is 8192 Bytes[1]. The maximal data packet

[1] It should be noted that there is 8000 ISO cycles per second and thus, isochronous video transmission only consumes approximately 500Mb/s worth of the total bandwidth of the IEEE1394b bus (800Mb/s). The extra bandwidth is reserved, in part, for asynchronous communication (camera control, etc).

size is used to determine the theoretical number of cameras that can be connected to an IEEE1394b bus.

An example of bandwidth consumption computations for the 640x480@30fps (YUV411) video format is shown. The required number of pixels per packet is provided by the manufacturer [51] for each video format. For this particular video mode, 1280 pixels are transferred per packet. The size of a pixel, in the YUV411 pixel format, is 12 bit. Thus, the total number of Bytes per packet (Bpp) is calculated such that:

$$\frac{1280 pixels}{packet} x \frac{12 bit}{pixel} x \frac{1 Byte}{8 bit} = \frac{1920 Byte}{packet} = 1920 Bpp \qquad (A.1)$$

The maximal packet size is then divided by the number of Bytes per packet for one camera. This determines the theoretical number of cameras that can be connected to a single IEEE1394b bus in the selected video format:

$$\text{MaxCameras} = \left\lfloor \frac{8192 Bpp}{1920 Bpp} \right\rfloor = \lfloor 4.26 \rfloor = 4 \text{ cameras} \qquad (A.2)$$

It should be noted that other restrictions need to be considered in determining the effective maximal number of cameras that can be connected to an IEEE1394b bus:

- The maximal number of isochronous channels (16)
- The maximal number of simultaneous DMA channels (typically 4 to 8)
- Saturation on the PCI bus (controlled by the operating system)
- The size of FIFO buffers (on 1394 cards) that are used to recover for PCI saturation issues

Appendix B. Triangulation of Image Points in the 3D Space

The 2D image projection of any known 3D world point can be computed directly using the camera calibration parameters according to the equations presented earlier in section 2.4.1. However, the loss of information in the passage from 3D to 2D-perspective disables the reverse operation of computing the position of a 3D world point from its 2D projection. Indeed, Figure B.1 shows the existence of an infinite number of 3D points (a ray of 3D points) that project onto the same 2D pixel position.

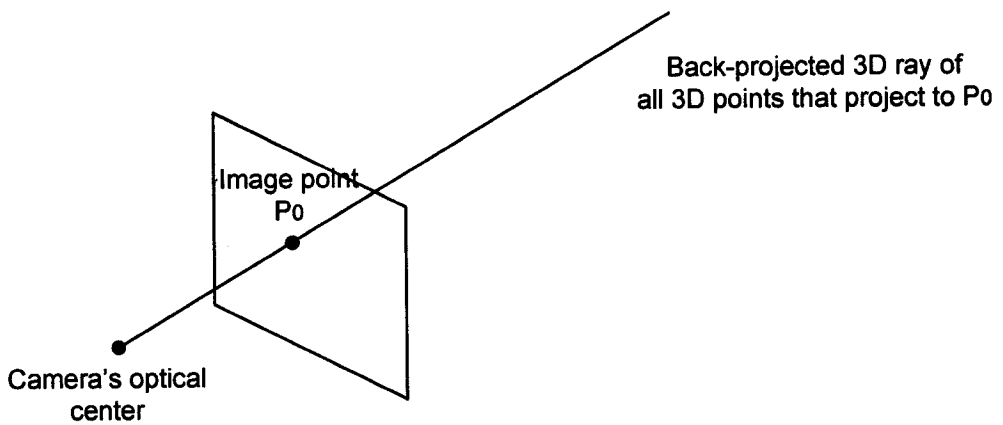


Figure B.1. Back-projection of an image point in 3D space

In order to reconstruct the world position of a 3D point, its corresponding pixel projection is required in at least two image views. This process is referred as triangulation. Figure B.2 shows two distinct image views where P_0 and P_1 are the image coordinates of a same 3D feature point (P_{3D}). If the two views are perfectly calibrated, both P_0 and P_1 can be back-projected and intersected in the 3D space to the exact position corresponding to P_{3D}. Unfortunately, in real-world applications, cameras are never perfectly calibrated and thus, back-projecting multiple 3D vectors will never result in an exact intersection. Taking this observation into account, several methods were proposed to approximate the intersection of two or more 3D vectors [33, 46, 52].

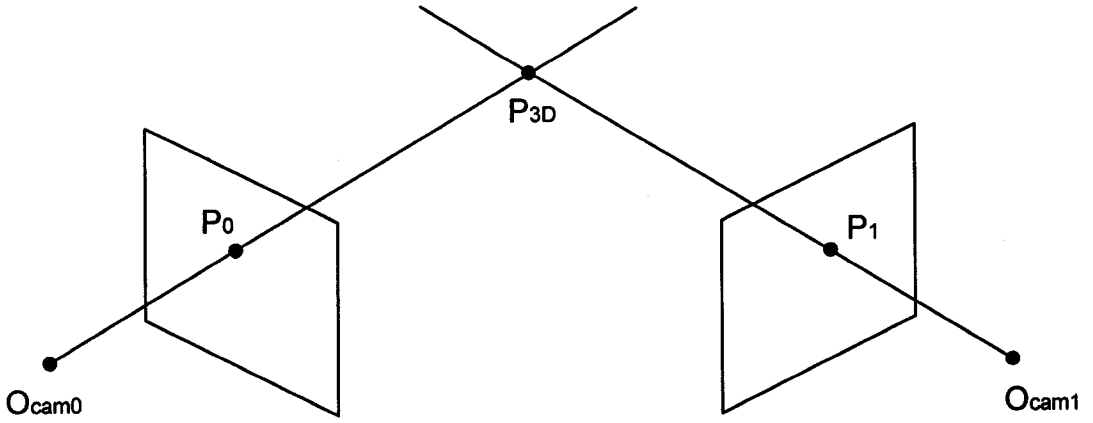


Figure B.2. Naïve triangulation of 2 image points in 3D space

In this work, triangulation is used in several occasions within the calibration procedure presented in Chapter 4. Two different methods were implemented and used: middle-point [46, 52] and linear least-square (Linear-LS) [46] triangulation. Mathematical derivations for those two methods are provided for completeness.

Middle-Point Triangulation

Middle-point triangulation is concerned about computing the segment of intersection between two 3D vectors, as shown in Figure B.3, rather than the exact point of intersection. This segment of intersection is perpendicular to both 3D vectors. The reconstructed point P_{3D} is estimated as the half-distance of this segment of intersection.

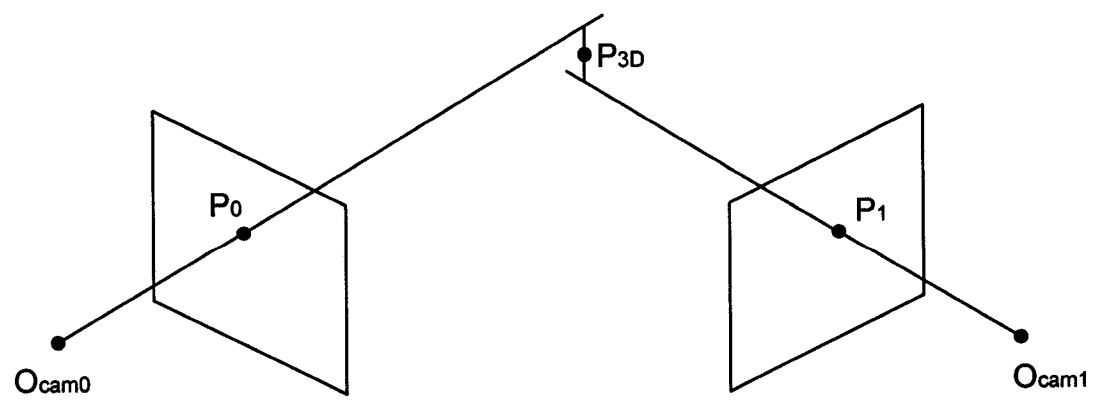


Figure B.3. Triangulation of 2 image points using the middle-point algorithm

Middle-point triangulation is discouraged because it is neither affine nor projective invariant [46]. This is however not a concerned because Euclidean reconstruction is used for multi-camera calibration. Hartley denoted that, even for Euclidean reconstruction, other linear methods [46] surpass middle-point triangulation. Nevertheless, middle-point triangulation is used in specific areas of our multi-camera calibration procedure. In particular, it is used in section 4.3.3 to disambiguate the four-fold solution obtained from fundamental matrix decomposition. For this particular application, the magnitude of the segment of intersection's half-distance is used to quantify the triangulation's precision and, thus, to disambiguate special cases of camera pose where two out of four solutions yield to a triangulated point simultaneously in front of both cameras. Middle point triangulation is also used in section 4.3.4 to build the graph of cameras with consistent pair-wise scaling. This particular use of middle-point triangulation deals with generalized 3D vector intersection rather than the intersection of back-projected image points. The logic, however, remains the same.

The process of middle point triangulation is described as followed:

1) The inverse of the perspective equation (2.2), presented in section 2.4.1, is applied to back-project $P_0=(x_0,y_0,1)^T$ and $P_1=(x_1,y_1,1)^T$ in 3D space. In equation (B.1), K_0 and K_1 are the intrinsic matrices of both cameras. P_0 and P_1 are the image projections of P_{3D} with adequate radial and tangential lens distortion compensation. P_{cam0} and P_{cam1} are the back-projected 3D points, in both views, and they are known up to a depth ambiguity.

$$P_{cam0} = K_0^{-1} P_0$$
$$P_{cam1} = K_1^{-1} P_1 \qquad \text{(B.1)}$$

2) Then P_{cam0} and P_{cam1} are expressed with respect to the global reference frame using the extrinsic camera parameters of both cameras (R_0, T_0) and (R_1, T_1).

$$P_{world0} = R_0{}^T P_{cam0} - R_0{}^T T_0$$
$$P_{world1} = R_1{}^T P_{cam1} - R_1{}^T T_1 \quad \text{(B.2)}$$

3) With the knowledge of P_{world0} and P_{world1}, the two 3D vectors to be intersected (v_0 and v_1) are computed as well as the translation between the two cameras (t_{01}). In equation (B.3), O_{cam0} and O_{cam1} are the optical centers of the two cameras and they are expressed with respect to the world reference frame. Thus, $O_{cam0} = -R_0{}^T T_0$ and $O_{cam1} = -R_1{}^T T_1$.

$$v_0 = P_{world0} - O_{cam0}$$
$$v_1 = P_{world1} - O_{cam1} \quad \text{(B.3)}$$
$$t_{01} = O_{cam1} - O_{cam0}$$

4) With the knowledge of v_0, v_1 and t_{01}, the problem of middle-point triangulation consists in solving a linear system of equations [52] as follows:

$$a.v_0 - b.v_1 + c.(v_0 \times v_1) = t_{01} \quad \text{(B.4)}$$

5) In equation (B.4), a, b and c are found such that:

$$a = \frac{\det([t_{01} \quad -v_1 \quad (v_0 \times v_1)])}{d}$$
$$b = \frac{\det([v_0 \quad t_{01} \quad (v_0 \times v_1)])}{d} \quad \text{(B.5)}$$
$$c = \frac{\det([v_0 \quad -v_1 \quad t_{01}])}{d}$$
$$\text{where } d = \det([v_0 \quad -v_1 \quad (v_0 \times v_1)])$$

6) Having solved the linear system of equations (B.4), the segment of intersection is described by two points, expressed with respect to the camera 0 frame.

$$P_{segmentA} = a.v_0$$
$$P_{segmentB} = b.v_1 + t_{01} \quad \text{(B.6)}$$

7) The actual reconstructed 3D point corresponds to the middle point between $P_{segmentA}$ and $P_{segmentB}$. The magnitude of this half-distance is also calculated and serves as a metric to quantify the triangulation's precision.

$$P_{3d}\big|_{cam0} = \frac{P_{segmentB} - P_{segmentA}}{2}$$
$$\text{HalfDistance} = \text{EuclideanDistance}(P_{3D}\big|_{cam0}, P_{segmentA}) \tag{B.7}$$

8) In order to express P_{3D} in terms of world coordinates, rather than with respect to the camera 0 frame, a final transformation is required:

$$P_{3D} = P_{3d}\big|_{cam0} - R_0{}^T T_0 \tag{B.8}$$

Linear-LS Triangulation

Besides middle-point triangulation, the use of Linear Least-Square (Linear-LS) triangulation is very common even though it remains less accurate than iterative methods [46]. In our multi-camera calibration procedure, Linear-LS triangulation is used, in section 4.3.6, to reconstruct the cloud of virtual calibration points (in 3D) which becomes an input to the bundle adjustment. The use of a linear over an iterative method is preferred, for speed of computation. Also, triangulated calibration points do not need to be iteratively optimized, because this task is already performed by the bundle adjustment. Furthermore, at this point, the camera calibration parameters, used for triangulation, are imprecise anyway. Linear-LS is also preferred over middle-point triangulation for this specific task because it can be easily generalized to the case of multi-camera triangulation (more than 2 views), as shown in Figure B.4.

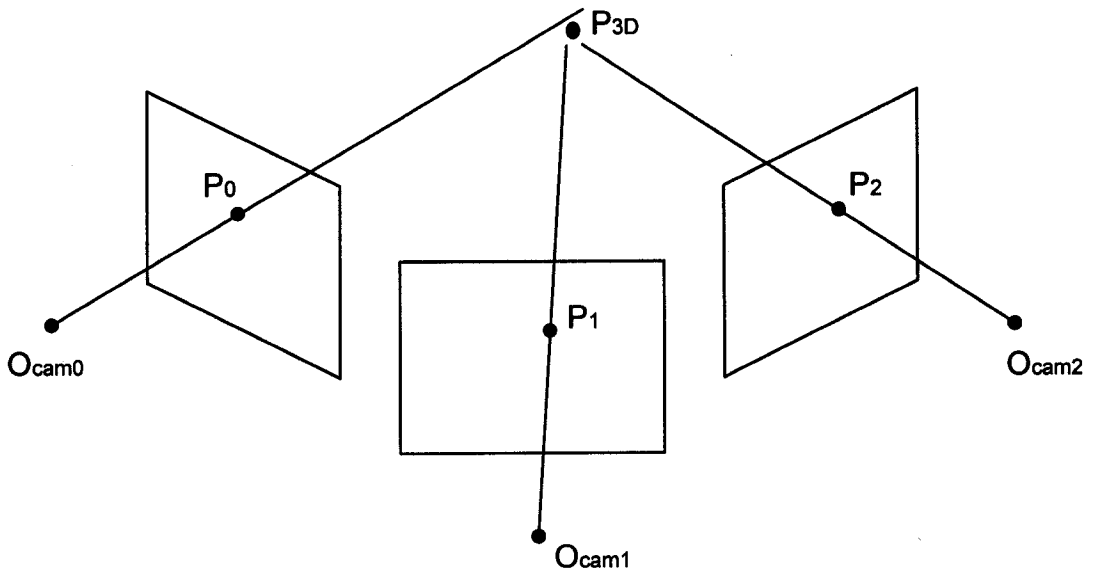


Figure B.4. Triangulation of multiple image points using the linear-LS algorithm

Steps to perform Linear-LS triangulation of N image matches ($N \geq 2$) are given as follows:

1) A 3x4 transformation matrix (C_i) for each camera is first computed, from the intrinsic matrix (K_i) and the extrinsic parameters ($R_i \mid T_i$) such that:

$$C_i = K_i \cdot [R_i \mid T_i] \tag{B.9}$$

2) The camera matrices allow the projection of a 3D point in each image view such that $P_{2D\_i} = C_i \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix}$. Hartley [33] solves the triangulation using an over-determined linear system of equations of the form $A \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} = \vec{0}$ where:

$$A = \begin{bmatrix} x_0 C_0(3,:)^T - C_0(1,:)^T \\ y_0 C_0(3,:)^T - C_0(2,:)^T \\ x_1 C_1(3,:)^T - C_1(1,:)^T \\ y_1 C_1(3,:)^T - C_1(2,:)^T \\ \cdots \\ \cdots \\ x_{N-1} C_{N-1}(3,:)^T - C_{N-1}(1,:)^T \\ y_{N-1} C_{N-1}(3,:)^T - C_{N-1}(2,:)^T \end{bmatrix} \tag{B.10}$$

In equation (B.10), $C_i(\alpha,:)^T$ represents the α^{th} row of C_i (the i^{th} camera matrix) expressed as a 1x4 row vector and (x_i, y_i) are the image coordinates in view i.

3) The least-square solution to this system of equations is obtained from singular value decomposition of A such that:

$$A = USV^T \tag{B.11}$$

4) The last column of V corresponds to P_{3D} and needs to be normalized to retrieve the form $[x\ y\ z\ 1]^T$, that is:

$$P_{3D} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} V(1,4)/V(4,4) \\ V(2,4)/V(4,4) \\ V(3,4)/V(4,4) \\ 1 \end{bmatrix} \tag{B.12}$$

Remarks:

The dimensions of the A matrix is 2Nx4 as there exist two linearly independent equations per camera. These are obtained from the standard projection camera equation:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = C_i \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} \tag{B.13}$$

Developing the matrix multiplication $C_i \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix}$, we obtain 3 equations:

$$\begin{aligned} x_1 &= C_i(1,:)^T \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} \Leftrightarrow x_1 - C_i(1,:)^T \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} = 0 \\ x_2 &= C_i(2,:)^T \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} \Leftrightarrow x_2 - C_i(2,:)^T \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} = 0 \\ x_3 &= C_i(3,:)^T \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} \Leftrightarrow x_3 - C_i(3,:)^T \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} = 0 \end{aligned} \tag{B.14}$$

With the substitution of $x = x_1/x_3$ and $y = x_2/x_3$ (pixel coordinates), this system is reduced to two linearly independent equations such that:

$$\begin{aligned} xC_i(3,:)^T \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} - C_i(1,:)^T \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} = 0 \\ yC_i(3,:)^T \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} - C_i(2,:)^T \cdot \begin{bmatrix} P_{3D} \\ 1 \end{bmatrix} = 0 \end{aligned} \tag{B.15}$$

Finally, all $\begin{bmatrix} P_{3D} \\ 1 \end{bmatrix}$ terms are cancelled, thus retrieving the form presented earlier in the A matrix of equation (B.10):

$$\begin{aligned} xC_i(3,:)^T - C_i(1,:)^T = 0 \\ yC_i(3,:)^T - C_i(2,:)^T = 0 \end{aligned} \tag{B.16}$$

Appendix C. Average Reprojection Error Calculation

The average reprojection error is a useful metric to assess the calibration accuracy of a multi-camera system. It consists of performing, in cycle, the reconstruction and reprojection of a large database of image matches in order to determine the average displacement of the reprojected over the measured (original) image points in each view.

Figure C.1 shows an example where P_0, P_1 and P_2 correspond to one image match or, in other words, the image coordinates of P_{3D}, a 3D calibration point, in multiple views. These image points are triangulated using the Linear-LS method elaborated in Appendix B. If the multi-camera calibration is imprecise, the reconstructed position of P_{3D} will inevitably be imprecise as well. Consequently, P_{3D} will be reprojected back into each image view with an accumulated error caused by both, the inaccurate estimation of P_{3D} and the inaccuracy of calibration parameters used to reproject P_{3D}.

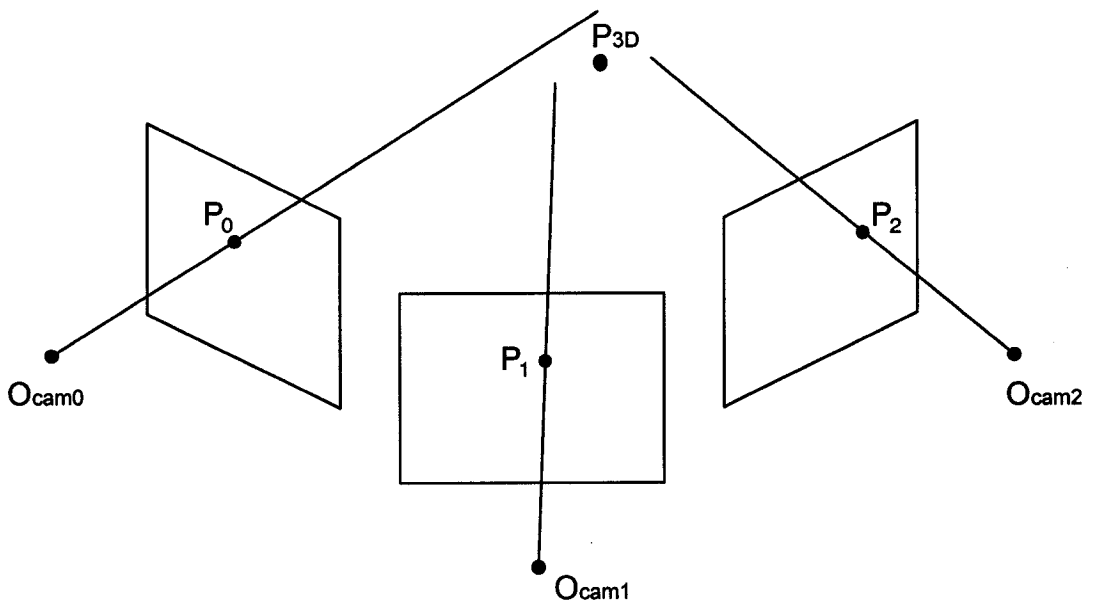


Figure C.1. A triangulated 3D point using imprecise camera calibration

Figure C.2 shows the projection of P_{3D} back into each image plane. $\hat{P}_0$, $\hat{P}_1$ and $\hat{P}_2$ are the reprojected points and P_0, P_1 and P_2 are the original (measured) points. The reprojection

error is calculated as the Euclidian distance between the original and the measured point, in each view separately:

$$error_i = \sqrt{\left(P_i(x) - \hat{P}_i(x)\right)^2 + \left(P_i(y) - \hat{P}_i(y)\right)^2} \qquad \text{(C.1)}$$

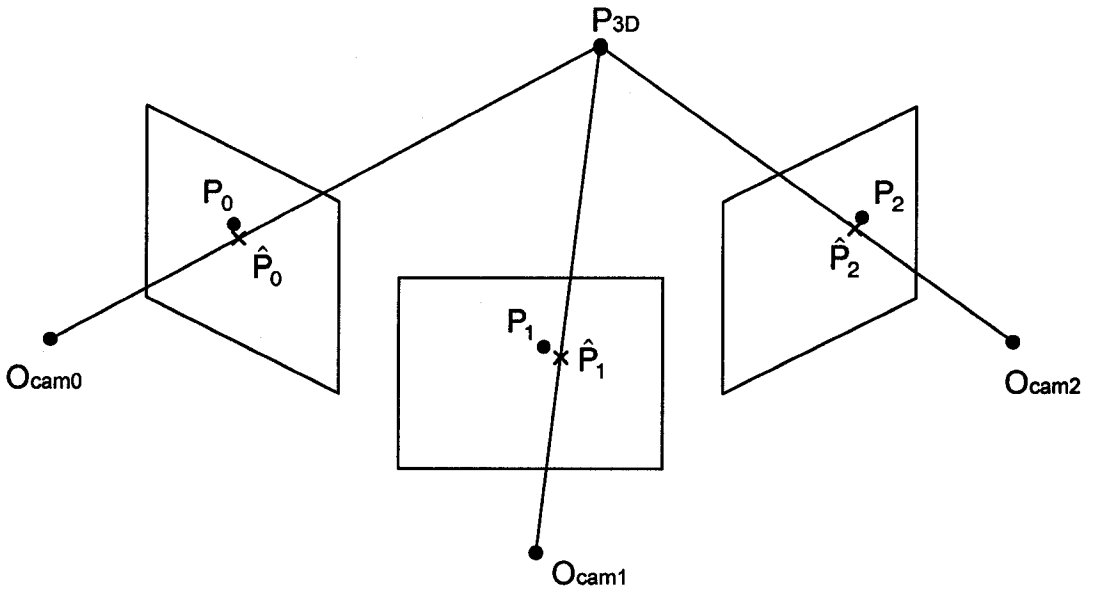


Figure C.2. A triangulated 3D point is reprojected back into each image plane with an error

Repeating this procedure over a large database of image matches provides a meaningful assessment of the calibration accuracy. When a multi-camera system is precisely calibrated, the calibration points are reconstructed, from triangulation, and reprojected back into each image plane with very little displacement. In Chapter 4, it has been shown that precise multi-camera calibration can yield to an average reprojection error below ½ pixel.