HERIOT-WATT UNIVERSITY

Disparity Map Completion for Trilinear-Tensor View Synthesis from Wide-Baseline Stereo

Špela Ivekovič

SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

ON COMPLETION OF RESEARCH IN THE

DEPARTMENT OF ELECTRICAL, ELECTRONIC AND COMPUTING ENGINEERING.

October 2008

Abstract

In this thesis, we present a model-based disparity-map completion method for high-quality novel-view synthesis from wide-baseline stereo, using trilinear-tensor transfer. In immersive videoconferencing environments, the wide-baseline stereo setups are used to gain maximum information about the scene with a minimum number of cameras in order to limit the amount of data that must be processed or transferred. To promote the immersive experience, the cameras are arranged around the screen with their viewpoint differing from that of a remote videoconference participant. The correct viewpoint is synthesised by means of novel-view synthesis.

When using trilinear tensor to synthesise novel views, dense disparity maps between the stereo pair of images are crucial to the quality of the synthesised view. However, as the scene in a wide-baseline stereo setup is imaged from two relatively different viewpoints, the correspondence search presents a difficult problem which does not result in dense disparity maps. In order to complete them, some form of post-processing is normally used. As a simple interpolation of the available disparities produces limited results, we look at using prior knowledge about the scene in the form of a human body model to complete the missing disparities for high-quality view synthesis in a videoconferencing environment.

We describe two alternative methods to disparity completion with a generic body model, the first working in 3-D space and the second in disparity space. We present an articulated subdivision surface model of a human body in 3-D and its equivalent in disparity space, address articulated pose estimation in 3-D and disparity space using particle swarm optimisation and present a quasi-interpolation method for fitting a subdivision surface body model to an unstructured, incomplete and noisy cloud of data points in 3-D and disparity space. Finally, we show the results of novel-view synthesis with disparity maps completed using the generic body model.

Acknowledgements

This thesis would not have been possible without the selfless contribution of many people.

First of all, thank you to Manuel Trucco for giving me the opportunity to work on this PhD and providing a prolific supervisory input of research ideas as well as contributing an extensive list of Latin quotations in the process.

A very big thanks to Yvan Petillot for his invaluable support on many levels, especially when things got tough.

Thank you to Craig Robertson for introducing me to the field of swarm intelligence and for his countless attempts to discuss with me his various pet projects, usually tangentially related to my research, which invariably broadened my research horizons.

Thanks to Gregor Miller for patiently listening to all my PhD rants and sharing his with me and thanks to all my friends who put up with my absence and busy schedule and still made an effort to stay in touch.

Thank you to the fantastic support staff at Heriot-Watt. Without people like them the academic departments would most likely be a complete chaos and nothing would ever get done.

A huge thanks to my parents, Melita and Željko, and my brother Aljaž, for believing in me and supporting my ambitions, even if so far away from home.

Last but not least, Daniel, thank you for always being there to receive the first rant when something went wrong and enjoy the first laugh of success. You've been an invaluable source of support and advice all the way through.

Contents

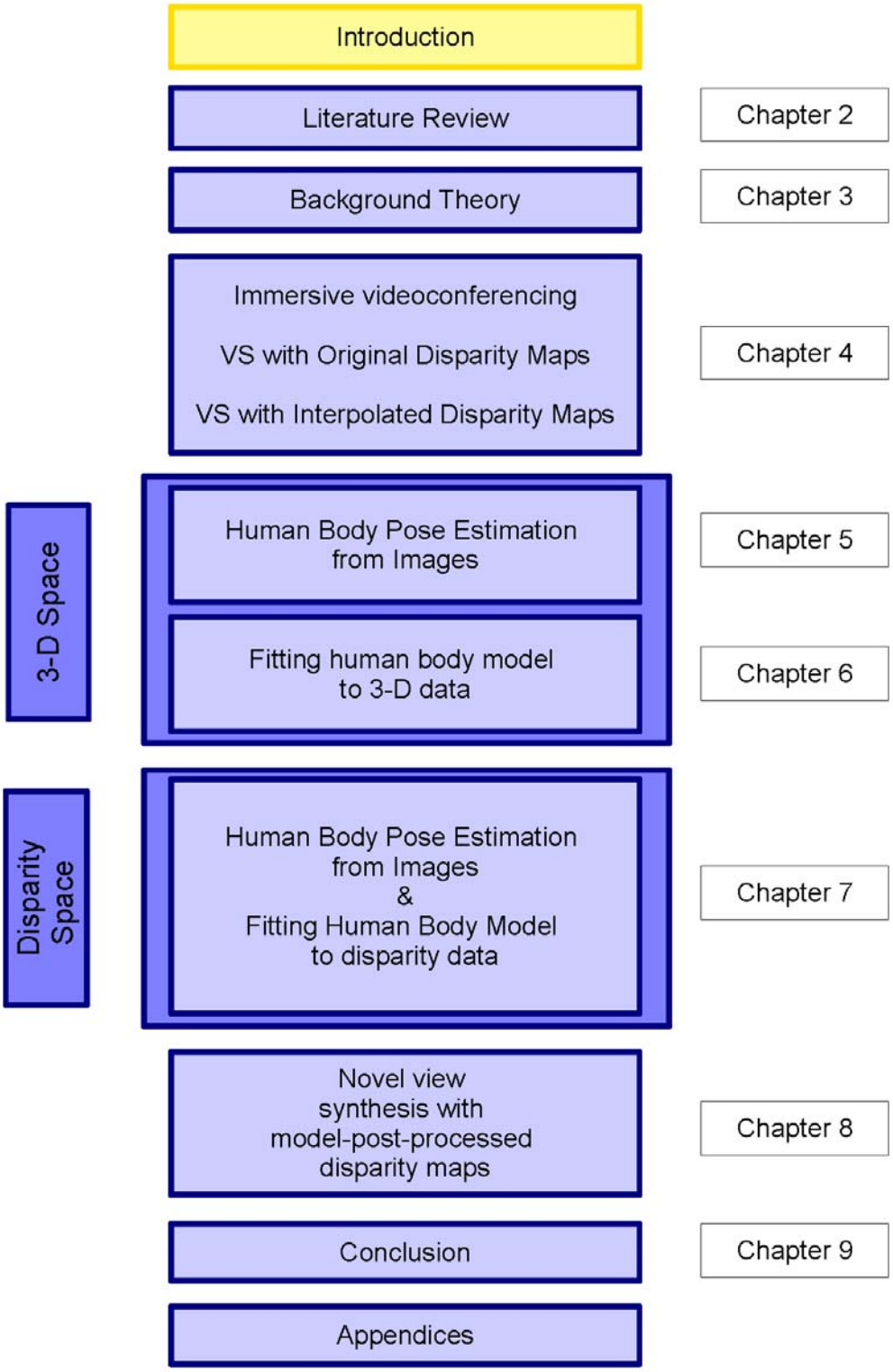


Figure 1: The thesis structure diagram. This diagram is shown at the beginning of every chapter. The highlighted part indicates the upcoming chapter and its place in the thesis structure.

Chapter 1

Introduction

1.1 Immersive Videoconferencing and IBR

Image-Based Rendering (IBR) addresses the problem of synthesising photorealistic novel views of the scene using the existing images and avoiding an explicit 3-D scene reconstruction. A formal definition of Image-Based Rendering given by Zhang and Chen [182] is as follows:

Given a continuous plenoptic function that describes a scene, image-based rendering is a process of two stages - sampling and rendering. In the sampling stage, samples are taken from the plenoptic function for representation and storage. In the rendering stage, the continuous plenoptic function is reconstructed with the captured samples.

Plenoptic function is a $7-D$ function which has been proposed by Adelson and Bergen [2]:

$$P = P(\theta, \phi, \lambda, t, V_x, V_y, V_z), \tag{1.1}$$

where an idealised eye is placed at every possible (V_x, V_y, V_z) location and the intensity of the light rays passing through the center of the pupil is recorded at every possible angle (θ, ϕ), for every wavelength λ, at every time t.

Due to the complexity of the plenoptic function, the amount of data required is so big that no one has been able to sample the full function into one representation. Research on

IBR tends to make reasonable assumptions to reduce the sample data size while keeping a satisfactory rendering quality.

On the highest level, two main strategies exist for data size reduction. The first one restrains the viewing space of the viewers and the second one introduces additional information such as, e.g., information about scene geometry [182].

In the view-restriction techniques, the area with the limited view is sampled densely enough to enable plenoptic function reconstruction by means of warping and linear interpolation. Examples of such methods are plenoptic modelling [108], lightfield [94] and lumigraph [65].

In the methods which use additional knowledge about the scene, the view is instead unlimited and the knowledge about scene geometry compensates for the small number of plenoptic function samples. Examples of such methods are view interpolation [34], view morphing [139] and view synthesis in tensor space [15].

Immersive videoconferencing using the concept of a shared virtual table environment, such as [136], is an example of an application where the knowledge about the scene geometry becomes necessary. In this application, the cameras are placed around a large screen and the goal is to synthesise and transmit a novel view simulating the conversation partner's viewpoint. This requires placing the camera in the centre of the screen, which is not possible unless the screen is semi-transparent [87].

For the mentioned immersive videoconferencing application, in an attempt to image as much of the scene as possible while also constraining the computation time and the amount of information eventually transmitted to other videoconferencing stations, a wide-baseline stereo setup offers a sensible solution [82]. This means that the plenoptic function can only be sampled sparsely and the knowledge about scene geometry becomes integral to the quality of the novel-view synthesis.

This knowledge can be introduced in the form of a disparity map describing the relative displacement of pixels in the original stereo pair of images. The existence of a disparity map enables the novel view synthesis from sparsely sampled plenoptic function under the condition that the two sample views are in full correspondence, *i.e.*, their disparity map is dense. This then allows for the use of IBR methods such as novel view synthesis in tensor space [15].

A dense disparity map is defined as a complete collection of pairs of corresponding pixels in the left and right image of the stereo pair. In a stereo pair of images, not every part of the scene is visible in both images simultaneously, which makes the correspondence information for such parts of the scene unavailable by definition. In a wide-baseline stereo setup, where the views of the scene are significantly different due to a large difference in camera rotation and translation, a dense disparity map is theoretically impossible to achieve.

In practice, a dense disparity map becomes an even more challenging problem. The quality of the disparity information obtained using correlation-based stereo correspondence algorithms invariably depends on the length of the baseline, scene depth variation, and sufficient texture information. The traditional correspondence search algorithms systematically compare image intensity information in a stereo pair of images to establish the corresponding regions. Factors such as changes in lighting conditions between the views, different camera colour response and textureless regions introduce ambiguities in the intensity-based correlation search methods. As a result, a number of additional constraints are required to guide the search.

One such constraint is the epipolar constraint which constrains the matching points to lie on conjugate epipolar lines [70] and significantly reduces the complexity of the search. Other constraints used in limiting the match ambiguity are uniqueness and continuity constraint [103], similarity constraint [66] and ordering constraint [17]. Although these constraints do prove useful in particular settings, they are not always sufficient when dealing with real scenes, as they can be violated [56].

In order to facilitate a high-quality and realistic novel-view synthesis, disparity maps derived from a wide-baseline stereo pair of images require some form of post-processing to complete the missing regions[13, 14].

The work presented in this thesis addresses the problem of incomplete disparity maps in a wide-baseline stereo setup and their use for novel-view synthesis. It focuses on disparity map completion methods for better quality novel-view synthesis and makes the following assumptions:

- The camera setup is low-cost.
- The cameras are arranged on a wide baseline.

and choices:

- The disparity maps are generated using a correlation-based pyramidal stereo correspondence search.
- The novel-view synthesis algorithm uses the trilinear-tensor point transfer method.

Our choices were motivated by the VIRTUE [136, 82] immersive videoconferencing system which highlighted the need for fast and dense wide-baseline disparity map computation for high-quality novel-view synthesis. In Chapter 9, we discuss the reasons for making these choices and how they fit in with the recent related research in the area of novel-view synthesis from wide-baseline stereo.

Although this work has been motivated by the challenges of the immersive videoconferencing setup, it explores the algorithmic solutions for a better quality novel-view synthesis and does not attempt to meet the real-time requirements of a videoconferencing application.

1.2 Thesis Structure

We begin with the review of the literature relevant to the research presented in this thesis in Chapter 2.

In Chapter 3, we present the background theory necessary for the understanding of the work presented in later chapters. We first describe the methods necessary to generate the disparity data and the reconstructed 3-D points, followed by the methods which allow us to model and manipulate a subdivision surface model of the upper human body. We conclude with the description of the trilinear-tensor geometric constraint and its use for novel-view synthesis.

In Chapter 4, we introduce the problem of novel-view synthesis in a wide-baseline stereo setup, such as immersive videoconferencing. We show the low quality results which can be achieved when synthesising novel views with the disparity maps obtained from the correlation-based stereo correspondence search without any post-processing. We then demonstrate the improvement in the view synthesis when the disparity maps are interpolated using different interpolation methods. We conclude by showing the potential of using the *a priori* knowledge in the form of a generic scene model to complete disparity maps and synthesise novel views.

In Chapter 5, we present an articulated subdivision surface 3-D upper body model, which we use as a generic model of the scene for disparity data completion. We then describe the algorithm for articulated body pose estimation from multi-view images based on the particle swarm optimisation and show the pose estimation results. We compare the performance of the particle swarm optimisation method with equivalent algorithms using simulated annealing and gradient descent.

In Chapter 6, we describe an algorithm for fitting a 3-D subdivision surface model to an unstructured cloud of noisy and incomplete data points in order to complete the missing regions without modelling the noise. We show results of the fit on accurate synthetic data

from a range scanner as well as 3-D reconstructed noisy stereo data originating from a correspondence search algorithm.

In Chapter 7, we present the concept of disparity space as an alternative to 3-D space. We introduce the disparity space articulated human body model, which allows us to perform disparity data completion directly in disparity space without having to use 3-D reconstruction. We then describe articulated body pose estimation in disparity space and disparity body model fitting to disparity data. We show the results of pose estimation and model fit for the disparity space model.

In Chapter 8, we present the results of trilinear-tensor novel-view synthesis from wide-baseline stereo images based on disparity maps which were completed in disparity space, using the generic disparity model of the upper human body. We compare the novel-view synthesis results using deformed and undeformed generic models to the ground truth views and discuss the need for model deformation in the presented disparity map completion framework.

We present our conclusions and talk about the present and future work in Chapter 9.

Every Chapter is preceded by a thesis structure diagram which highlights the topic presented in the corresponding chapter.

1.3 Summary of Contributions

1. Wide-baseline stereo disparity map completion with linear, bilinear and cubic spline interpolation algorithm for better view synthesis (Chapter 4)

    - S. Ivekovic, E. Trucco: Dense wide-baseline disparities from conventional stereo for immersive videoconferencing, In *Proceedings of ICPR 2004*, pages 921 - 924.

2. Articulated human body pose estimation from images with particle swarm optimiza-

tion (Chapter 5)

- S. Ivekovic, E. Trucco: Human Body Pose Estimation with PSO, In Proceedings of *IEEE Congress on Evolutionary Computation*, CEC 2006 (WCCI 2006), pages 1256 - 1263.
- S. Ivekovic, E. Trucco, Y.R.Petillot: Human Body Pose Estimation With Particle Swarm Optimisation, *MIT Evolutionary Computation Journal*, Special Issue on Evolutionary Computer Vision, Volume 16, Number 4, 2008.

3. Quasi-interpolation subdivision surface fitting method for noisy and incomplete stereo disparity data (Chapter 6)

   - S. Ivekovic, E. Trucco: Fitting Subdivision Surface Models to Noisy and Incomplete 3-D Data, In Proceedings of *Mirage 2007*, LNCS 4418, pages 542 - 554.

4. Articulated subdivision-surface disparity-space body model, articulated human body pose estimation in disparity space and model fitting to disparity data in disparity space (Chapter 7)

   - S. Ivekovic, E. Trucco: Articulated 3-D Modelling in a Wide-Baseline Disparity Space, In *Proceedings of CVMP 2007*.

5. Novel view synthesis using disparity maps completed with a generic human body model in disparity space (Chapter 8)

The following publications are indirectly related to the work in this thesis and are not included in the above list:

- C. Robertson, E. Trucco, S. Ivekovic: Dynamic body posture tracking using evolutionary optimisation, *Electronics Letters 41(25)*, 2005, pages 1370 - 1371.
- S. Ivekovic, A. Fusiello, E. Trucco: Chapter 6 - Fundamentals of Multiple-view Geometry in *3D VideoCommunication: Algorithms, concepts and real-time systems in human centred communication*, Wiley 2005

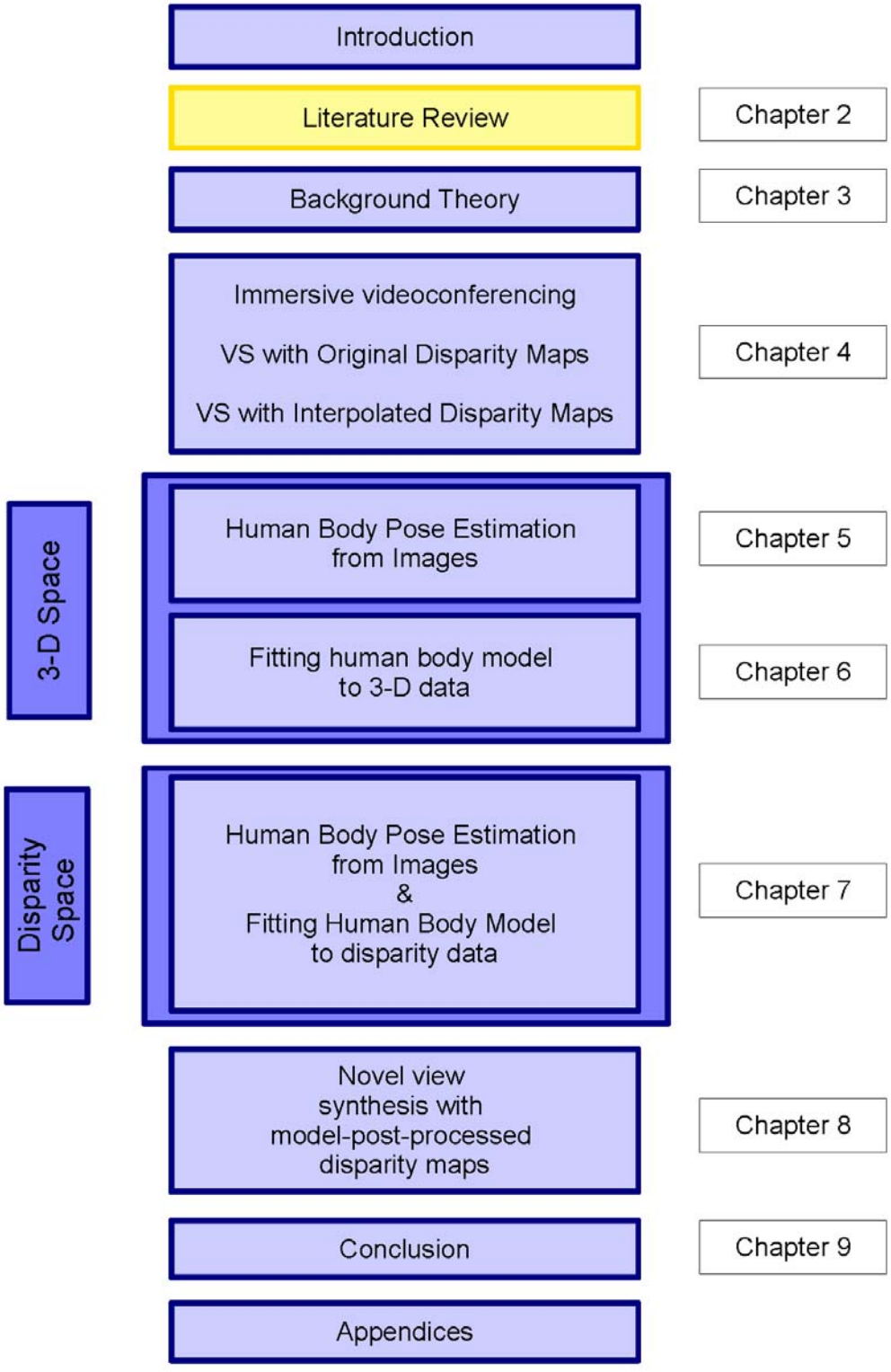


Figure 1.1: The thesis structure diagram.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, we present the literature review of the research areas relevant to this thesis. We organised the contents of the review to reflect the structure of the thesis, *i.e.*, the topics appear in the same order as the related chapters later on.

In Section 2.2, we begin with the review of immersive videoconferencing systems including the VIRTUE immersive videoconferencing system which motivated the work presented in this thesis. Section 2.3 looks at the pose estimation research and within that area the choice of body models, optimisation techniques and data sources. In Section 2.4 we talk about dense wide-baseline stereo correspondence approaches. We then look at techniques which fit a subdivision surface model to a cloud of data points in Section 2.5. Next, we review the research on disparity space in Section 2.6 and finally look at different approaches to the problem of novel view synthesis in Section 2.7.

2.2 Immersive Videoconferencing

The immersive videoconferencing is a communication concept involving a conferencing session, where geographically distributed conference participants get the impression of taking part in an ordinary conference session with everybody physically present in the same room, sharing the same conference table. The use of technology to establish a sense of

shared presence or shared space among geographically separated members of a group is called *telepresence* [29]. With the advent of Virtual Reality, telepresence evolved into *teleimmersion* [45], a merger of virtual reality and collaboration technology. The ultimate goal of teleimmersion is to design immersive portals, which offer rich communication modalities and where people can meet and collaborate as they would do in a real face to face event. In the ideal case they should not be able to perceive any difference to a real meeting venue [87].

2.2.1 Line-of-Sight Systems

The first attempts to satisfy the requirements of telepresence used an optical solution. *Video tunnels* used a combination of a mirror and a half-silvered mirror to achieve an apparent direct eye-to-eye contact as the mirrors effectively placed the camera right in the line of sight [29]. A step forward were the semi-transparent displays, *e.g.*, Fresnel lenticular displays [87], which allowed the light to pass through the display depending on its directionality. This enabled placing the camera physically behind the screen, in the line-of-sight. An example of a commercial system using a semi-transparent display is the videoconferencing solution by Teleportec [164]. Examples of other more recent promising alternatives which are currently only found in experimental setups are displays with holographic optical elements [87] and shuttered LCD projector-display combination [68]. A step towards a multi-party telepresence system is the France Telecom RD's Telepresence Wall, which uses a large multi-projection system serving a curved panorama screen and supports several lines of sight [87].

2.2.2 Round Table Systems

The requirement for a multi-party telepresence conference sessions triggered the so-called *round table concept*, where N parties are virtually grouped around a round table. This situation contains $N(N-1)$ different lines of sight, *i.e.*, every individual at the round table sees all the other $N-1$ conference participants. The first round-table experimental systems attempted to physically simulate the round table, *e.g.*, the Hydra system [140]. The drawback of this system was its complexity. The amount of video to be transferred increased

disproportionately with the number of multi-point connections. Each site of an N-party communication needed $(N-1)$ video links to the remote partners, resulting in $N(N-1)$ video links in total [87].

2.2.3 Shared Virtual Table Environments

More recent round-table approaches are based on the new concept which integrates a generic 3-D video representation of the conference participants into a *shared virtual table environment* (SVTE)[14, 13, 82, 87]. The round table concept is no longer built physically, but instead simulated by means of Virtual Reality (VR). In comparison with physical round table systems, the complexity is much lower as in this approach the same generic 3-D video representation is sent to all $(N-1)$ remote stations which use it to render the remote participants from the correct perspective.

Early experimental SVTE attempts include the MAJIC and MONJUnoCHIE systems [87]. A step further are TELEPORT [63] and US National Tele-Immersion Initiative [90], while the more recent solutions are the HHI's immersive meeting point (im.point) [162] and the France Telecom's RealMeet videoconferencing systems [87]. Other examples are Coliseum [18] and Gaze-2 [107], which are now focusing on PC desktop solutions [87].

2.2.4 VIRTUE

Based on the experiences of the tele-cubicles developed within the U.S. National Tele-Immersion Initiative [90], the European IST project VIRTUE (VIRtual Team User Environment) [82, 136] proposed a semi-immersive and transportable desktop solution for immersive videoconferencing [87].

The VIRTUE project proposed a new SVTE concept which integrated the shared virtual table environment with shared virtual working spaces. A seamless transition between the real table in front of the display and the virtual conference table in the screen gave the user the impression of being part of a single, extended perceptual and working space. The remote participants were rendered seamlessly and in the correct perspective into the virtual

conference scene using real video with adapted viewpoint [82].

The concept of adaptive viewpoint introduced the need for novel-view synthesis, where the views of remote participants were rendered so as to match the local participant's point of view. Ideally, the desired point of view would be achieved by placing a physical camera at the appropriate location, however, in case of VIRTUE, this would mean placing the camera in the middle of the screen, where it would either disrupt the immersive experience if placed in front of the screen or not be able to record anything if behind the screen as the screen was not transparent. The VIRTUE solution was to mount the cameras around the screen and use the so obtained video sequences to synthesise views for the virtual camera which better imitated the required point of view.

The novel-view synthesis method used in the VIRTUE project [93] relied on the high quality dense disparity maps derived from wide-baseline stereo pairs of cameras mounted around the screen. In order to achieve the required quality of disparity maps, they had to be post-processed after the matching step. This was achieved by first segmenting them into different disparity regions and using a hand mask to identify the hands. The disparities were then carefully interpolated and extrapolated within the individual regions to recover the missing values [135].

The problem of incomplete wide-baseline disparity maps which required post-processing for high-quality view synthesis inspired the work on disparity map completion, presented in this thesis.

2.3 Human Body Articulated Pose Estimation

Human body pose estimation is an active research area with solutions applicable in various domains including surveillance, motion capture, human gait analysis, human activity recognition, human-computer interaction and immersive communications.

Articulated pose estimation has been attempted in various ways including with and without full body models, using various optimisation techniques based on image, volumet-

ric and range data, and in the case of image data, using static images or sequences, with single or multiple viewpoints. In the following sections we discuss each of these approaches in more detail.

2.3.1 Surveys

Several survey papers giving an overview of the research in this area have been published, showing the extent of research interest in pose estimation and tracking. The not so recent ones include Aggarwal *et al.* [6], Cedras and Shah [33], Aggarwal and Cai [5], and Gavrilla [62]. The more recent surveys have been published by Moeslund and Granum [113] and Wang *et al.* [173]. The surveys by Buxton [28], Hu *et al.* [78] and Aggarwal and Park [7] also to some extent cover this research area but do not focus on it exclusively. The most recent survey papers are by Moeslund *et al.* [114] who focus on the research published after the year 2000, intended as a sequel of the survey published in [113] and by Poppe [121] who expands the taxonomy of the past survey papers and identifies current trends in this research field.

2.3.2 Body Model

Although majority of the articulated pose estimation problems use the notion of the kinematic tree, implicitly or explicitly, to represent the body pose, not all problems require the human body model to be represented as a full 3-D skeleton-based articulated model. In some problems, the kinematic structure can be represented in a much simpler way, with a collection of 2-D parts, or even just a stick figure. Although these cases require a simpler representation to actually estimate the pose, they can, however, still represent the final results with a more visually appealing model if necessary.

The full 3-D body model is usually used in the *analysis-by-synthesis* approaches which estimate the body pose by synthesising suggested pose with the model and evaluating its fit to the available constraints. Estimating the pose with a full 3-D body model has in the recent years been the most widely investigated approach to human pose estimation from video [114, 121] as the large number of different publications demonstrates.

The simplest representation is the one by Taylor [163] and Barron and Kakadiaris [21] who use a stick figure model. Poppe *et al.* [125], Parameswaran and Chellappa[116] and Mikić *et al.* [110] use a stick figure fleshed out with ellipsoids and cylinders representing the limb segments and torso, while Deutscher *et al.* [50] use a skeleton fleshed out with conical sections and Sminchisescu and Triggs [149] a skeleton fleshed out with superquadric ellipsoids. Drummond and Cipolla [53] use a representation based on intersections of pairs of quadrics. Implicit surface models are used by Herda *et al.* [72] and Plänkers and Fua[117]. Hilton *et al.* [73], Starck and Hilton [150] and Carranza *et al.* [31] use a generic humanoid mesh model and Allen *et al.* [10] a subdivision surface model. Cheung *et al.* [38] use a voxel model with an underlying skeleton, Caillette *et al.* [30] model the body as a mixture of Gaussian blobs which incorporate the information about the shape as well as colour of a particular body part, while Balan *et al.* [19] propose a new trend based on the work of Anguelov *et al.* [11] by using a detailed computer graphics mesh model of the human body. Cheng and Trivedi [37] use a multi-component Gaussian mixture model, where each component describes a rigid body segment and the entire collection is kinematically constrained according to a skeleton model. Husz and Wallace [79] model individual body parts using truncated frustums with constant elliptical subsection and heights. Sigal *et al.* [148] represent the body with a graphical model, in which each node represents a body part, and visualise the individual body parts using tapered cylinders.

The statistical methods for pose estimation from monocular images or sequences tend to focus on simpler model representations. Rosales *et al.* [130] use a vector of visual features and a Specialized Mapping Architecture (SMA) which has been trained to map feature vectors into 2-D body pose hypotheses. MacCormick and Isard [101] represent the hand pose with a spline parameterized by scale, orientation, x and y translation and 3 joint angles. Sidenbladh *et al.* [145] use a cylinder model to estimate 3-D body pose from monocular image sequences. Agarwal and Triggs [3] use a scaled prismatic model to estimate body pose in monocular image sequences. Felzenszwalb and Huttenlocher [57] use a pictorial structure, where an object is modelled by a collection of parts arranged in a deformable configuration with spring-like connections between certain pairs of parts.

Micilotta *et al.* [109] use a probabilistic body part assembly. Agarwal and Triggs [4] use shape descriptors of the silhouette edge and vectors of joint angles. Stefanov *et al.* [152] represent the articulated hand pose with a stick figure of the hand. Poppe [124] uses a histogram of oriented gradients as a pose descriptor which is matched against a training set of equivalent descriptors.

Subdivision Surface Models

Unlike in computer vision, in computer graphics the emphasis is on detailed and realistic body models. Starting with the initial mesh and spline surface modelling, a lot of research effort has recently been devoted to geometric modelling with subdivision surfaces [48, 137, 186] which are now an established tool for geometric modelling and animation and have in most cases successfully replaced traditional modelling with NURBS surfaces. They have also been used commercially by animation studios such as Pixar [156] and Blue Sky [16].

In Computer Vision, the majority of researchers still focus on simpler body models, however, work is emerging which attempts to set a new trend by using detailed and realistic human body models, *e.g.*, learned from databases of human body scans like in the work of Sigal *et al.* [147] and Balan *et al.* [19].

In our work, we model the human body with a Catmull-Clark subdivision surface. In this way, we can represent the human body with a simple and yet very powerful generic body model which can be simply customised to different individuals, used at different levels of complexity, as well as fit to clouds of unstructured data. Our model also to a large extent maintains the realism of the body shape at the joints, which is one of the main criticisms in [19] of the simplified approximations to human body models used so far.

2.3.3 Data

The pose estimation has been attempted from different data sources. While the majority of the work mentioned in this review estimates the pose from images directly, as that allows for

variety of scenarios from single to multi-view pose estimation, pose is also estimated from 3D stereo data [72, 117, 150], volumetric data [40, 110, 151, 37, 30], and range data[10].

2.3.4 Number of Images and Views

When the input data is image-based, various acquisition scenarios are possible. The pose can be estimated from a single image, a monocular image sequence or a multi-view image sequence.

Single Image

Pose estimation from a single image can be used for analysis of human posture in sports, medicine, archived photos, etc. It can also provide an initial guess for the tracking of human body pose in an image sequence.

The 3-D pose estimation from a single image is a difficult and ill-posed problem [114] as the depth information is not available. In order to compensate for the inherent ambiguity, a number of constraints are used to restrict the set of possible solutions. Taylor [163] assumes that the correspondences between image features and model joints are provided by the user and the relative lengths of the model segments are known *a priori* . He also assumes a scaled orthographic projection, which means that the method does not work on images with obvious perspective effects. Barron and Kakadiaris[21] also assume a scaled orthographic projection and make use of anthropometric statistics to constrain the estimation process. They rely on body segments that are almost parallel to the image plane to obtain good limb length estimates and cannot handle images which contain no such body segments. The body landmarks are marked by the user and so are the segments which are almost parallel to the image plane. Parameswaran and Chellappa [116] work with a general perspective camera. They also work with user-given location of the body landmarks and relative body lengths and assume a small torso twist and an isometry approximation where all subjects are assumed to have the same body part lengths when scaled. The user is expected to label the landmarks in terms of their relative proximity to the camera. Felzenszwalb and Huttenlocher [57] assume scaled orthographic projection and address the ambiguity of the reconstruction by training a pictorial structure model on a set of examples.

Monocular Image Sequence

Estimating the human body pose in an image sequence is usually adressed by various forms of tracking which ensure that a temporal consistency between the pose estimates in the individual frames is preserved. This helps to reduce the jitter typically present if the body pose is estimated on a frame-by-frame basis only and helps to constrain the search space. In the case of a monocular image sequence, the ambiguity and ill-posedness challenges of the 3-D pose estimation from a single static image remain unchanged as, apart from the temporal consistency, the conditions remain the same, *i.e.*, the depth information is lost. A number of researchers have addressed 2-D and 3-D pose tracking in a monocular image sequence.

MacCormick and Isard [101] use a particle filter with partitioned sampling to cope with the dimensionality of the search when tracking the hand joints. Sidenbladh *et al.* [145] estimate a full 3-D pose from a monocular image sequence by incorporating the knowledge of strong priors on human motion. Sminchisescu and Triggs [149] initialise their 3-D model with given model-image correspondences and use the corresponding covariance matrix to generate the initial set of hypotheses which are propagated in time as a Gaussian Mixture. They use covariance scaled sampling to ensure that the cost function is sampled widely enough to avoid getting caught in local minima. Local constraint-consistent optimization is used to find the centre of the associated likelihood peak of a sample. Agarwal and Triggs [3] use a scaled prismatic model to estimate the pose of a walking, running and turning person in a monocular image sequence. Micilotta *et al.* [109] train a probabilistic body part assembly model and use it to estimate the 2-D pose in a static image first and then use the estimate as a constraint in the next frame. Agarwal and Triggs[4] don't use an explicit model. They encode the pose as a vector of joint angles and use the shape description vectors derived from the image silhouette edges to encode the information about the body pose. They train different sparse Bayesian nonlinear regression models on a set of training examples and use them to find and track the body pose in a monocular sequence. Poppe [124] estimates the pose from training examples using histogram of oriented gradients and also reports results on multi-view sequences.

Multi-View Image Sequence

With the advent of affordable multi-camera setups and the growing interest of the computer vision and graphics community in the media and entertainment industry, studio setups acquiring sequences of actors with multiple cameras have become a popular research area.

Deutscher *et al.* [50] use the concept of Simulated Annealing in the particle filtering framework to perform a particle-based stochastic search. The idea is to reduce the complexity of the particle filter which is known to be inefficient in a high-dimensional search space. They estimate and track the full pose of a human body using 3 cameras. Rosales *et al.* [130] train the Specialised Mapping Architecture (SMA) model on 32 views and use it to estimate the body pose on 3-camera test sequences. Cheung *et al.* [38] use sequences acquired by 8 cameras to build a bounding edge visual hull of the imaged person over time. They combine the colour surface points of the visual hull with the image colour information to estimate the rigid motion of individual body segments and finally connect these into an articulated structure. The computed articulated model is then used to track the same person in new video sequences. Carranza *et al.* [31] use 8 cameras to record an actor performing in a studio. Off-line, they analyse the actor's motion by performing a nonlinear optimisation over the skeleton joint angles using the silhouettes as constraints. They texture map the generic humanoid model and then, using the model, render the actor's performance from an arbitrary viewpoint. Plänkers and Fua [117] use 3 cameras and an implicit surface (metaballs) based human body model to estimate the shape and pose using the Levenberg-Marquardt nonlinear least-squares estimator constrained by the silhouette and stereo data derived from the image sequence. Mikic *et al.* [110] use 6 cameras and the Extended Kalman Filter to estimate and track the pose of a full human body model in a volumetric data sequence which they derive from the input image sequences. Sigal *et al.* [148] track a walking person in a 4-camera video sequence. Husz and Wallace [79] use a hierarchical partitioned particle filter on multi-view HumanEva test sequences and test the performance for different number of views. Caillette *et al.* [30] perform a hierarchical volumetric reconstruction from 5 views and fit a mixture of Gaussian blobs to the reconstructed volumetric data.

2.3.5 Optimisation Methods

There is a general consensus among the researchers in the computer vision community that estimating an articulated human body pose is a complex multi-dimensional problem. In view of this, a number of different algorithms and approaches to estimating the articulated pose have been reported, including optimisation, filtering, learning algorithms and combinations of these.

Several researchers attempted to recover the body pose through some form of optimisation. Taylor [163] provides a closed form solution giving a family of possible pose estimates in a single image, parameterised by the scale. He suggests that the actual pose then be obtained by a standard optimisation method over the value of the scale parameter. Barron and Kakadiaris [21] use the BFGS (Broyden-Fletcher-Goldfarb-Shanno) nonlinear optimisation over joint angles and segment lengths to estimate the body pose from a single uncalibrated image. Allen *et al.* [10] use the BFGS optimisation to estimate the upper body pose of their range scans by minimising the distance between the calculated and observed marker positions. Carranza *et al.* [31] use Jacobian optimisation to recover the segment lengths and Powell's method to optimise over the joint angles of an articulated full-body model. Cheung *et al.* [38] use the Levenberg-Marquardt optimisation to estimate the motion parameters of the individual rigid body segments. Plaenkers and Fua [117] also use the Levenberg-Marquardt optimisation to estimate the joint angles of the implicit surface model by minimising the distance between the model and the stereo and silhouette observations. Chun *et al.* [40] describe an iterative solution which finds the nonlinear axes of the volumetric cloud of genus 0, representing the human body. Herda *et al.* [72] use a Levenberg-Marquardt optimisation on a Damped Least Squares problem which they use to incorporate motion-capture derived joint constraints into the pose estimation.

When working with sequences, the temporal consistency is conveniently modelled using a Bayesian filtering framework. Deutscher *et al.* [50, 51] use an annealed particle filter, a particle-based stochastic search which draws the inspiration from the simulated annealing. The algorithm samples the cost function in a coarse-to-fine manner and so avoids getting trapped in local minima. MacCormick and Isard [101] use a particle filter with par-

titioned sampling to highlight the fact that the kinematic structure of the human body pose can be estimated in a hierarchical fashion. Sidenbladh *et al.* [145] represent the human motion with a large database of example motions, which they structure into a binary tree to allow for efficient search and combine with a particle filtering framework to estimate the articulated 3-D body pose from monocular sequences. Mikic *et al.* [110] use an extended Kalman filter to track the pose of a full articulated human body model in volumetric data derived from image silhouettes. Sminchisescu and Triggs [149] use a particle filter with covariance-scaled sampling which ensures that the space of possible pose configurations is sampled widely enough to prevent the filter from getting stuck in the local minima. Sigal *et al.* [148] learn conditional probabilities relating the 3-D pose of connected limbs and the temporal motion priors for each limb from a database of motion capture data. They then estimate the human pose and motion with non-parametric belief propagation using a variation of particle filtering. Hou *et al.* [76] combine a particle filter framework with a back-constrained Gaussian process latent variable model (GPLVM) and the variable length Markov model (VLMM). They use the GPLVM to obtain a low-dimensional representation of the training motion data and learn a motion prior for the particle filter, while the VLMM captures the temporal dependencies between the elementary movements. Husz and Wallace [79] use a hierarchical partitioned particle filter, where a switching model combines the Gaussian noise model with action primitives predicting the next pose on the basis of the previous poses. Stefanov *et al.* [152] train a VLMM on training sequences exhibiting particular behaviour and combine it with the annealed particle filter to track the articulated hand pose in a video sequence. Caillette *et al.* [30] use a similar approach for articulated full-body tracking from volumetric data. They integrate the annealed particle filter with a VLMM which can explain high-level behaviours over a long history, so that both continuous movements and a discrete representation of a structured behaviour are jointly represented.

Learning models of articulated body pose from training sets is attractive when attempting to estimate the pose from a monocular image sequence or a limited number of views, where the trained models help to disambiguate the pose to some extent. Rosales *et al.* [130] train a Specialised Mapping Architecture (SMA) to take silhouette-based feature vectors as

input and a set of 2-D body pose hypotheses as an output. The 3-D body pose and the camera positions are then found using an EM algorithm on a probabilistically formulated structure from motion problem. Agarwal and Triggs [3] learn autoregressive models for different classes of human motion and use them to track body movements in monocular video sequences. Felzenszwalb and Huttenlocher [57] find the pose of a full body in a single image by sampling the posterior distribution of object configurations represented with pictorial structures to obtain multiple hypotheses of the body pose. The pictorial structure model is trained on 10 images and then tested on new images by evaluating the fit of each sample configuration by calculating the Chamfer distance between the model and the input image silhouette. Micilotta *et al.* [109] learn a Gaussian mixture prior from hand-labelled examples of upper-body configurations and combine it with a RANSAC-based body configuration selection process. Lee and Cohen [92] use a data driven Markov chain Monte Carlo approach to estimate the articulated 3-D pose from still images and learn the priors for the pose parameters from manually labelled joint positions in a training set of images. Agarwal and Triggs [4] train several different nonlinear regression models of joint angles using the histograms of the silhouette edge as an input and use them to estimate the body pose from a monocular sequence. Urtasun *et al.* learn PCA motion models for walking, running and golf swing in [172] and balanced Gaussian process dynamical model (GPDM) priors for various walking styles in [171] and combine them with a hill-climbing optimisation to track people in single and multiple views. Roberts *et al.* [126] recover part-based descriptions of human pose from single real-world images by learning view based probabilistic models of body part shapes and formulating an approach which allows principled estimation of partial configurations. Poppe [124] estimates the pose from examples by computing a histogram of oriented gradients for the images in the training set and finding the pose in a new image by interpolating 25 best matching histograms from the training set. Urtasun and Darrell [170] describe an efficient method for learning the appearance to pose mapping by training a local mixture of Gaussian processes on a database of marker-based motion-capture-labelled video sequences to estimate the 3-D body pose from monocular still images. Similarly, Bo *et al.* [23] address the problem of scalability of large scale discriminative methods to articulated pose estimation by presenting an efficient method to train conditional Bayesian Mixture of Experts which allows for training of models that are one order of magnitude

larger, in time that is more than one order of magnitude faster than previous methods, and demonstrate the results on 3-D pose estimation from monocular images.

Human Body Pose Estimation with Evolutionary Techniques

The Evolutionary Computation research addressed the articulated body pose estimation and analysis in various ways including using Genetic Algorithms [181, 144, 77] and Neural Networks [69, 67]. Particle swarm optimisation has been successfully applied to various problems, however, we were not able to find many references to its use for articulated body pose estimation and our conclusions have been confirmed by the recent survey of PSO application areas by Poli [118]. The few existing related work publications are by Schutte *et al.* [138] who report a parallel PSO implementation and illustrate its performance on an example of an ankle joint, Robertson *et al.* [128] who use the PSO to track the stick figure in reconstructed stereo data and Robertson and Trucco [127] who use PSO to fit a stick model to sparse 3-D stereo data reconstructed from an array of cameras.

2.4 Dense Wide-Baseline Stereo Correspondence

Dense stereo correspondences are typically generated with some form of area-based correspondence search using an image-intensity-based similarity measure to determine the matching regions [133]. In order to constrain the search and reduce its complexity, additional information, such as epipolar geometry and anticipated depth of the scene, is used. As the main source of information available is the image intensity, in practice, a number of factors exist which make the search for dense image correspondences a difficult problem.

Different cameras tend to have a different colour response, untextured regions are a source of ambiguity, the number of occlusions in the scene (regions only visible to one camera) increases with the scene complexity, and the illumination conditions, when not sufficiently controlled, may vary between different viewpoints. In a wide-baseline stereo configuration these problems become even more pronounced and require further consideration when designing dense correspondence algorithms.

Recently, a significant progress has been achieved in the state-of-the-art dense stereo matching algorithms, demonstrated on the Middlebury stereo page [134]. While the performance of these methods is undeniably very impressive, the test images used for performance evaluation are acquired with a baseline of 160 mm, which is relatively small compared to the 700 mm baseline which we used in our setup.

Wide-baseline stereo matching algorithms, on the other hand, are concerned with matching very different views of the same scene, related by affine distortion, large viewpoint change, image blur, image compression, and changes in lighting conditions. These methods proceed by first detecting a set of interest points or regions in each of the input images and then computing their image descriptors, such as, e.g., scale invariant feature transform (SIFT) [98] or gradient location and orientation histogram (GLOH) [111]. Sparse correspondences between the interest points in the two views are then computed through the process of matching their image descriptors.

The middle ground, which is the closest to our wide-baseline scenario, is represented by image-based modelling methods, which are interested in generating detailed 3-D models of the scene from stereo. Most of these wide-baseline approaches generate dense disparity maps by using the interest point approach to generate initial matches and then passing these as an input to a dense stereo correspondence estimation method. Usually, only a limited number of interest points are computed to limit the computational complexity of the approach. Strecha *et al.* [155] propose a multi-view stereo system where they start by matching affine invariant regions across N input views, followed by propagating these by an inhomogeneous time diffusion process that takes into account the matching of all views. Strecha *et al.* [154] address the N-view stereo problem with a probabilistic approach, introducing the prior knowledge in terms of depth smoothness. They solve the N-view stereo problem with an EM algorithm iterating between the pixel visibility map estimation and the computation of depth values, which must be consistent across all input views. Kannala and Brandt [85] compute quasi-dense point correspondences by detecting affine covariant regions, combining them with SIFT descriptors and using the matches as seeds for match propagation which models the local transformation between corresponding image patches

with an affine model. Their method is much faster than the related approaches, however, at the expense of computing only quasi-dense wide-baseline disparity maps. Recently, Tola *et al.* [165] have proposed a novel local image descriptor, called DAISY, which they feed to a graph-cuts based dense depth map estimation algorithm. They report that, unlike competing techniques which require many high-resolution images to produce good reconstructions, their method is capable of producing good 3-D reconstructions from pairs of low-quality images.

2.5 Fitting Subdivision Models to Unstructured Data

Although they have been around for a while, with the first publications in 1978 by Catmull and Clark [32] and Doo and Sabin [52], subdivision surface models are still primarily a domain of computer graphics rather than computer vision. In computer graphics, one of the important research areas is fitting subdivision surfaces to other mesh models with the aim of producing a simplified model with an equivalent visual quality. The area that we are interested in, however, is fitting subdivision surfaces to unstructured clouds of 3-D data points.

An overview of subdivision surfaces, which also touches on subdivision surface fitting, has been published by Ma [99]. Hoppe *et al.* [75] use a Loop subdivision surface augmented with rules for various irregular vertices and fit it to scattered range data. They formulate the fit as a linear least squares optimisation which looks for the best data approximation with the most concise surface. Suzuki *et al.* [158] generate a subdivision surface which approximates the unstructured point cloud but does not capture the fine features by formulating the fit as a set of simultaneous linear equations for which they propose an iterative algorithm. Ma and Zhao [100] use the Catmull-Clark subdivision surfaces for the reverse engineering problem where they fit a subdivision model to CT scan data and formulate the fit as a least squares problem. Litke *et al.* [96] fit a Catmull-Clark subdivision surface model to high-quality range data using the method of quasi-interpolation. Jeong and Kim [83] reconstruct a displacement subdivision surface from an unorganised point cloud by employing a local subdivision surface fitting of Suzuki *et al.* [158] and a geometric sampling technique to

recover the scalar offsets. Allen *et al.* [10] fit an articulated displaced subdivision surface model of the upper human body to a set of range scans of the upper body parts and use the resulting deformed models in a number of pre-set poses to animate the human body by linearly blending the available shape examples. Cheng *et al.* [35] fit a Loop subdivision surface to a set of unorganised data points by means of iterative squared distance minimisation. Cheng *et al.* [36] discuss three different optimisation methods for subdivision surface fitting, point distance minimisation, tangent distance minimisation and squared distance minimisation, to unorganised point sample data and analyse their convergence properties.

Not many attempts have been made to fit subdivision surface models to stereo data. We identified the work by Ilić [80], who fits Loop subdivision surface models of a face to noisy stereo data using a least-squares optimisation framework.

2.6 Disparity Space

The idea that the disparity values can in some way be interpreted as either two-dimensional images or a three-dimensional space, has been used by various researchers to facilitate the search for image correspondences in stereo vision.

Disparity space related research can be roughly split in two subgroups - the *3-D disparity space* methods and the *disparity space image* methods. 3-D Disparity Space (x, y, d) is a more general concept, a three-dimensional projective space defined as a projective transform of the 3-D space, where x and y are the column and row dimensions of the image plane, respectively, and d is the disparity of a left image point with respect to the corresponding right image point. *Disparity space image* (DSI) is an image or a function defined over a continuous or discretised version of the disparity space, explicitly representing the matching space, and is constructed to facilitate the stereo matching process like in [24, 180, 133].

The concept of a two-dimensional *disparity space image* (DSI) appears in the literature in various forms. Tsai and Katsaggelos [168] construct a DSI where the x-axis contains the indices of the left image scanline and the y-axis contains the indices of the right image

scanline. The disparity estimation between the two scanlines is then equivalent to finding a path with the smallest cost from the lower left corner across to the upper right corner of the DSI. They combine DSI with a Divide-and-Conquer Technique to estimate stereo disparity with integrated occlusion detection. Intille and Bobick [81] describe another type of DSI, where the *x*-axis contains the indices of the left image scanline and the *y*-axis contains the disparity values. They present DSI as a data structure containing an explicit representation of the matching space. The correspondence search is formulated as a dynamic programming algorithm finding the best path through the DSI which simultaneously finds matches and occlusions. In [161] Szeliski and Scharstein define the DSI as a squared difference image, analyse the properties of a continuous DSI and discuss the issue of DSI generation.

Addressing the problem of structure from stereo, Braunegg[26] used *disparity space planes* to improve the efficiency of a stereo matching algorithm devised by Marr, Poggio and Grimson. Szeliski and Golland[160] formulated the problem of simultaneously recovering the disparities, true colours and opacities in a generalised 3-D (x,y,d) disparity space and solved it using iterative energy minimisation. Trucco *et al.* [166] reported an algorithm for plane detection in disparity space and Hong and Chen[74] described a segment-based stereo matching algorithm which fit disparity planes to segments of disparity data.

The three-dimensional disparity space idea is equally popular. Scharstein and Szeliski [132] present a Bayesian approach to stereo correspondence with non-linear diffusion, where the support for a correspondence match is defined over a three-dimensional disparity space $E(x,y,d)$. After aggregating the support in this disparity space, the final disparity value for a particular image pixel, $d(x,y)$, is computed as $\min E(x,y,d)$ over possible disparity values $d \in D$. Zitnick and Kanade [184] describe a stereo algorithm in which they define disparity space as a 3D space with dimensions row *r*, column *c* and disparity *d*. They construct a three-dimensional array of match values in the disparity space, where each element of the array corresponds to a pixel in the reference image and a disparity relative to another image. A local 2D or 3D support is then chosen, depending on the nature of the surface, and used to refine the match values by means of iterative averaging while imposing the uniqueness and continuity constraints. The algorithm explicitly detects occlusions by

thresholding the match value.

In the area of motion estimation from stereo, Demirdjian *et al.* [47] described rigid 3-D motion estimation in disparity space. Derpanis and Chang[49] reported an extension of [47] with a closed form linear solution for rigid motion estimation in disparity space. Agrawal and Konolige[8] described a mobile robot localisation system for outdoor environments which estimated the motion in disparity space and Hattori and Takeda[71] reported dense stereo matching in disparity space used in an implementation of a side collision system for a road vehicle.

2.7 Novel-View Synthesis

The problem of novel-view synthesis can be stated as follows. Given a limited number of images of a particular scene, generate novel views of the same scene, *i.e.*, views not included in the original set and therefore "not seen before". The broad spectrum of the novel view synthesis techniques can be roughly split into three different groups.

2.7.1 Interpolation Methods

On one end of the spectrum are the algorithms which assume (almost) no knowledge of scene geometry. Instead, they use a large number of images acquired from fairly densely sampled viewpoints to produce realistic results. The acquired images are typically warped and linearly interpolated to produce a novel view. These approaches result in elaborate image storage and retrieval requirements and generally restrict the viewing space of the synthesised views. An example of such a technique is the lightfield [94] where the images are interpreted as 2-D slices of a 4-D lightfield function and novel views created by combining and resampling available images without the need for depth information or feature matching. Gortler *et al.* [65] use a technique similar to that of Levoy and Hanrahan [94], where they sample and reconstruct a 4-D function which they call the *lumigraph*. Unlike [94], they also use a rough 3-D shape approximation in the form of an octree volumetric model which aids the capture and reconstruction of images from the lumigraph. McMillan

and Bishop use the plenoptic modelling approach [108] where the view synthesis is formulated as an attempt to reconstruct the plenoptic function from a sample set of that function. The discrete samples have a form of cylindrically projected panoramic images. Disparity images relating each cylinder pair are computed and cylindrical samples together with the disparity information are then used to generate image warps that map reference images to novel views.

2.7.2 Explicit 3-D Reconstructions

On the other end of the spectrum are the methods which explicitly reconstruct a 3-D geometry of the scene in the form of a textured 3-D model, which is then rendered from novel viewpoints. Kanade *et al.* [84] make extensive use of image-based stereo to produce Visible Surface Models which are then combined to create a textured triangle mesh model which is rendered from different viewpoints. Starck and Hilton [150] deform a generic computer graphics model to match the stereo, silhouette and feature constraints. The model is then textured and an animation rendered from novel views. Plänkers and Fua [117] use an implicit surface model and recover its shape and motion from stereo and silhouette data. Carranza *et al.* [31] also estimate the pose of a generic 3-D model from image silhouettes and render novel views of the textured model. Silhouette-based constraints are exploited by Matusik *et al.* [106], who compute a polyhedral visual hull and render it from novel views using view-dependent texture. Combining visual hulls and correspondence over time, Cheung *et al.* [39] compute a temporal visual hull from silhouette constraints and render it from novel viewpoints.

2.7.3 Geometric Constraint Methods

Between these two extremes are the image-based rendering methods which render novel views from existing images without explicity reconstructing the full 3-D scene. Instead, they restrict the number of necessary input images by implicitly incorporating the knowledge of scene geometry in the view synthesis process through the use of stereo correspondence information and various geometric constraints.

In the view interpolation approach by Chen and Williams [34], dense correspondences are found between a pair of views and then used together with the image texture as an input to a morphing algorithm which generates an in-between view. Laveau and Faugeras [91] synthesise novel views by taking advantage of the epipolar geometry combined with the stereo correspondence information for the pairs of input views. Seitz and Dyer [139] use the view morphing approach which preserves the 3-D effects. They first pre-warp the input images by applying appropriate projective transforms, then compute a morphed novel image by linearly interpolating positions and colors of corresponding points in the input image pair and finally post-warp the resulting image to obtain the final novel view. A correspondence map between the two input views is required. Avidan and Shashua [15] use the trilinear tensor geometric constraint on three views, two of which are the input views while the third is a novel view. The dense correspondence map between the two input views combined with the tensor point transfer method is used to synthesise the novel view. Connor and Reid [41] report a backward mapping scheme for trifocal transfer, an approach similar to Li and Hartley [95], who describe an inverse trilinear tensor transfer approach, where, for every position in the virtual image, they sample the corresponding epipolar line in a reference image and for every sample find a set of matching points across several views using the trilinear tensor constraint. The texture of a set of samples which maximises a colour consistency metric is then assigned to the pixel in the virtual view. Connor and Reid [42] construct a multiple-view layered representation for tracking and segmentation of multiple objects in a scene by estimating layer parameters consistent across different views. The layered representation then provides a convenient rendering model for dynamic novel-view synthesis by view interpolation. Criminisi *et al.* [43] report an improved dynamic programming stereo algorithm for efficient novel-view generation and describe a method using a direct projection of the minimum-cost surface for novel-view synthesis, used for gaze manipulation in one-to-one videoconferencing.

Strecha *et al.* [154] approach the view synthesis from a probabilistic point of view, introducing the prior knowledge in terms of depth smoothness. They first formulate an N-view stereo problem, which they solve with an EM algorithm iterating between the pixel

visibility map estimation and the computation of depth values, with one of the input images chosen as the reference view. The algorithm then naturally extends into a novel-view synthesis approach by specifying the reference image as the virtual image and thus performing view interpolation. A slightly different approach is used by Fitzgibbon *et al.* [58], who constrain the novel views to lie in the space of the input images through the use of a Bayesian approach combining a likelihood in the form of a photoconsistency constraint with an image-based texture prior modelled as 5×5 texture patches sampled from the set of input images. Woodford *et al.* [177] improve on the work in [58] by reducing the patch priors to local pairwise priors which then allow for solving the problem with global optimisation and Shahrokni *et al.* [141] extend this work further by using temporal priors for rendering novel video sequences. Another approach by Woodford *et al.* [178] looks at the novel-view synthesis using fields of experts, a technique originally developed for the purpose of inpainting and denoising images. The fields of experts framework explicitly models the inter-dependence of neighbouring (overlapping) image patches by learning a parametric prior over the patches and enables faster inference than the non-parametric texture priors framework used in [58].

The implicit methods frequently rely on the knowledge of dense correspondences between the available input views. In these cases, the correctness and completeness of the correspondence map is crucial for high quality novel view synthesis. In view of this, several researchers focus on producing a high quality disparity map to achieve better novel view synthesis. Zitnick *et al.* [185] achieve convincing results in dynamic scene rendering by using a colour-segmentation based stereo algorithm and extracting mattes for the areas near the depth discontinuities. In 3DTV context, Kauff *et al.* [86] post-process the disparity maps after the consistency check by first segmenting them based on colour clustering and change detection and then using a variety of interpolation operators on the segmented areas, depending on their properties. In 3-D Video, Waschbüsch *et al.* [176] use a "video brick" which acquires texture and structured light stereo images simultaneously. They then use a multi-window based matching algorithm with a subsequent subpixel disparity refinement to compute accurate disparity maps.

2.8 Conclusion

In this chapter we presented the literature review of the topics relevant to this thesis. We looked at the research areas in immersive videoconferencing, articulated human body pose estimation, fitting subdivision models to unstructured data, disparity space and novel-view synthesis. In the next chapter, we describe the background theory required for the work presented in subsequent chapters.

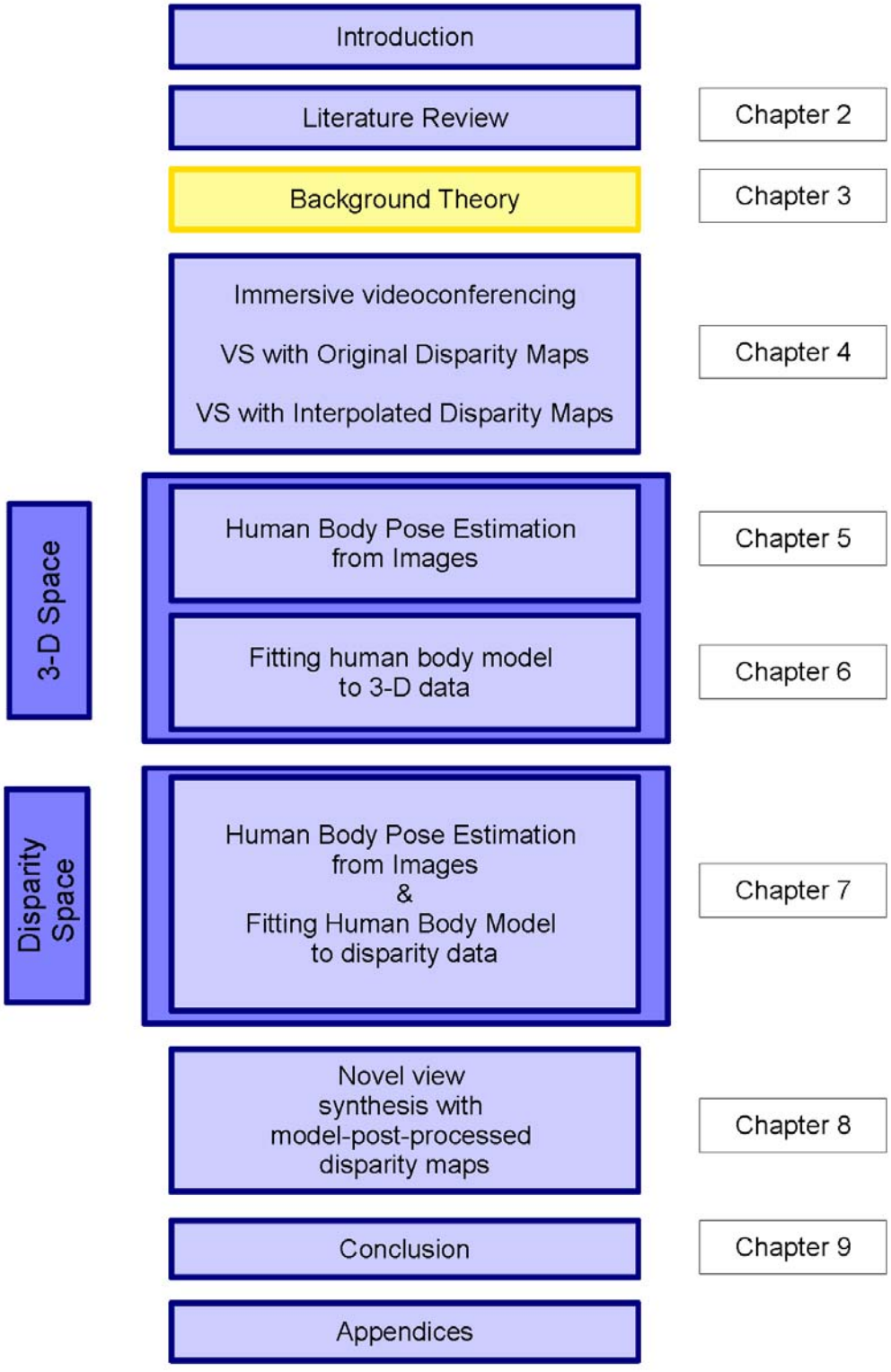


Figure 2.1: The thesis structure diagram.

Chapter 3

Background Theory

3.1 Introduction

In this chapter, we present the background theory relevant to the contents of this thesis. Each section describes a method that we used in our disparity map completion framework. The contents of this chapter are split into three thematic parts.

In the first part, we present the methods necessary to generate the disparity data and the reconstructed 3-D point cloud. We begin with camera calibration in Section 3.2, followed by image rectification in Section 3.3. Disparity map correspondence search is the topic of Section 3.4, while Section 3.5 describes the 3-D reconstruction algorithm.

In the second part, we look at the methods needed to model, manipulate and deform a human body model. We present the subdivision surfaces, which we used to model the upper human body, in Section 3.6, and describe the quasi-interpolation method used to deform the surface of the subdivision body model to fit the data in Section 3.7. The particle swarm optimisation method, used to estimate the body pose of the human body model, is presented in Section 3.8.

The last part consists of Section 3.9, where we look at the trilinear tensor transfer novel view synthesis algorithm, the performance of which we want to improve with disparity map completion. We conclude in Section 3.10.

3.2 Camera Calibration

3.2.1 Introduction

The work presented in this thesis requires a calibrated multi-camera setup. In order to calibrate it, we use a multi-camera self-calibration algorithm, which we describe in this section. In Section 3.2.2, we first explain the conventional camera calibration with a known calibration object in 3-D and highlight the need for camera calibration from image information alone. In Section 3.2.3 we describe the multi-camera self-calibration approach.

3.2.2 Conventional Camera Calibration

Given a set of n known 3-D points $\mathbf{X}_j = [X_j, Y_j, Z_j, 1]^\top$ and their 2-D images $\mathbf{x}_j = [x_j, y_j, 1]^\top$, $j = 1, \ldots, n$, the problem of camera calibration is finding a camera projection matrix, P, which satisfies the following projection equation:

$$\lambda \mathbf{x}_j = P\mathbf{X}_j, \tag{3.1}$$

where λ is the *projective depth* of the point $\mathbf{X}_j$.

The standard way of determining the matrix P is to acquire images of a *calibration object* (see Figure 3.1) with known 3-D coordinates $\mathbf{X}_j$ and compute the camera matrix P which minimizes the error between the measured image coordinates $\mathbf{x}_j$ of the calibration object and the projected 3-D coordinates $P\mathbf{X}_j$:

$$\arg\min_P E(\mathbf{x}_j, P\mathbf{X}_j), \qquad j = 1, \ldots, n, \tag{3.2}$$

where E is an error measure, usually a sum of squared distances.

Apart from being called the *conventional* method of camera calibration, in the literature, this method of calibration is also referred to as *photogrammetric calibration* [183] and *resectioning* [70]. Tsai calibration method [169] is a classic example of this type of calibration.

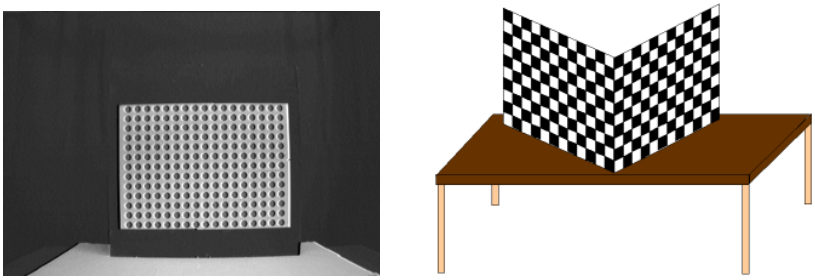


Figure 3.1: Example calibration objects used in a conventional camera calibration method. The origin of the world coordinate system is usually defined to lie on the calibration object and the 3-D coordinates of the points on the calibration object are relative to the chosen world origin.

In Equation (3.1), the relationship between the world coordinates $\mathbf{X}_j$ and image coordinates $\mathbf{x}_j$ is modelled as a linear transformation. This is only true if using a *pinhole* camera model. When using real lenses, *radial (and tangential) lens distortion* must be accounted for as well. The Gold Standard Algorithm [70] for estimating the matrix P consists of four main steps (normalisation, direct linear transform, geometric error minimisation, denormalisation) and the additional step of radial distortion estimation. We describe each of these steps in more detail next, as given in [70].

Gold Standard Algorithm for Estimation of P

1. Normalisation. Use a similarity transformation T to normalise the image points, $\tilde{\mathbf{x}}_j = T\mathbf{x}_j$, and a similarity transformation U to normalise the 3-D points, $\tilde{\mathbf{X}}_j = U\mathbf{X}_j$. The normalisation process involves moving the origin to the centroid of the points and scaling the points so that their root-mean-squared distance from the origin is $\sqrt{2}$ for the image points and $\sqrt{3}$ for the 3-D points.

2. Direct Linear Transform. Each correspondence $\tilde{\mathbf{x}}_j \leftrightarrow \tilde{\mathbf{X}}_j$ satisfies the projection equation $\tilde{\mathbf{x}}_j \simeq \tilde{P}\tilde{\mathbf{X}}_j$, where $\simeq$ denotes that the homogeneous vectors $\tilde{\mathbf{x}}_j$ and $\tilde{P}\tilde{\mathbf{X}}_j$ are equal up to a scale factor. This relationship can be expressed with the cross-product as:

$$\tilde{\mathbf{x}}_j \times \tilde{P}\tilde{\mathbf{X}}_j = 0. \tag{3.3}$$

Let us write the i-th row of the projection matrix $\tilde{P}$ as $\mathbf{P}^{iT}$:

$$\tilde{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} \mathbf{P}^{1T} \\ \mathbf{P}^{2T} \\ \mathbf{P}^{3T} \end{bmatrix} \tag{3.4}$$

and the coordinates of the image vector $\tilde{\mathbf{x}}_j$ as $\tilde{\mathbf{x}}_j = [x_j, y_j, w_j]^\top$. The cross product then becomes:

$$\tilde{\mathbf{x}}_j \times \tilde{P}\tilde{\mathbf{X}}_j = \begin{bmatrix} x_j \\ y_j \\ w_j \end{bmatrix} \times \begin{bmatrix} \mathbf{P}^{1T}\tilde{\mathbf{X}}_j \\ \mathbf{P}^{2T}\tilde{\mathbf{X}}_j \\ \mathbf{P}^{3T}\tilde{\mathbf{X}}_j \end{bmatrix} = \begin{bmatrix} y_j\mathbf{P}^{3T}\tilde{\mathbf{X}}_j & - & w_j\mathbf{P}^{2T}\tilde{\mathbf{X}}_j \\ w_j\mathbf{P}^{1T}\tilde{\mathbf{X}}_j & - & x_j\mathbf{P}^{3T}\tilde{\mathbf{X}}_j \\ x_j\mathbf{P}^{2T}\tilde{\mathbf{X}}_j & - & y_j\mathbf{P}^{1T}\tilde{\mathbf{X}}_j \end{bmatrix} = \mathbf{0}. \tag{3.5}$$

Using $\mathbf{P}^{iT}\tilde{\mathbf{X}}_j = \tilde{\mathbf{X}}_j^\top \mathbf{P}^i$, we can reformulate the Equation (3.5) as a set of three equations in the unknown elements of $\tilde{P}$:

$$\begin{bmatrix} \mathbf{0}^\top & -w_j\tilde{\mathbf{X}}_j^\top & y_j\tilde{\mathbf{X}}_j^\top \\ w_j\tilde{\mathbf{X}}_j^\top & \mathbf{0}^\top & -x_j\tilde{\mathbf{X}}_j^\top \\ -y_j\tilde{\mathbf{X}}_j^\top & x_j\tilde{\mathbf{X}}_j^\top & \mathbf{0}^\top \end{bmatrix} \begin{bmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{bmatrix} = \mathbf{0}. \tag{3.6}$$

The three equations in (3.6) are linearly dependent, *i.e.*, the third one can be obtained as a linear combination of the first two. As a result, only two equations are normally used and each correspondence pair $\tilde{\mathbf{x}}_j \leftrightarrow \tilde{\mathbf{X}}_j$ is said to contribute two constraints in the unknowns of $\tilde{P}$:

$$A_i = \begin{bmatrix} \mathbf{0}^\top & -w_j\tilde{\mathbf{X}}_j^\top & y_j\tilde{\mathbf{X}}_j^\top \\ w_j\tilde{\mathbf{X}}_j^\top & \mathbf{0}^\top & -x_j\tilde{\mathbf{X}}_j^\top \end{bmatrix} \begin{bmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{bmatrix} = \mathbf{0}. \tag{3.7}$$

In order to solve for 11 unknown elements of the matrix $\tilde{P}$ (as it is defined only up to a scale), at least 6 point correspondences are required. As the image measurements are usually corrupted by noise, the rule of thumb is to use a number of constraints which exceeds the number of the unknowns by a factor of five [70], *e.g.*, 28 points provide 56 constraints in the 11 unknown elements of the camera matrix $\tilde{P}$.

All available constraints are stacked up to give a homogeneous system of linear equations:

$$A\mathbf{p} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ \vdots \\ p_{34} \end{bmatrix} = 0, \tag{3.8}$$

where p_{ij} are the elements of the unknown projection matrix $\tilde{P}$ and A_i is as in Equation (3.7). The solution is obtained by performing the singular value decomposition of the matrix A, where the unit singular vector corresponding to the smallest singular value of A is the solution vector $\mathbf{p}$.

3. Minimise geometric error. Using the linear estimate from the previous step as a starting point, minimise the geometric error:

$$\arg\min_{\tilde{P}} \sum_{j=1}^{n} d(\tilde{\mathbf{x}}_j, \tilde{P}\tilde{\mathbf{X}}_j)^2 \tag{3.9}$$

over the entries of $\tilde{P}$, using an iterative algorithm such as Levenberg-Marquardt [122].

4. Denormalisation. The camera matrix P for the original coordinates (before normalisation) is obtained from $\tilde{P}$ as

$$P = T^{-1}\tilde{P}U. \tag{3.10}$$

Radial Distortion. The pinhole camera is a linear approximation of the imaging process. In reality, the camera contains a lens which is used to focus the light rays onto the image plane. The camera lens contains various aberrations, the most important of which is the radial distortion [60]. The effect of the radial distortion can be seen by identifying straight lines in the scene which appear curved in the image, the more so the further away from the centre of distortion they are imaged.

The radial distortion is modelled as

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = L(\tilde{r}) \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}, \tag{3.11}$$

where $(\tilde{x},\tilde{y})$ is the undistorted image position, (x_d,y_d) are the distorted image coordinates, $\tilde{r} = \sqrt{\tilde{x}^2+\tilde{y}^2}$ is the radial distance from the centre of radial distortion and $L(\tilde{r})$ is the distortion factor which is a function of the radius $\tilde{r}$.

The distortion correction is written as:

$$\begin{aligned} \tilde{x} &= x_c + L(r)(x_d - x_c) \\ \tilde{y} &= y_c + L(r)(y_d - y_c) \end{aligned} \tag{3.12}$$

where $(\tilde{x},\tilde{y})$ are the corrected coordinates which are related to the 3-D point by a linear projective camera and (x_c,y_c) is the centre of radial distortion, with $r^2 = (x_d - x_c)^2 + (y_d - y_c)^2$. An approximation to an arbitrary function $L(r)$ may be given by a Taylor expansion $L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \ldots$. In practice, estimating only the first distortion coefficient, κ_1, is sufficient for an accurate calibration [169].

The distortion function may be computed together with the camera matrix $\tilde{P}$ during the iterative minimisation of the geometric error described earlier. A more general approach is to determine $L(r)$ by the requirement that images of straight lines must also be straight and iteratively minimise a cost function over the parameters κ_i of the distortion function and the centre of distortion (x_c,y_c) [70].

Limitations of the Conventional Camera Calibration

The Euclidean 3-D reference structure might not always be available, *e.g.*, when no images of a calibration pattern with known 3-D coordinates could be taken at the time of acquisition. In such cases, the conventional calibration method described in this section cannot be used. Another example is a setup consisting of multiple cameras. The conventional method does not extend very well to this case either, as it requires that the calibration pattern be visible in all images simultaneously. Reasons like these triggered research on different calibration methods which do not require an advance knowledge of the 3-D calibration pattern but instead use image information alone to calibrate the cameras, known as *self-calibration* methods. We describe a self-calibration method for multiple cameras in the next section.

3.2.3 Multi-Camera Self Calibration

Self-calibration (also called *auto-calibration*) is the process of determining camera parameters directly from multiple uncalibrated images [70]. As opposed to the conventional calibration (see Section 3.2.2), where the camera projection matrix P is determined from the image of a known calibration object, in self-calibration the metric properties are determined directly from the constraints on the internal and external camera parameters and no specific knowledge about the scene is used in the process.

The general approach to self-calibration can be summarised with the following three steps:

1. Build a measurement matrix W consisting of images of 3-D points in all views.
2. Obtain a projective reconstruction $\{P, X\}$, where $W = PX$.
3. Determine a rectifying homography H from self-calibration constraints and transform the projective reconstruction to a metric one $\{PH, H^{-1}X\}$, where $W = PHH^{-1}X$.

To calibrate our camera setup, we use the multi-camera self-calibration method by Svoboda *et al.* [159] which works by acquiring an image sequence of a moving bright object in a dark room. The three steps described above are as follows.

In step 1, the acquired sequence is processed and image locations of the bright spot extracted in all the views for all frames of the sequence. In step 2, the image measurements are collected in a measurement matrix W, which is then factorised into the projective motion P and structure X. In step 3, the metric reconstruction based on the concept of the absolute conic is used to compute the rectifying homography and convert the projective estimates to Euclidean structure and motion, thereby completing the camera calibration.

In the remainder of this section, we describe the projective reconstruction algorithm (step 2) in more detail. In Appendix B, we provide the derivation of the rectifying homography (step 3) used in [159], which we extended to explicitly show all the steps of the derivation.

Projective Structure and Motion

Consider a set of n 3-D points $\mathbf{X}_j = [X_j, Y_j, Z_j, 1]^\top$, $j = 1, ..., n$ viewed by m cameras with projection matrices P^i, $i = 1, ..., m$. If each point is visible in every view, we can write the following relationship [70]:

$$W = PX \tag{3.13}$$

or, equivalently:

$$\begin{bmatrix} \lambda_1^1 \mathbf{x}_1^1 & \lambda_2^1 \mathbf{x}_2^1 & \dots & \lambda_n^1 \mathbf{x}_n^1 \\ \lambda_1^2 \mathbf{x}_1^2 & \lambda_2^2 \mathbf{x}_2^2 & \dots & \lambda_n^2 \mathbf{x}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1^m \mathbf{x}_1^m & \lambda_2^m \mathbf{x}_2^m & \dots & \lambda_n^m \mathbf{x}_n^m \end{bmatrix} = \begin{bmatrix} P^1 \\ P^2 \\ \vdots \\ P^m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1, & \mathbf{X}_2, & \dots, & \mathbf{X}_n \end{bmatrix}, \tag{3.14}$$

where $\mathbf{x}_j^i$ is the image point representing the j-th $3D$ point, $\mathbf{X}_j$, as seen by the i-th camera, P^i.

Equation (3.14) is valid if all the weighting factors λ_j^i for each of the measured points $\mathbf{x}_j^i$ are known and the image measurements are not corrupted by noise. The matrix W then has a rank 4 and can be decomposed via the singular value decomposition into projective motion P and projective shape X. The factorization $W = PX$ recovers the projective motion and projective shape up to a 4×4 projective transformation H:

$$W = PX = PHH^{-1}X = \hat{P}\hat{X}, \tag{3.15}$$

where $\hat{P} = PH$ and $\hat{X} = H^{-1}X$ [70].

Projective Depths. The projective depths λ_p^i of a point $\mathbf{X}_p$ can be estimated using the knowledge of the epipolar geometry for pairs of views and combining together pairwise estimates of depth obtained from the epipoles and the fundamental matrix [157]:

$$\lambda_p^i = \frac{(\mathbf{e}^{ik} \times \mathbf{x}_p^i) \cdot (F^{ik} \mathbf{x}_p^k)}{\|\mathbf{e}^{ik} \times \mathbf{x}_p^i\|^2} \lambda_p^k, \tag{3.16}$$

where F^{ik} is the fundamental matrix relating the views i and k, e^{ik} is the epipole, *i.e.*, the image of the camera centre k in view i, $\mathbf{x}_p^i$ is the image of the point $\mathbf{X}_p$ as seen by camera P^i and $\mathbf{x}_p^k$ is the image of the point $\mathbf{X}_p$ as seen by camera P^k. Such equations can be recursively

chained together to give estimates for the complete set of depths for the point $\mathbf{X}_p$, starting from some arbitrary initial value such as $\lambda_p^1 = 1$ [157].

Missing Points. Occlusions are a common problem in a multi-camera setup and mean that not all points are simultaneously visible in all cameras. When occlusions are present, the measurement matrix W is incomplete which complicates the projective factorisation. Due to the recursive nature of the projective depth estimation algorithm, missing points also mean that the computation of projective depths for some of the detected points is not possible.

For the factorisation in (3.14) to work, the measurement matrix must be complete. The completion can be achieved by fitting an unknown complete $3m \times n$ matrix $\tilde{W}$ of rank 4 to the incomplete measurement matrix W. Martinec and Pajdla [104] achieve this by expanding the incomplete matrix W with zeroes and standard bases for every missing image point or projective depth, effectively "filling" the matrix W, so that the span of the expanded matrix includes the span of $\tilde{W}$. A detailed explanation of this procedure is given in [104].

Rectifying Homography for Metric Reconstruction

The last step in the multi-camera self-calibration method by Svoboda *et al.* [159], the computation of the rectifying homography for metric reconstruction, is given in Appendix B. We have extended the derivation to show the step-by-step construction of the homography.

3.2.4 Conclusion

In this section, we described the camera calibration algorithm used to calibrate our multi-view camera setup. In the next section, we turn our attention to image rectification, which allows for an efficient computation of disparity maps and a definition of a three-dimensional disparity space, discussed in detail later in Chapter 7.

3.3 Image Rectification

3.3.1 Introduction

Let us assume a stereo pair of cameras with projection matrices P_l and P_r, for the left and right camera, respectively. In the imaging process, given a 3-D point $\mathbf{X}$ visible in both cameras, we get two different projections of $\mathbf{X}$, one in the left camera image plane, $\mathbf{x}_l$, and one in the right camera image plane, $\mathbf{x}_r$:

$$\mathbf{x}_l \simeq P_l \mathbf{X}, \tag{3.17}$$

$$\mathbf{x}_r \simeq P_r \mathbf{X}. \tag{3.18}$$

In a stereo reconstruction problem, the task is inverted. Given a stereo pair of images, we start from a pair of image points, $\mathbf{x}_l$ and $\mathbf{x}_r$, which are the projections of the same 3-D point $\mathbf{X}$, whilst the exact location of $\mathbf{X}$ is unknown. These two points are called the *corresponding points*. The knowledge of two such points and the camera calibration information allow us to reconstruct the position of the 3-D point $\mathbf{X}$.

Searching for pairs of corresponding points in a stereo pair of images is a difficult problem which, in the most general case, when nothing is known about the camera setup or epipolar geometry of the stereo pair, requires a full 2-D image search to find an image point $\mathbf{x}_r$ corresponding to a particular point $\mathbf{x}_l$.

Knowledge of the *epipolar geometry* of the stereo setup, *i.e.*, the fundamental matrix F relating the two views, allows us to reduce the search to a line in 2-D. The *epipolar constraint* states that, for a given image point $\mathbf{x}_l$, its corresponding point $\mathbf{x}_r$ must lie on the *epipolar line* $\mathbf{l}_l$, determined by the point $\mathbf{x}_l$. Epipolar line $\mathbf{l}_l$ is a projection, in the right image plane, of the *optical ray* connecting the left camera centre $\mathbf{c}_l$ and the image point $\mathbf{x}_l$ and can be algebraically computed using the fundamental matrix F as follows:

$$\mathbf{l}_l = F\mathbf{x}_l. \tag{3.19}$$

As the corresponding right image point lies on the epipolar line $\mathbf{l}_l$, the relationship between the two corresponding points, given the knowledge of the fundamental matrix F, can be

written as:

$$\mathbf{x}_r^\top F \mathbf{x}_l = 0. \tag{3.20}$$

The search for image correspondences can be even further constrained by transforming the images so that the epipolar lines coincide with the horizontal scanlines and are collinear. In this case, the search for the corresponding point reduces to a 1-D search for the corresponding offset along the appropriate horizontal scanline. Such a pair of images, where the epipolar lines are collinear and parallel to the image x-axis, is said to be *rectified*.

Image rectification not only simplifies the correspondence search but also transforms the original 2-D disparity vectors to 1-D scalar values. This is important for the definition of a 3-D disparity space, which we discuss in Chapter 7.

For the purpose of this thesis, we use the rectification algorithm by Fusiello *et al.* [61], which assumes a calibrated stereo pair of cameras and provides a compact derivation of rectified camera matrices and corresponding image transformations for a general, unconstrained stereo rig. We now describe this algorithm in more detail.

3.3.2 Rectification algorithm

The rectification algorithm assumes that the cameras are calibrated, *i.e.*, that the two projection matrices, P_l^o and P_r^o are known, where the index o denotes the original camera matrices. The idea behind rectification is to define two new projection matrices, P_l^n and P_r^n, obtained by rotating the old camera matrices around their optical centres of projection, until the focal planes become coplanar [61].

Camera Matrix Rectification

In order for the epipolar lines to become horizontal and collinear, the baseline must be parallel to the new x-axis of both cameras and the new cameras must have the same intrinsic parameters. The two new projection matrices will differ from each other only in the position of their optical centre and can be thought of as a single camera translated along the x-axis of its reference coordinate system [61].

Using QR factorisation [153], the old camera projection matrices factorise into:

$$P_l^o = K_l^o[R_l^o|\mathbf{t}_l^o], \tag{3.21}$$

$$P_r^o = K_r^o[R_r^o|\mathbf{t}_r^o], \tag{3.22}$$

where K is the intrinsic parameters matrix, R is the rotation matrix denoting the rotation of the camera relative to the world coordinate system, and $\mathbf{t}$ is the translation vector denoting the translation of the camera centre from the origin of the world coordinate system.

To show how the new, rectified camera matrices are derived from the old ones, let us first write the new matrices in their factorised form:

$$P_l^n = K[R|-R\mathbf{c}_l], \tag{3.23}$$

$$P_r^n = K[R|-R\mathbf{c}_r]. \tag{3.24}$$

The intrinsic parameters matrix K is assumed to be the same for both projection matrices and can be chosen arbitrarily, *e.g.*, $K = K_l^o$. The optical centres $\mathbf{c}_l$ and $\mathbf{c}_r$ coincide with the optical centres of the old projection matrices and can be computed from the old matrices using

$$\mathbf{c} = -R^{o\top}\mathbf{t}^o, \tag{3.25}$$

where $^\top$ denotes a matrix transpose.

The new rotation matrix R is the same for both projection matrices and is defined in terms of its row vectors:

$$R = \begin{bmatrix} \mathbf{r}_x^\top \\ \mathbf{r}_y^\top \\ \mathbf{r}_z^\top \end{bmatrix}, \tag{3.26}$$

which are the x-, y- and z-axis of the new camera reference frame, expressed in the world coordinate system. The individual row vectors of the new rotation matrix are computed as follows:

1. The new x-axis is parallel to the baseline: $\mathbf{r}_x = (\mathbf{c}_l - \mathbf{c}_r)/\|\mathbf{c}_l - \mathbf{c}_r\|$.

2. The new y-axis is orthogonal to $\mathbf{r}_x$ and an auxiliary vector $\mathbf{k}$: $\mathbf{r}_y = \mathbf{k} \times \mathbf{r}_x$.

3. The new z-axis is orthogonal to $\mathbf{r}_x$ and $\mathbf{r}_y$: $\mathbf{r}_z = \mathbf{r}_x \times \mathbf{r}_y$.

The auxiliary vector $\mathbf{k}$ is a unit vector and can be set to the old z-axis of the left projection matrix, for example.

Rectifying image transformation

Once the camera matrices have been rectified, they can then be used to compute the appropriate rectifying image transformations.

Let us write the old and new projection matrices as $P^o = [Q^o|q^o]$ and $P^n = [Q^n|q^n]$, respectively, where Q is the 3×3 submatrix of P. The left rectifying image transformation is derived as follows. For any 3-D point $\mathbf{X}$ we have:

$$\mathbf{x}_l^o \simeq P_l^o \mathbf{X} \tag{3.27}$$

$$\mathbf{x}_l^n \simeq P_l^n \mathbf{X} \tag{3.28}$$

The equations of the optical rays back-projected through the image points $\mathbf{x}_l^o$ and $\mathbf{x}_l^n$ are:

$$\mathbf{X}^o = \mathbf{c}_l + \lambda^o (Q_l^o)^{-1} \mathbf{x}_l^o, \quad \lambda^o \in \mathbb{R} \tag{3.29}$$

$$\mathbf{X}^n = \mathbf{c}_l + \lambda^n (Q_l^n)^{-1} \mathbf{x}_l^n, \quad \lambda^n \in \mathbb{R}, \tag{3.30}$$

and by setting $\mathbf{X}^o = \mathbf{X}^n$, we get:

$$\mathbf{x}_l^n = \lambda Q_l^n (Q_l^o)^{-1} \mathbf{x}_l^o, \quad \lambda \in \mathbb{R} \tag{3.31}$$

The transformation $T_l = Q_l^n (Q_l^o)^{-1}$ is applied to the pixels in the original left image to produce the rectified left image. The derivation of the right rectifying image transformation T_r follows the same algorithm. The full rectification algorithm is shown in Algorithm 3.1.

When rectifying the images by applying the transformations T_l and T_r, the calculated positions of the rectified pixels in general correspond to non-integer positions in the image and

the grayscale values of the integer-position pixels must be computed using bilinear interpolation.

Algorithm 3.1: Rectification Algorithm [61].

Input: Stereo pair of images, I_l and I_r, and the corresponding camera projection matrices, $P_l^o = K_l^o[R_l^o|\mathbf{t}_l^o]$ and $P_r^o = K_r^o[R_r^o|\mathbf{t}_r^o]$.

- Set the new intrinsic matrix to $K = K_l^o$.
- Compute the optical centres $\mathbf{c}_l = -R_l^{o\top}\mathbf{t_l}^o$ and $\mathbf{c}_r = -R_r^{o\top}\mathbf{t_r}^o$.
- Compute the new rotation matrix R:

$$R = \begin{bmatrix} \mathbf{r}_x^\top \\ \mathbf{r}_y^\top \\ \mathbf{r}_z^\top \end{bmatrix}, \tag{3.32}$$

  where

  1. $\mathbf{r}_x = (\mathbf{c}_l - \mathbf{c}_r)/\|\mathbf{c}_l - \mathbf{c}_r\|$.
  2. $\mathbf{r}_y = \mathbf{k} \times \mathbf{r}_x$.
  3. $\mathbf{r}_z = \mathbf{r}_x \times \mathbf{r}_y$.

- Compute the new, rectified matrices: $P_l^n = K[R|-Rc_l] = [Q_l^n|q_l^n]$ and $P_r^n = K[R|-Rc_r] = [Q_r^n|q_r^n]$.
- Compute the rectifying image transformations: $T_l = Q_l^n(Q_l^o)^{-1}$ and $T_r = Q_r^n(Q_r^o)^{-1}$.
- Apply the rectifying image transformations, T_l and T_r, to the original images, I_l and I_r.

Output: Rectified images, I_l^n and I_r^n.

3.3.3 Conclusion

As already mentioned, the image rectification simplifies the correspondence search to a 1-D search problem. Once the images have been rectified, a correspondence search algorithm is then run to produce an ordered set of pairs of corresponding pixels, called a *disparity map*. If the corresponding pairs are known for every pixel position in the image, the disparity maps are referred to as *dense* disparity maps. We explain more about disparity maps and correspondence search in the next section.

3.4 Disparity Maps

3.4.1 Introduction

Disparity information provides an important geometric constraint in the image-based novel view synthesis. It allows for the novel views to be synthesised from a relatively small number of images as it contains implicit information about the depth of the scene. In this section, we first explain the concept of disparity and disparity map. We then proceed to describe the pyramidal correspondence search algorithm for disparity map generation, which we used for the purpose of research described in this thesis.

3.4.2 Disparity and Disparity Map

Let us assume that a 3-D point $\mathbf{X} = (X, Y, Z, 1)^\top$ is viewed by two distinct cameras, left camera with the projection matrix P_l and right camera with the projection matrix P_r, and that the image of the point $\mathbf{X}$ is defined as $\mathbf{x}_l = (x_l, y_l, 1)^\top \simeq P_l\mathbf{X}$ and $\mathbf{x}_r = (x_r, y_r, 1)^\top \simeq P_r\mathbf{X}$ in the left and right camera's image plane, respectively, where $\simeq$ denotes equality up to a scale factor.

The corresponding points $\mathbf{x}_l$ and $\mathbf{x}_r$ are related by a *disparity* $d(\mathbf{x}_l, \mathbf{x}_r)$:

$$d(\mathbf{x}_l, \mathbf{x}_r) = \mathbf{x}_l - \mathbf{x}_r = (x_l - x_r, y_l - y_r), \tag{3.33}$$

so that

$$\mathbf{x}_r = \mathbf{x}_l - d(\mathbf{x}_l, \mathbf{x}_r) = (x_l - (x_l - x_r), y_l - (y_l - y_r)) = (x_r, y_r) \tag{3.34}$$

Given a stereo pair of images, I_l and I_r, a *(dense) disparity map* is a collection of disparity values, one for every pixel in the stereo pair.

Disparity map is usually defined with respect to one of the images. For example, the *left-to-right* disparity map contains values $D_l(i_1, j_1) = d(I_l(i_1, j_1), I_r(i_2, j_2))$ while the *right-to-left* disparity map contains values $D_r(i_2, j_2) = d(I_r(i_2, j_2), I_l(i_1, j_1))$, where $I_l(i_1, j_1)$ and $I_r(i_2, j_2)$ is a corresponding pair of image pixels from the left and right image, respectively.

3.4.3 Image Pyramid

An image pyramid is a collection of decreasing-resolution images arranged in the shape of a pyramid. The base of the pyramid contains the original image and the apex contains the lowest-resolution approximation. The image pyramid was introduced by Burt and Adelson[1, 27] who described an image coding algorithm using a Laplacian pyramid.

Let us denote the original image as I_0 with R_0 rows and C_0 columns. This image represents the level 0 of the pyramid. Level 1 consists of the image I_1, which is a filtered and subsampled version of I_0. In general, a pyramid level l consists of the image I_l, which is a filtered and subsampled version of I_{l-1}.

The number of levels in the pyramid is limited by the size of the original image. For example, if the original image dimensions are $R_0 = C_0 = 2^N$, then a fully populated image pyramid will have $N + 1$ levels, *i.e.*, the original image at level 0 and its N filtered and subsampled versions. Most pyramids are truncated to only $L + 1$ levels, where $L < N$, as a 1×1 (single pixel) approximation of the original image is of little value [64].

The Laplacian Pyramid Image Code [27] introduced two different types of image pyramids [64]: the *approximation pyramids* (*i.e.*, the *Gaussian Pyramid* in [27]) and the *prediction residual pyramids* (*i.e.*, the *Laplacian Pyramid* in [27]).

The approximation pyramids are constructed by filtering and downsampling the original image in an iterative manner. Depending on the chosen filtering operation, the ap-

proximation pyramid can be a *mean pyramid* (using neighbourhood averaging), *Gaussian pyramid* (using low-pass Gaussian filtering), or even a *subsampling pyramid* (using no filter at all) [64].

There are three standard applications of the Gaussian approximation pyramids to image processing [60]. *Search over scale* convolves a constant size template with the image on different levels of the pyramid to find variably sized occurrences of a particular pattern. *Stereo correspondence search* looks for an initial set of correspondences on the highest level of the pyramid and proceeds by propagating these onto the next lower level to initialise a more detailed correspondence search. *Feature tracking* identifies features on the lowest level of the pyramid and then traces them up to the highest level. Only features with clearly identifiable "parents" on the highest pyramid level are accepted and others discarded as noise.

The prediction residual pyramids are constructed from the approximation pyramids. Each level l of the prediction residual pyramid is constructed by upsampling and then filtering the level $l-1$ of the corresponding approximation pyramid, which creates the *prediction image* on level l. The difference between the prediction image and the actual image of the approximation pyramid on level l is then computed, which generates the *prediction residual* [27].

Using the highest level of the approximation pyramid together with the prediction residuals on all levels enables a complete reconstruction of the approximation pyramid and thus the original input image. As the prediction residual images can be highly compressed, this technique of image reconstruction represents a form of image coding [64].

3.4.4 Burt and Adelson's Gaussian Image Pyramid

For levels $0 < l < L$ and pixels i, j, $0 \leq i \leq R_l$ and $0 \leq j \leq C_l$, the original image approximation on level l of the Gaussian approximation pyramid is computed from the approximation

on level $l-1$ according to the following equation [27, 1]:

$$I_l(i,j) = \sum_{u=-m}^{m} \sum_{v=-n}^{n} w(u,v) I_{l-1}(2i+u, 2j+v) \tag{3.35}$$

where $w(2m+1, 2n+1)$ is the weighting window, and R_l and C_l are the row and column dimensions of the original image approximation on pyramid level l.

Generating Kernel. The $(2m+1)\times(2n+1)$ pattern of weights w which is used to generate each level of the pyramid from its preceding level is called the *generating kernel*. The kernel weights are chosen subject to four constraints which will be illustrated on a kernel size 5×5 ($m = n = 2$).

1. For computational convenience, the generating kernel is separable:

$$w(2m+1, 2n+1) = \hat{w}(2m+1)\hat{w}(2n+1), \tag{3.36}$$

2. The one-dimensional function $\hat{w}$ is symmetric:

$$\hat{w}(0) = a, \quad \hat{w}(-1) = \hat{w}(1) = b, \quad \hat{w}(-2) = \hat{w}(2) = c. \tag{3.37}$$

3. $\hat{w}$ is normalised:

$$a + 2b + 2c = 1 \tag{3.38}$$

4. Each pixel on level l must contribute the same total weight to pixels on level $l+1$:

$$a + 2c = 2b. \tag{3.39}$$

The last property is illustrated in Figure 3.2, extended from [27].

Iterative pyramid generation is equivalent to convolving the image I_0 with a set of equivalent weighting functions h_l:

$$I_l = h_l * I_0 \tag{3.40}$$

or

$$I_l(i,j) = \sum_{u=-m_l}^{m_l} \sum_{v=-n_l}^{n_l} h_l(u,v) I_0(2^l i + u, 2^l j + v), \tag{3.41}$$

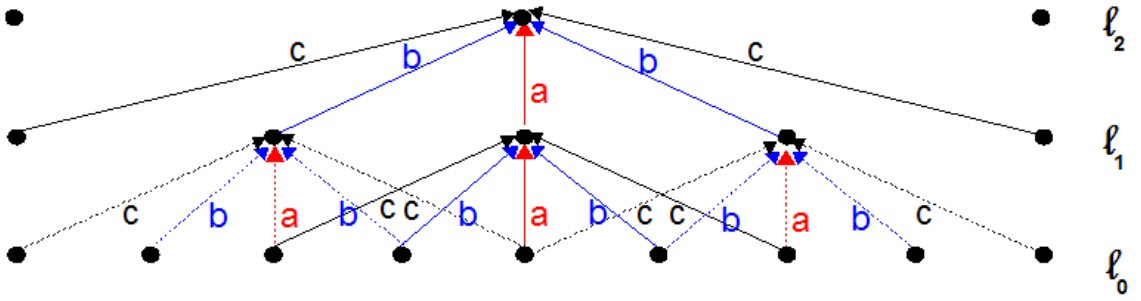


Figure 3.2: A one-dimensional graphic representation of the process which generates a Gaussian pyramid. Figure has been extended from [27]. Each row of dots represents nodes (pixels) within a level of the pyramid.

where the subscript l denotes the pyramid level, $0 \leq l \leq L$. The size of the equivalent weighting function and the distance between the image samples doubles from one level to the next while the shape of the function remains the same and depends on the value of parameter a in the generating kernel. For example, if $a = 0.4$, the functions resemble the Gaussian probability density function. Pyramid construction with a corresponding generating kernel is then equivalent to convolving the original image with a set of Gaussian-like functions to produce a corresponding set of filtered images. Example illustrations of equivalent weighting functions h_l are given in [27].

3.4.5 Correlation-Based Stereo Correspondence

Correlation-based stereo correspondence search looks for intensity matches between the stereo pair of images. The goal is a pixel-wise dense disparity map. Ideally, the disparity information would be known for every pixel in the left and right image. In practice however, not all regions are equally visible in both images due to the parallax effect arising from the change in camera viewpoint between the left and the right image. Consequently, some of the pixels cannot have a disparity value assigned.

The correspondence search algorithm traverses the left image pixel by pixel. A rectangular correlation window is centred on the current pixel and an equivalent-size search window is moved around a search region in the right image. For every displacement of the search window in the right image, the corresponding (dis)similarity function is computed

based on the image intensity values contained in the window. The displacement value which gives the highest similarity score (or lowest dissimilarity score) is recorded as the disparity value of the pixel in the left image.

Let $\mathbf{x}_l$ and $\mathbf{x}_r$ be pixels in the left and right image, $2W+1$ the width (in pixels) of the correlation window, $S(\mathbf{x}_l)$ the search region in the right image associated with $\mathbf{x}_l$ and $\psi(u,v)$ a (dis)similarity function of two pixel values, u, v. Algorithm 3.2 details the matching process [167].

Algorithm 3.2: Stereo Correspondence Search Algorithm [167].

Input: Stereo pair of images, $I_l(R,C)$ and $I_r(R,C)$, with R and C the number of rows and columns in each image, respectively.

Loop:

For each pixel $\mathbf{x}_l = [i, j]^\top$ in the left image:

1. For each displacement $\mathbf{d} = [d_1, d_2]^\top \in S(\mathbf{x}_l)$ compute

$$c(\mathbf{d}) = \sum_{k=-W}^{W} \sum_{l=-W}^{W} \psi(I_l(i+k, j+l), I_r(i+k-d_1, j+l-d_2)). \quad (3.42)$$

2. The disparity of $\mathbf{x}_l$ is the vector $\bar{\mathbf{d}} = [\bar{d_1}, \bar{d_2}]^\top$ that maximizes $c(\mathbf{d})$ over $S(\mathbf{x}_l)$:

$$\bar{\mathbf{d}} = arg \max_{d \in S} \{c(\mathbf{d})\}. \quad (3.43)$$

Output: Dense correspondence map for the input images $I_l(R,C)$ and $I_r(R,C)$.

The (dis)similarity function $\psi = \psi(u,v)$ can have various forms. Setting:

$$\psi(u,v) = uv \quad (3.44)$$

yields a so-called *simple cross-correlation* [129] between the window in the left image and the search region in the right image. Examples of other possible functions are (normalised, zero-mean) cross-correlation, (normalised, zero-mean) sum of squared differences, (zero-mean) sum of absolute differences. Which similarity function is best to use depends on the computational requirements and image contents [129].

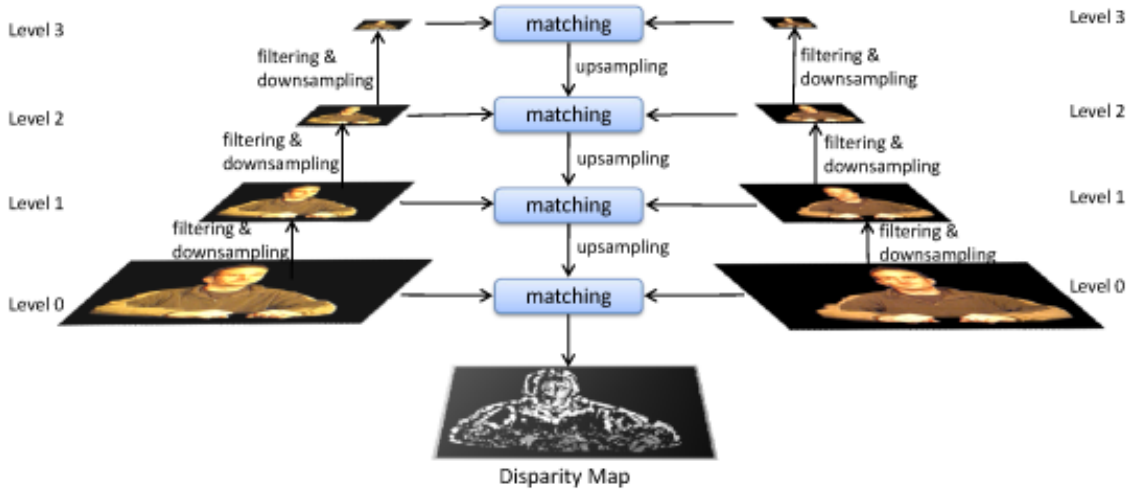


Figure 3.3: Illustration of the pyramidal stereo-correspondence search algorithm.

3.4.6 Pyramidal Stereo Correspondence Algorithm

The pyramidal stereo-correspondence search algorithm begins the correspondence search on the highest level of the pyramid. Once the correspondences on the highest level are determined, they are transferred to the next lower level of the pyramid and used to initialise the correspondence search on that level. This process continues until the correspondences are found on level 0 of the pyramid for the full-resolution original image (see Figure 3.3 and Algorithm 3.3).

Advantages. In wide-baseline stereo, the disparity search interval is expected to be large, which implies that the correspondence search starting on the full resolution original image would have to traverse a significant portion of the image in the search for the right match. Using an image pyramid reduces the computational complexity by initialising the search on a lower resolution image where only a subset of features are matched and propagating the results downwards through the pyramid, while at the same time limiting the search region that needs to be covered on a particular resolution level. Search window size remains the same throughout the pyramid and the search region is much more limited than if the search had to be performed on the original image directly, without subsampling. As such, pyramidal stereo correspondence search presents a good choice for a wide-baseline disparity search algorithm.

Algorithm 3.3: Pyramidal Stereo Correspondence Search Algorithm, based on [167, 20].

Input: left image I_l, right image I_r.

Initialisation:

- Create two Gaussian image pyramids with $K+1$ levels, one for I_l and one for I_r, where $2 \leq K+1 \leq N+1$ and $N+1$ is the highest possible number of pyramid levels.
- Start on level $k = K$.
- For each pixel $I_l^K(i,j)$ in the left image, find a corresponding pixel $I_r^K(i-d_1, j-d_2)$, such that $\psi(I_l^K(i,j), I_r^K(i-d_1, j-d_2))$ is maximal if ψ is a similarity function, or minimal if ψ is a dissimilarity function.

Loop:

repeat

- Descend to the next pyramid level, $k = k-1$.
- Upsample the correspondence information from the one found on level $k+1$ and use it to initialise the correspondence search.
- For each pixel $I_l^k(i,j)$ in the left image, find a corresponding pixel $I_r^k(i-d_1, j-d_2)$, such that $\psi(I_l^k(i,j), I_r^k(i-d_1, j-d_2))$ is maximal if ψ is a similarity function or minimal if ψ is a dissimilarity function.

until *l=0*

Output: Dense correspondence map for the input images I_l and I_r.

Disadvantages. If the highest level of the pyramid is too coarse, incorrect matches can be found due to aliasing [20]. These are then propagated downwards through the pyramid to the lowest level and due to the constrained search region cannot be corrected once the discriminating intensity information becomes available. The choice of the number of pyramid levels to use depends on the image size and contents and is usually determined through

experimental trials.

Although performing the matching on high levels of the pyramid reduces the computational complexity of the search, it is important not to set the number of pyramid levels too high as in practice very high pyramid levels do not contain enough information for an unambiguous correspondence search. For example, if there are several instances of the same shape distributed randomly around an image, the pyramidal approach will be less useful as the background information which would be used on the original resolution level to differentiate between the individual instances is not present on the highest pyramid level due to subsampling. According to Roma *et al.* [129] better results can be achieved with fewer hierarchical levels because of distortions and feature loss when the image resolution is too coarse, as this gives rise to a critical loss of important reference points, essential to perform the matching of image regions.

3.4.7 Conclusion

Once the correspondences in a stereo pair of images have been found, this information can be used to reconstruct the 3-D positions of the points they represent. In the next section, we describe the 3-D reconstruction algorithm that we use for this purpose.

3.5 3-D Reconstruction

3.5.1 Introduction

When corresponding pairs of image points are available for a stereo pair of views, the corresponding 3-D points can be reconstructed as well. The nature of the reconstruction depends on the amount of information that is available about the stereo setup.

If no knowledge about the cameras is available, the reconstruction can be obtained up to an unknown projective transformation. If only the intrinsic parameters of the stereo system are known, the reconstruction and the extrinsic parameters can be estimated up to an unknown scaling factor. If the stereo setup is fully calibrated, *i.e.*, the intrinsic and the extrinsic parameters are known, the reconstruction problem can be solved unambiguously by

triangulation [167].

Algorithm 3.4: 3-D Reconstruction by Triangulation [167]

Input: Left-to-right consistency-checked disparity map, D_{lr}, for a stereo pair of images I_l and I_r.

Loop:

For every pair of corresponding points $\mathbf{x}_l$ and $\mathbf{x}_r = \mathbf{x}_l - D_{lr}(\mathbf{x}_l)$:

- Let the ray l through the left centre of projection, O_l, and the left image point, $\mathbf{x}_l$, be $a\mathbf{x}_l$, where $a \in \mathbb{R}$.
- Let the ray r through the right centre of projection, O_r, and the right image point, $\mathbf{x}_r$, expressed in the left centre of projection's reference frame, be: $\mathbf{t} + bR^\top \mathbf{x}_r$, where $b \in \mathbb{R}$
- Let $\mathbf{w}$, a vector orthogonal to both l and r, be $\mathbf{w} = \mathbf{x}_l \times R^\top \mathbf{x}_r$.
- The endpoints of the segment, $a_0\mathbf{x}_l$ and $\mathbf{t} + b_0 R^\top \mathbf{x}_r$, can be computed by solving the following linear system of equations:

$$a\mathbf{x}_l - bR^\top \mathbf{x}_r + c(\mathbf{x}_l \times R^\top \mathbf{x}_r) = \mathbf{t}, \tag{3.45}$$

  for a_0, b_0 and c_0, where R is the rotation matrix denoting the rotation of the second camera relative to the first camera and the vector $\mathbf{t}$ is the translation vector denoting the displacement of the second camera with respect to the first camera.
- The point X is approximated by the midpoint X' of the segment joining $a_0\mathbf{x}_l$ and $\mathbf{t} + b_0 R^\top \mathbf{x}_r$.

Output: Reconstructed cloud of 3-D points $\{X\}$ for all pairs of corresponding points $\{(\mathbf{x}_l, \mathbf{x}_r)\}$.

Our camera setup is fully calibrated and we describe the triangulation algorithm that we use in the next section.

3.5.2 Reconstruction by Triangulation

The triangulation reconstruction method described in this section is taken from Trucco and Verri [167]. This method requires the cameras to be fully calibrated, *i.e.*, the intrinsic and extrinsic parameters of the stereo system must be known.

Let a 3-D point X be imaged as $\mathbf{x}_l$ and $\mathbf{x}_r$ in a stereo pair of images. The point X can be reconstructed from its two images, $\mathbf{x}_l$ and $\mathbf{x}_r$, by finding the intersection of rays from O_l through $\mathbf{x}_l$ and from O_r through $\mathbf{x}_r$, where O_l and O_r are the left and right centre of projection, respectively. Because of image discretisation, the two rays will not intersect exactly. Instead, the 3-D point is reconstructed as the point which minimises the distance from both rays. Algorithm 3.4 contains the detailed description of the steps required.

3.5.3 Conclusion

With this section we conclude the first part of the background theory review, where we described the methods necessary to generate disparity and 3-D data for use in our disparity map completion algorithm. In the second part, we turn our attention to methods which enable us to model the shape and articulated motion of a human body, the model of which we use to complete the patchy disparity data for better quality novel-view synthesis results.

3.6 Subdivision Surfaces

3.6.1 Introduction

In the second part of the background theory review, we look at the methods that we use for modelling and manipulating an articulated model of a human body. In this section, we begin by describing the theory of subdivision surfaces and their properties, which make them a good choice for 3-D modelling. We then show the relationship between the spline

curves and subdivision, followed by description of the subdivision schemes, in particular the Catmull-Clark scheme, used in our human upper-body model.

3.6.2 Subdivision Surface Modelling

In 1978, Catmull and Clark [32] and Doo and Sabin [52] published a paper in the same issue of the Computer-Aided Design, describing the *recursively generated b-spline surfaces* and *recursive division surfaces*, respectively. Their work marked the beginning of the use of subdivision for geometric surface modelling. Since then, the subdivision surfaces have become a popular modelling tool in movie production, commercial modellers, game engines, and more recently also found their way into wide application in computer graphics and computer assisted geometric design [186].

Modelling surfaces using subdivision conveniently addresses many important issues in the computer-aided geometric design. According to Zorin *et al.* [186], the strengths of subdivision modelling are:

- **Arbitrary topology:** Subdivision generalises classical spline patch approaches to an arbitrary topology. There is no need for awkward constraint management between patches.
- **Scalability:** Because of its recursive structure, subdivision naturally accommodates level-of-detail rendering and adaptive approximation with error bounds.
- **Uniformity of representation:** Much of traditional modeling uses either polygonal meshes or spline patches. Subdivision spans the spectrum between these two extremes. Surfaces can behave as if they are made of patches, or they can be treated as if consisting of many small polygons.
- **Numerical stability:** The meshes produced by subdivision have many of the nice properties finite element solvers require, which makes the subdivision representations suitable for many numerical simulation tasks of importance in engineering and computer animation settings.

- **Code simplicity:** Last but not least, the basic ideas behind subdivision are simple to implement and execute very efficiently.

Several different approaches for smooth surface modelling exist, *e.g.*, splines, implicit surfaces, variational surfaces. Each approach has its own strengths which should be considered when choosing an appropriate modelling tool. Zorin *et al.* [186] give a comparative description of the mentioned modelling approaches, which we summarise here:

- **Efficiency:** Subdivision is easy to implement and computationally efficient. Only a small number of neighbouring old points are used in the computation of the new points. Implicit surfaces are much more costly. An algorithm such as marching cubes is required to generate the polygonal approximation needed for rendering. Variational surfaces can be even worse: a global optimization problem has to be solved each time the surface is changed.

- **Arbitrary topology:** Implicit surfaces can be of arbitrary topological genus, but the precise location, and connectivity of a surface are typically difficult to control. Variational surfaces can handle arbitrary topology better than any other representation, but the computational cost can be high. Subdivision can handle arbitrary topology quite well without losing efficiency; this is one of its key advantages.

- **Surface features:** Variational surfaces provide the most flexibility and exact control for creating features. Implicit surfaces are very difficult to control, since all modeling is performed indirectly and there is much potential for undesirable interactions between different parts of the surface. Spline surfaces allow very precise control, but it is computationally expensive and awkward to incorporate features. Subdivision allows more flexible controls than is possible with splines. In addition to choosing locations of control points, one can manipulate the coefficients of subdivision to achieve effects such as sharp creases or control the behavior of the boundary curves.

- **Complex geometry:** Because subdivision is based on repeated refinement, it is very straightforward to incorporate ideas such as level-of-detail rendering and compression for the internet. During interactive editing locally adaptive subdivision can generate just enough refinement based on geometric criteria, for example. For applica-

tions that only require the visualisation of fixed geometry, other representations, such as progressive meshes, are likely to be more suitable.

As the subdivision surface theory is based on the spline theory, we first present a short introduction to the spline curves in the next section. In the subsequent section, we then show the connection between spline curves and subdivision.

3.6.3 Cubic B-Splines

Let $C(u)$ be a parametric curve defined on the interval $u_{min} \leq u < u_{max}$ and let us split the parameter domain into individual segments separated by $m+5$ special parameter values u_i, called *knots*, so that [22]:

$$u_0 < u_1 < u_2 < u_3 = u_{min}$$

$$u_{min} = u_3 < u_4 < \cdots < u_{m+1} = u_{max} \tag{3.46}$$

$$u_{max} = u_{m+1} < u_{m+2} < u_{m+3} < u_{m+4},$$

where the sequence $\{u_i\}$, $i = 0, \ldots, m+4$, is referred to as *knot sequence* (the choice of m will be explained shortly).

A cubic B-spline basis function is a piecewise cubic polynomial function defined on four successive parameter segments. On each segment, the basis function is defined by a cubic polynomial of the form [22]:

$$a_j + b_j u + c_j u^2 + d_j u^3, \quad i-3 \leq j \leq i. \tag{3.47}$$

The shape of the basis function depends on the length of the segments on which it is defined. On a uniform knot sequence with the knots at integers and all segments of length 1, the basis function B_i consists of the following 4 cubic polynomials (see Figure 3.4):

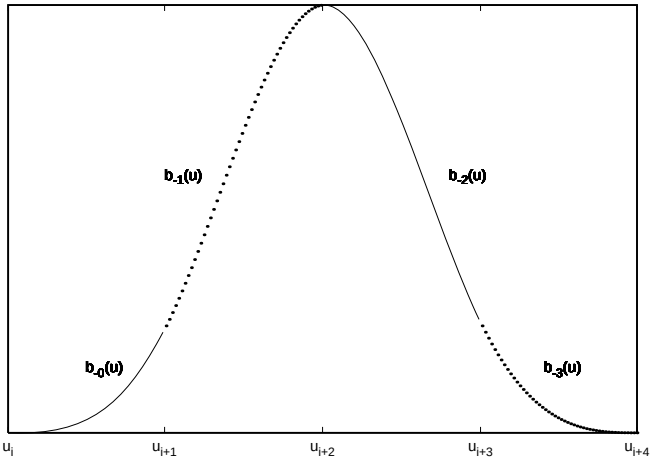


Figure 3.4: The cubic B-spline basis polynomials.

$$
\begin{aligned}
b_{-0}(u) &= \frac{1}{6}u^3 \qquad (3.48)\\
b_{-1}(u) &= \frac{1}{6}(1+3u+3u^2-3u^3)\\
b_{-2}(u) &= \frac{1}{6}(4-6u^2+3u^3)\\
b_{-3}(u) &= \frac{1}{6}(1-3u+3u^2-u^3)
\end{aligned}
$$

The indices of the segments indicate the basis functions that they come from. For example, on a segment $[u_i, u_{i+1})$, the four polynomials are the first polynomial of B_i, the second polynomial of B_{i-1}, the third polynomial of B_{i-2}, and the fourth polynomial of B_{i-3}, respectively (see Figure 3.5).

A B-spline curve is defined as a linear combination of the B-spline basis functions B_i, scaled by the corresponding control points P_i:

$$C(u) = \sum_i P_i B_i(u). \qquad (3.49)$$

On a uniform and integer knot sequence, the basis functions B_i are simply shifted copies of each other, and are defined as in Equation (3.48). On a nonuniform sequence, the basis functions have to be computed for each segment separately. The value of a basis function

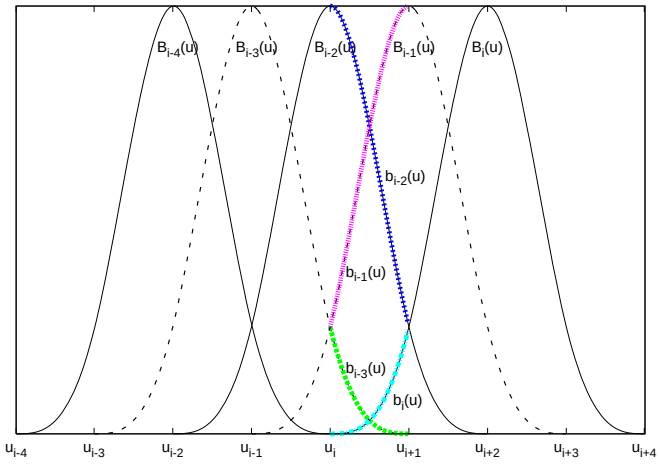


Figure 3.5: The cubic B-spline basis functions. The basis polynomials are shown in colour.

$B_i(u)$ for each value of the parameter u can be computed using the following recursive formula [46]:

$$B_{i,0}(u) = \begin{cases} 1, & if\ u_i \leq u < u_{i+1} \\ 0, & otherwise \end{cases} \tag{3.50}$$

$$B_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i}B_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}B_{i+1,p-1}(u),$$

where p is the degree and i is the segment number defined by its left knot u_i.

This formula demonstrates the recursive nature of B-spline functions, *i.e.*, any basis function $B_{i,p}$ of degree p can be computed as a linear combination of degree $p-1$ basis functions, $B_{i,p-1}$ and $B_{i+1,p-1}$. The recursion stops at the degree 0 basis function, the *step function* B-spline [22].

Four basis functions are required to define one cubic B-spline curve segment. If we index the basis functions (and the corresponding control points) from 0 to m and observe that each basis function is nonzero over four consecutive parameter segments, each segment bounded by a pair of knots, it follows that to define $m-2$ curve segments bounded by $m-1$ knots, we require $m+1$ basis functions (and control points) and $m+5$ knots in total. The total number of knots explains the choice of m in equation (3.46).

3.6.4 Spline Curves and Subdivision

The theory of subdivision is based on the refinement property of the spline curves. Here we show an example for B-spline curves, which are exactly the curves generated by the Catmull-Clark subdivision on a uniform knot sequence. First, we define the B-splines via repeated convolution and describe their refinement property [186]. We then show how the refinement property represents a direct connection between the subdivision methods and spline curves.

Definition of B-splines via repeated convolution

Let us define the *B*-splines via repeated convolution [186]. Any piecewise constant function can be written as:

$$x(t) = \sum x_i B_0^i(t), \tag{3.51}$$

where $B_0(t)$ is the box function defined as:

$$B_0(t) = \begin{cases} 1, & \text{if } 0 \leq t < 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.52}$$

and the functions $B_0^i(t) = B_0(t-i)$ are the translates of $B_0(t)$.

Let us represent the continuous convolution of two functions $f(t)$ and $g(t)$ with

$$(f * g)(t) = \int f(s)g(t-s)ds \tag{3.53}$$

A *B*-spline basis function of degree l can be obtained by convolving the basis function of degree $l-1$ with the box function $B_0(t)$:

$$\begin{aligned} B_1(t) &= \int B_0(s)B_0(t-s)ds, \\ B_2(t) &= \int B_1(s)B_0(t-s)ds, \\ &\vdots \\ B_l(t) &= \int B_{l-1}(s)B_0(t-s)ds. \end{aligned} \tag{3.54}$$

Refinement property of *B*-splines

The following is the refinement equation for *B*-splines of degree l:

$$B_l(t) = \frac{1}{2^l} \sum_{k=0}^{l+1} \binom{l+1}{k} B_l(2t-k). \tag{3.55}$$

It means that a *B*-spline of degree l can be written as a linear combination of translated and dilated copies of itself. This property is proven in [186].

Refinement and Subdivision

The refinement property is the link between the splines and subdivision. Let the spline curve be written as follows:

$$\gamma(t) = \sum_i p_i B_l(t-i), \tag{3.56}$$

where $B_l(t)$ is defined as in Equation (3.55) and p_i are the control points $p_i = (x_i, y_i)^\top \in \mathbb{R}^2$.

Let the vector of control points for this curve be written as:

$$\mathbf{p} = [\cdots, p_{-2}, p_{-1}, p_0, p_1, p_2, \cdots]^\top, \tag{3.57}$$

and the translates of the function B as a vector $\mathbf{B}(t)$:

$$\mathbf{B}(t) = [\cdots,\ B(t+2),\ B(t+1),\ B(t),\ B(t-1),\ B(t-2),\ \cdots] \tag{3.58}$$

We can now write the spline curve as $\gamma(t) = \mathbf{B}(t)\mathbf{p}$.

Using the refinement relation from Equation (3.55), each of the elements of B can now be rewritten in terms of its dilates:

$$\mathbf{B}(2t) = [\cdots,\ B(2t+2),\ B(2t+1),\ B(2t),\ B(2t-1),\ B(2t-2),\ \cdots], \tag{3.59}$$

and the refinement equations become:

$$\mathbf{B}(t) = \mathbf{B}(2t)S, \tag{3.60}$$

where the entries of the matrix S are given by Equation (3.55).

The spline curve $\gamma(t)$ can now be written as:

$$\gamma(t) = \mathbf{B}(t)\mathbf{p} = \mathbf{B}(2t)S\mathbf{p}, \tag{3.61}$$

which is the same curve written with respect to half as wide and twice as dense *B*-splines. The change from the basis $\mathbf{B}(t)$ to $\mathbf{B}(2t)$ also requires the control points to change from $\mathbf{p}$ to $S\mathbf{p}$.

Repeating the refinement process, we get:

$$\begin{aligned}
\gamma(t) &= \mathbf{B}(t)\mathbf{p}^0 && (3.62)\\
&= \mathbf{B}(2t)\mathbf{p}^1 = \mathbf{B}(2t)S\mathbf{p}^0 && (3.63)\\
&\vdots && (3.64)\\
&= \mathbf{B}(2^j t)\mathbf{p}^j = \mathbf{B}(2^j t)S^j\mathbf{p}^0, && (3.65)
\end{aligned}$$

where S is the *stationary subdivision matrix*, defining the relationship between control points on different subdivision levels:

$$\mathbf{p}^{j+1} = S\mathbf{p}^j \tag{3.66}$$

For cubic spline curves, the subdivision matrix entries, as given by Equation (3.55), are:

$$\begin{aligned}
s_0 &= \frac{1}{2^3}\binom{4}{0} = \frac{1}{8},\\
s_1 &= \frac{1}{2^3}\binom{4}{1} = \frac{4}{8},\\
s_2 &= \frac{1}{2^3}\binom{4}{2} = \frac{6}{8},\\
s_3 &= \frac{1}{2^3}\binom{4}{3} = \frac{4}{8},\\
s_4 &= \frac{1}{2^3}\binom{4}{4} = \frac{1}{8}.
\end{aligned} \tag{3.67}$$

The even coefficients s_0, s_2 and s_4 are applied to the control points from the previous level, and the odd coefficients s_1, s_3 to the newly inserted control points at each iteration of the

subdivision.

Although the presented theory describes the refinement property of *B*-spline curves, it also generalises to *B*-spline surfaces, where the basis function is a tensor product of the corresponding *B*-spline curve basis functions.

Modelling surfaces with uniform spline patches has its limitations. The spline curves are defined on a uniform knot sequence and the corresponding spline surfaces on a uniform grid. The Euler characteristic for a planar graph states that the topology of the surfaces modelled by a regular mesh (all vertices having degree 4 for quadrilateral meshes and 6 for triangular meshes) can only be an infinite plane, an infinite cylinder or a torus [186]. Modelling surfaces with uniform splines is therefore limited to particular topologies only. Additionally, enforcing higher-order continuity over patch borders when using rectangular spline patches in arbitrary control meshes is difficult and increases modelling complexity.

Subdivision schemes generalise the spline surface modelling on uniform grids with limited topology to surfaces on arbitrary grids with arbitrary topology. The central idea of subdivision is to model smooth and complex-shaped curves or surfaces by starting with a coarse approximation in the form of a control mesh and producing a smooth curve or surface as the limit of a sequence of successive refinements. We talk about the different subdivision schemes in the next section.

3.6.5 Subdivision Schemes

Many subdivision methods are simply generalisations of knot insertion, well-known in spline modelling [175, 186]. An interesting application of repeated knot insertion was described in 1974 by Chaikin [175]. Starting with a control polygon of a quadratic B-spline curve over a uniform knot sequence, in every iteration, a new knot was inserted at the midpoint of every interval of the knot sequence. If this procedure was repeated infinitely many times, the resulting sequence of polygons converged to the original curve. Chaikin's algorithm can be described as *corner cutting*, where, starting with a coarse polygon, at each iteration a new polygon is produced by cutting off the corners of the polygon from the pre-

vious iteration [55].

In terms of generalising the spline theory, different subdivision methods generalise different spline curves and surfaces. For example, the Catmull and Clark subdivision method [32] is based on tensor-product bi-cubic B-splines, while the Loop scheme generalises the three-directional quartic box splines [97].

The subdivision algorithm for modelling smooth 3-D objects starts with a coarse polyhedron, approximating the shape of the desired object. The coarse model is then refined by producing increasingly faceted approximations to the associated smooth shape. The subdivision rules used during this refinement process depend only on the topological connectivity of the initial polygonal model and yield surfaces with guaranteed smoothness [186]. During refinement, the rules associated with a subdivision scheme are applied recursively to construct a sequence of polygonal models. If these subdivision rules are represented by the operator S, the process has the form [174]

$$p^k = S\,p^{k-1}. \tag{3.68}$$

(*Cf.* Equation(3.66)). Applying S to an initial model p^0 yields a sequence of polygonal models: $p^1, p^2, \ldots, p^N$, where $N \to \infty$. The rules comprising S specify how the polygonal faces of p^{k-1} are split, as well as how the vertices of p^k are positioned in terms of the vertices p^{k-1}. The limit of this process is a smooth surface p^∞ that approximates or interpolates the coarse model p^0, depending on the chosen subdivision scheme.

A number of different subdivision schemes exist. The most common ones can be classified according to 4 different criteria [186]:

- Type of refinement rule - face split (primal) or vertex split (dual)
- Type of generated mesh (triangular or quadrilateral)
- Type of limit surface (approximating or interpolating)
- Smoothness of the limit surface for regular meshes (C^1, C^2, etc.)

	Triangular meshes	*Quadrilateral meshes*
Approximating	Loop Subdivision (C^2)	Catmull-Clark Subdivision (C^2)
Interpolating	Modified Butterfly (C^1)	Kobbelt (C^1)

Table 3.1: Face split (primal) subdivision schemes

scheme	continuity
Doo-Sabin, Midedge	(C^1)
Biquartic	(C^2)

Table 3.2: Vertex split (dual) subdivision schemes

The subdivision schemes which can be classified using these criteria (some exist which can't be placed this way) are listed in Table 3.1 and Table 3.2. We describe the Catmull-Clark subdivision scheme, which we used when constructing our generic upper-body model, in more detail in the next section.

3.6.6 Catmull-Clark Subdivision Scheme

The Catmull-Clark scheme is based on the tensor product bi-cubic spline. The rules of the Catmull-Clark scheme are defined for meshes with quadrilateral faces where the *ordinary* vertices have valence 4 and vertices with other valences are called *extraordinary*. The valence of a vertex is the number of edges that meet at that vertex. The scheme produces surfaces that are C^2-continuous everywhere except at extraordinary vertices, where they are C^1-continuous.

The Catmull-Clark scheme can be applied to an arbitrary polygonal mesh by performing a *Catmull-Clark split* on the input mesh, which transforms it into a quadrilateral mesh and isolates the extraordinary vertices. Catmull-Clark split divides every n-sided face into n quadrilateral faces by inserting a new face vertex at the centroid of the existing face vertices and n new edge vertices at the midpoints of the current n edges. The new face vertex is then

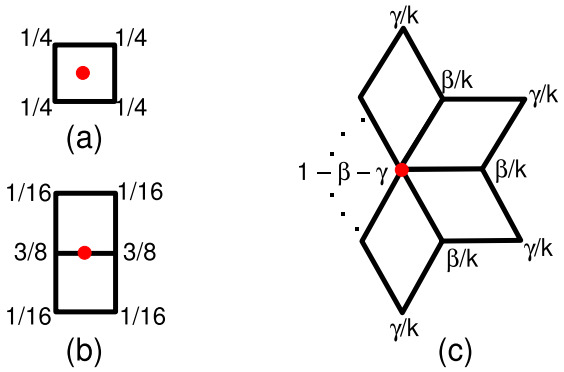


Figure 3.6: Catmull-Clark Subdivision rules for interior mesh vertices. (a) new face vertex, (b) new edge vertex, (c) old vertex with $\beta = 3/2k$, $\gamma = 1/4k$ and k the vertex valence. Diagrams were taken from [186].

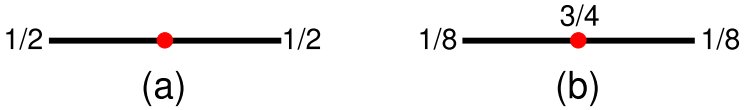


Figure 3.7: Catmull-Clark Subdivision rules for boundary mesh vertices. (a) new edge vertex, (b) old edge vertex. Diagrams taken from [186].

connected with new edge vertices, thus producing n new quadrilateral faces.

A general set of Catmull-Clark subdivision rules thus starts with an arbitrary polygonal mesh, turns it into a quadrilateral mesh after the first iteration using the Catmull-Clark split, and then proceeds with the subdivision steps described in the Algorithm 3.5 [123]. These subdivision steps can be implemented in different ways. We describe two popular approaches, the *template-based* approach and the *factored* approach.

Template-based approach

In the template-based approach, a *template* (also called *mask* or *stencil*) is given, which specifies the weights that should be used when calculating the new vertex positions by means of a weighted average of the neighbouring vertices [186]. It assumes that the mesh is quadrilateral. An example of such templates is given in Figure 3.6 and Figure 3.7.

The template rules are interpreted as follows. A new face vertex is computed according to the template (a) in Figure 3.6, which states that it is simply an average of the old face vertices, or,

$$v_n = \frac{1}{4}v_1 + \frac{1}{4}v_2 + \frac{1}{4}v_3 + \frac{1}{4}v_4 = \frac{1}{4}\sum_{i=1}^{4} v_i. \tag{3.69}$$

Similarly, a new interior (non-boundary) edge vertex is computed with the template (b) in Figure 3.6, which states that it is a weighted average of vertices belonging to two faces adjacent to that edge, with the edge vertices having weights $3/8$ and the remaining vertices $1/16$. The other templates are interpreted similarly.

Factored approach

Subdivision can also be efficiently performed in a factored way, where the polygon faces are first linearly subdivided to produce a new, finer control mesh, followed by repositioning the vertices in an averaging (smoothing) step [174].

A linear subdivision step uses the *Catmull-Clark split* to divide the polygonal face into smaller quadrilaterals by first inserting new vertices at the midpoint of each edge of the face and one vertex at the centroid of the face. The newly inserted vertices are then connected with edges to form m quadrilaterals from an m-sided polygonal face (see Figure 3.8).

The linear subdivision step is followed by an averaging step, where each vertex is repositioned at the average of the centroids of all quadrilaterals that contain this vertex (see Figure 3.9).

Both steps can be implemented very efficiently, each as a single pass over the list of faces. The rules for linear subdivision and averaging produce surfaces which are C^2-continuous everywhere except at extraordinary vertices, where they are only C^1-continuous. To reduce the appearance of these discontinuities, a correction factor introduced by Maillot and Stam [102] can be used as follows.

Let $\hat{p}_i^k$ be the i-th vertex after the k-th linear subdivision pass and p_i^k be the position of

Algorithm 3.5: General Catmull-Clark subdivision algorithm [186].

Input: Control mesh of an arbitrary topology on level $k = 0$.

repeat

- For each face on level k, introduce a new face point which is the average of all the old vertices defining the face.
- For each (nonboundary) edge on level k, introduce a new edge point which is the average of the following four points: two old vertices defining the edge and two new face points of the faces adjacent to the edge.
- For each (nonboundary) vertex V on level k, introduce a new vertex V_{new} whose position is

$$V_{new} = \frac{F}{n} + \frac{2n}{E} + \frac{(n-3)V}{n}$$

  where F is the average of the new face vertices of all faces adjacent to the old vertex V, E is the average of the midpoints of all edges incident on the old vertex V, and n is the number of the edges incident on the vertex.
- Form new edges by connecting each new face point to the new edge points of the edges defining the old face and by connecting each new vertex point to the new edge points of all old edges incident on the old vertex point.
- Define faces on level $k+1$ as those enclosed by new edges.
- $k = k+1$.

until *k is large enough.*

Output: Smooth approximation of the limit surface.

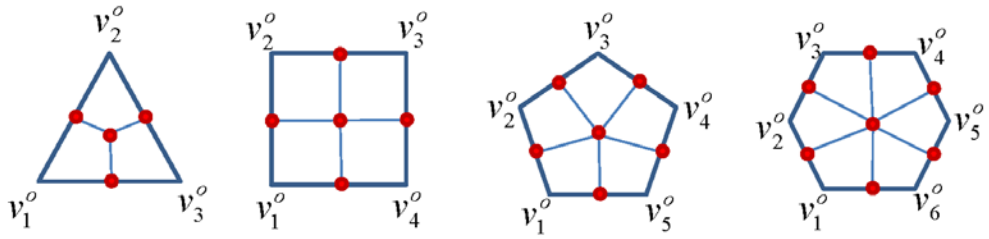


Figure 3.8: Linear subdivision. A new vertex is inserted at the centroid of the face and at the midpoint of every edge. All new vertices are then connected with new edges to form new quadrilateral faces. The old vertices are shown as v_i^o.

that vertex after the k-th averaging pass. We reposition the vertices of the mesh according to the update equation:

$$\hat{p}_i^k + w(n)(p_i^k - \hat{p}_i^k) \tag{3.70}$$

where n is the valence of the vertex and $w(n)$ a function describing the correction factor. The correction factor for quadrilateral subdivision can be chosen as $w(n) = \frac{4}{n}$.

3.6.7 Conclusion

The subdivision schemes described in this section are used to model a geometric shape of an arbitrary topology. We use the Catmull-Clark subdivision surface to model the upper human body for the purpose of completing the sparse disparity data. The model is first used to estimate the body pose of the imaged person from the available silhouette constraints, as described in Chapter 5. Once the pose is estimated, the model is then fit to the disparity data by means of surface deformation (Chapter 6). The method we use to deform the surface layer of the model, represented by the subdivision surface, to better fit the data is that of *quasi-interpolation*, which we describe in the next section.

3.7 Quasi-Interpolation

3.7.1 Introduction

The use of quasi-interpolation for subdivision surface fitting was first suggested by Litke *et al.* in [96]. The basic idea behind quasi-interpolation is to use local, weighted averages of samples of the desired surface as control points. The method computes the subdivision

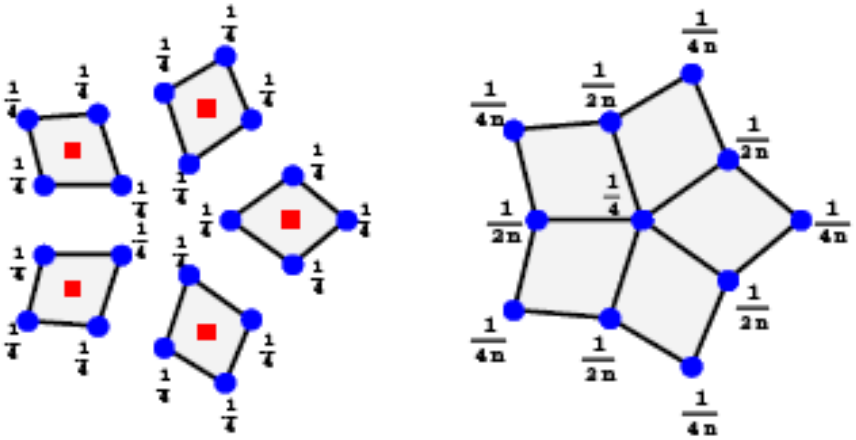


Figure 3.9: Averaging step (illustration taken from [174]).

surface control polygon, also called the *base mesh*, for which the resulting subdivision limit surface best approximates the given data. No solution of global linear systems with an *a priori* fixed set of coefficients is required. Quasi-interpolation has an optimal order of convergence on regular meshes, making it asymptotically as good as least-squares [96].

In the next section we describe the quasi-interpolation operator which allows us to deform the mesh of our generic body model to better fit the data.

3.7.2 Quasi-Interpolation Stencils

Summarising Litke *et al.* [96], it is instructive to first illustrate the quasi-interpolation operator on the univariate approximation problem. Let us assume that samples of a sufficiently smooth function f are given which we would like to approximate with a uniform cubic spline curve. The solution is to find values of control points for an optimal cubic spline with knots at the integers. The least-squares solution would be to set up a linear system whose size is proportional to the number of knots and solve it for the control point values. Instead of solving a global system, the quasi-interpolation operator can be applied locally as follows:

$$p_i = -\frac{1}{6} f(i-1) + \frac{4}{3} f(i) - \frac{1}{6} f(i+1), \quad \forall i \in \mathbb{Z}, \tag{3.71}$$

where p_i is the control point at the knot i. Although this quasi-interpolant is a local operator, it is an exact interpolant whenever f is a cubic polynomial, *i.e.*, the data samples come

from a cubic polynomial curve [96].

The univariate limit stencil with weights for cubic B-spline curves can be written as $L = (1/6, 4/6, 1, 6)$. Writing the identity stencil as $I = (0, 1, 0)$, we can derive the quasi-interpolation stencil Q as $Q = 2I - L$:

$$(-\frac{1}{6}, \frac{4}{3}, -\frac{1}{6}) = (0, 2, 0) - (\frac{1}{6}, \frac{4}{6}, \frac{1}{6}). \tag{3.72}$$

The stencil is applied to the mesh by centering it on the mesh vertex currently being updated and using the remaining weights on its 1-neighbourhood. The rule for constructing quasi-interpolation stencils from cubic B-spline limit stencils can be simply extended to a bivariate case on a regular mesh by means of the tensor product:

$$L = \frac{1}{36}\begin{bmatrix} 1 & 4 & 1 \\ 4 & 16 & 4 \\ 1 & 4 & 1 \end{bmatrix} \quad I = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad Q = 2I - L = \frac{1}{36}\begin{bmatrix} -1 & -4 & -1 \\ -4 & 56 & -4 \\ -1 & -4 & -1 \end{bmatrix}, \tag{3.73}$$

where L is a limit stencil for a bicubic B-spline. Given that limit stencils L are known for all popular subdivision rules such as creases, boundaries, etc., the formula $Q = 2I - L$ is simply extended to the irregular mesh case by substituting L with the appropriate limit stencil.

3.7.3 Conclusion

We use the quasi-interpolation method described in this section to fit the Catmull-Clark subdivision model of the upper body to an unstructured cloud of disparity data points (Chapter 6). However, before the model can be fit to the disparity data, it must first assume the correct body pose, *i.e.*, the body pose of the imaged person. To achieve this, we perform body pose estimation from multi-view images (Chapter 5) using a Particle Swarm Optimisation method, which we describe in the next section.

3.8 Particle Swarm Optimisation

3.8.1 Introduction

Particle Swarm Optimisation (PSO) is an evolutionary computation technique introduced by Kennedy and Eberhart in 1995 [88]. The idea originated from the simulation of a simplified social model where the agents were thought of as collision-proof birds and the original intent was to graphically simulate the unpredictable choreography of a bird flock in their search for food. We describe the PSO algorithm in detail in this section.

3.8.2 Original PSO Algorithm

Let us assume an n-dimensional search space $\mathbb{S} \subseteq \mathbb{R}^n$, a swarm consisting of N particles and a fitness (cost) function $f : \mathbb{S} \to \mathbb{R}$ defined on the search space $\mathbb{S}$. The position of the i-th particle at time t is represented as an n-dimensional vector $X_i(t) = (x_{i1}, x_{i2}, ..., x_{in})^\top \in \mathbb{S}$. The velocity of this particle is also an n-dimensional vector $V_i(t) = (v_{i1}, v_{i2}, ..., v_{in})^\top \in \mathbb{S}$. The best position encountered by the i-th particle up to time t (*personal* best) is denoted as $P_i(t) = (p_{i1}, p_{i2}, ..., p_{in})^\top \in \mathbb{S}$ and the value of the fitness function at that position $pbest_i = f(P_i(t))$. The index of the particle with the overall best position up to time t (*global* best) is denoted as g and $gbest = f(P_g(t))$.

In the original algorithm, the positions and velocities of the particles are updated as follows:

$$V_i(t) = V_i(t-1) + \varphi_1(P_i(t-1) - X_i(t-1)) + \varphi_2(P_g(t-1) - X_i(t-1)), \quad (3.74)$$

$$X_i(t) = X_i(t-1) + V_i(t), \quad (3.75)$$

where $\varphi_1 = c_1 rand(0,1)$ and $\varphi_2 = c_2 rand(0,1)$ are two parameters controlling the influence of the cognitive and social memory on an individual particle's behaviour. The constants c_1 and c_2 were originally set to $c_1 = c_2 = 2.0$, which gave the stochastic factor $rand(0,1)$ a mean of 1.0 and caused the particles to "overfly" the target for about half of the time [88].

Equation (3.74) describes how a particle's velocity is dynamically updated. The equation consists of three parts. The first part, $V_i(t-1)$, is the *momentum*, which ensures that the velocity is not changed too abruptly. The second part, $(P_i(t-1)-X_i(t-1))$, is the *cognitive* part, which represents the particle's memory based on its own experience. The third part, $(P_g(t-1)-X_i(t-1))$, is the *social* part, representing the collaboration among particles and learning from the group's experience [142]. Equation (3.75) describes how a particle's position changes with the updated velocity.

The original PSO algorithm was later modified by its authors and other researchers to improve its search capabilities and convergence. One of the important modifications of PSO was introduced in 1998 by Shi and Eberhart [143]. They changed the velocity update equation of the swarm by adding an additional parameter called *inertia weight*, *w*. The aim of this parameter was to guide the search behaviour of the swarm. The larger the inertia parameter value, the more global the search, and vice versa. Several other modifications were added later, but for the purpose of this work we focus on the contribution of [143], as it is also the version of PSO which we used in our experiments. We describe the PSO algorithm with inertia weight parameter next.

PSO Algorithm with Inertia Weight Parameter

The step-by-step description of the PSO algorithm with inertia weight parameter is shown in Algorithm 3.6. We describe the relevant PSO parameters in more detail next. In this section, we provide only a generic description of the PSO parameters. The actual parameter settings used in our pose estimation experiments are given later in Chapter 5.

Inertia Weight Parameter

The inertia weight parameter *w* is added to the velocity update equation as follows:

$$V_i(t) = wV_i(t-1) + \varphi_1(P_i(t-1)-X_i(t-1)) + \varphi_2(P_g(t-1)-X_i(t-1)). \qquad (3.76)$$

The value of the inertia weight can remain constant throughout the search or change with time. It plays an important role in directing the exploratory behaviour of the particles. Higher inertia values push the particles to explore more of the search space and emphasise

Algorithm 3.6: PSO algorithm with inertia weight parameter [143].

Input: Population of N particles $\{X_i\},\ i = 1,\ldots,N$.

Initialisation

- $t = 0$.
- Initialise a population of particles $\{X_i(0)\},\ i = 1,\ldots,N$, with random positions and velocities in the search space $\mathbb{S}$.
- For each particle evaluate the desired fitness function and set $P_i(0) = X_i(0)$ and $pbest_i = f(X_i(0))$.
- Identify the best particle in the swarm and store its index as *gbest* and its position as $P_g(0)$.

Loop:

repeat

- t = t+1.
- Move the swarm by updating the position of every particle according to:

$$\begin{aligned} V_i(t) &= wV_i(t-1) + \varphi_1(P_i(t-1) - X_i(t-1)) \\ &\quad + \varphi_2(P_g(t-1) - X_i(t-1)) \end{aligned}$$

$$X_i(t) = X_i(t-1) + V_i(t)$$

  where w is the inertia weight parameter.

- For $i = 1,\ldots,N$ update $pbest_i$ and *gbest*.

until *stopping condition is met.*

Output: Best particle $X_g(t_{final})$.

their individual velocity. In the evolutionary terminology this behaviour is called *exploration*. Lower inertia values force particles to focus on a smaller search area and move towards the best solution found so far, which is referred to as *exploitation* [89].

Social and Cognition Parameters

The parameters φ_1 and φ_2 influence the *social* and *cognition* components of the swarm behaviour [143]. They are composed of a random number and a constant and can be written as $\varphi_1 = c_1 rand_1(0,1)$ and $\varphi_2 = c_2 rand_2(0,1)$, where c_1 and c_2 are two constants and $rand_1(0,1)$ and $rand_2(0,1)$ two random numbers in the interval $[0,1]$.

3.8.3 Conclusion

PSO has become a very successful research area since its invention in 1995. Several interesting applications and modifications of PSO have been reported in the literature. For an overview of the relevant research in this area the interested reader will find a good starting point in [54, 118, 119, 120].

With this section we conclude the second part of the background theory review, which looked at the methods for modelling and manipulation of the articulated human body model that we used in this thesis. In the next part, we look at the novel-view synthesis algorithm which we use to synthesise novel views shown in Chapters 4 and 8.

3.9 Novel-View Synthesis

3.9.1 Introduction

This is the last part of the background theory review chapter where we look at the problem of novel-view synthesis. The idea behind the novel-view synthesis is to acquire a limited number of images of a scene, use them to construct some form of representation and use this representation to render novel views of the scene, *i.e.*, views that were not included in the original set and were therefore “not seen before”.

Different view synthesis approaches use different representations. On a very general level, they can be divided into three different groups. In the first group are the approaches which build an explicit textured 3-D model of the imaged scene and use it to render novel views. In the second group are the approaches which use a large collection of images and interpolate between them to obtain novel views. In the third group are the approaches which use a smaller collection of images combined with various geometric constraints which allow for novel view generation. The appropriate references are listed in Section 2.7.

Our novel-view synthesis framework presented in this thesis is a hybrid approach in the sense that it uses a combination of representations. The method that we use for the actual novel view synthesis is based on a *trilinear-tensor* geometric constraint proposed by Avidan and Shashua in [15] and, on its own, belongs to the latter of the three groups. However, we also use a generic 3-D model of the scene as the *a priori* knowledge about the geometry of the scene. The generic model allows us to complete the patchy disparity map, which is then used as an input to the trilinear-tensor-transfer view synthesis algorithm.

In this section, we describe the trilinear-tensor view synthesis method in more detail. We begin with the definition of the trilinear tensor and its incidence relations, then focus on the point-point-point correspondence and finally describe the novel-view synthesis algorithm using the trilinear-tensor point transfer method.

3.9.2 Trilinear Tensor and Incidence Relations

Analogous to the fundamental matrix, which captures the epipolar geometry of two views, *trilinear tensor* (also known as *trifocal tensor*) captures the information about the epipolar geometry of three views (see Figure 3.10), also called the *trifocal geometry*, and encapsulates all the geometric relations between three views that are independent of scene structure and invariant under 3-D projective transforms [70].

Various incidence relations between images of points and lines in three views can be conveniently expressed using the trilinear tensor. We first define the trilinear tensor and then summarise the most common incidence relations, as described in [70].

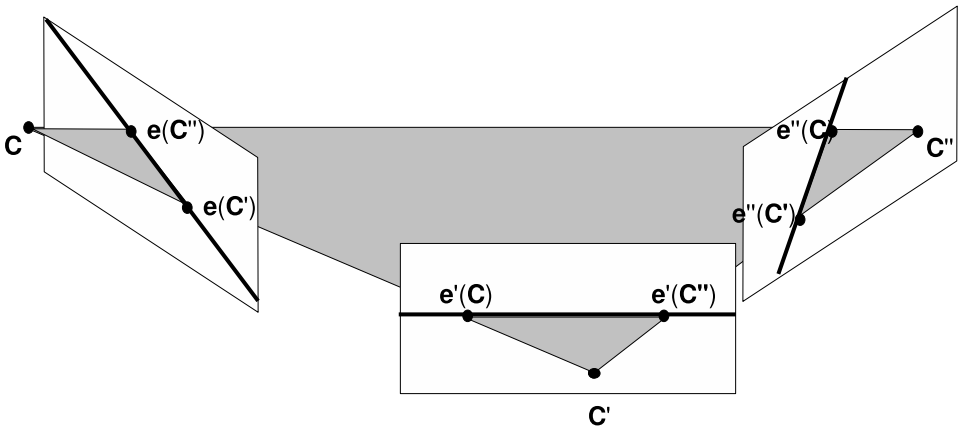


Figure 3.10: The trifocal geometry. The plane defined by the three camera centres, C, C' and C'', is called the *trifocal plane*. In a 3-focal camera setup there are 6 epipoles, $e(C'), e(C''), e'(C), e'(C''), e''(C), e''(C')$, where, *e.g.*, $e'(C'') = P'C''$.

Trilinear Tensor

The trilinear tensor T is a valency 3 tensor T_i^{jk} with one covariant (column) index, i, and two contravariant (row) indices, j,k. It is represented by a homogeneous $3 \times 3 \times 3$ array and is defined uniquely by the camera matrices of the three views it encapsulates.

As any set of three cameras, $\{P, P', P''\}$, is equivalent to a canonical set with $P = [I|\mathbf{0}]$ under projective transformations of space and the properties we are interested in are invariant to projective transforms, the trilinear tensor can be defined using the canonical camera set [70], without loss of generality.

Let us first define the trilinear tensor using matrix notation (the tensor notation will be introduced later in Section 3.9.3) and denote a canonical set of 3 cameras with camera projection matrices P, P' and P'' as

$$P = [I|0], \;\; P' = [A|\mathbf{a}_4], \;\; P'' = [B|\mathbf{b}_4], \tag{3.77}$$

where A and B are 3×3 matrices and the vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ are the i-th columns of the matrices P' and P'', respectively.

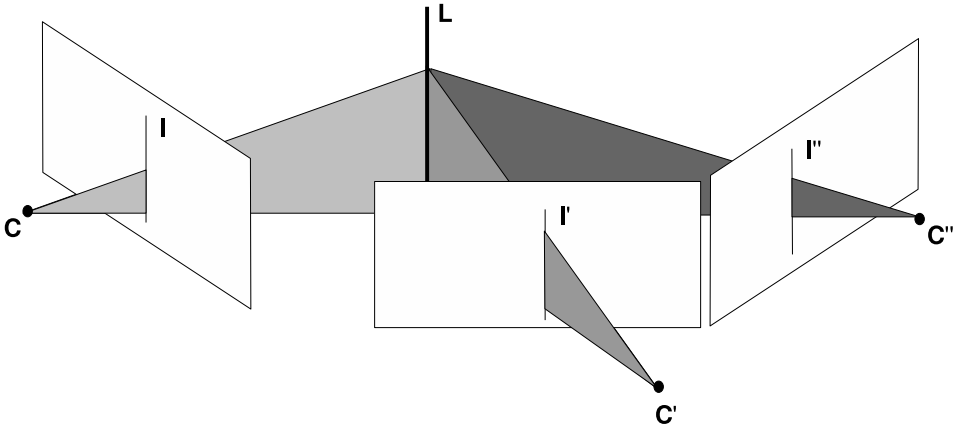


Figure 3.11: Line-line-line incidence relation in three views.

When these 3 camera projection matrices are known, the trilinear tensor can be written as a set of three matrices $\{T_1, T_2, T_3\}$, where

$$T_i = \mathbf{a}_i \mathbf{b}_4^\top - \mathbf{a}_4 \mathbf{b}_i^\top, \tag{3.78}$$

and $^\top$ denotes the vector transpose operation.

We now describe four incidence relations encapsulated by a trilinear tensor and the trifocal geometry: the *line-line-line* correspondence, the *point-line-line* correspondence, the *point-line-point* correspondence and the *point-point-point* correspondence.

Line-line-line correspondence

Let us denote image points in homogeneous coordinates as column vectors $\mathbf{x} = (x^1, x^2, x^3)^\top$ and image lines in homogeneous coordinates as row vectors $\mathbf{l} = (l_1, l_2, l_3)$. The indices of image point coordinates are written in superscript notation for reasons of compatibility with the tensor notation.

Using Equation (3.78), the *line-line-line* incidence relationship can be written as:

$$l_i \simeq \mathbf{l}'^\top T_i \mathbf{l}'', \tag{3.79}$$

or

$$\mathbf{l}^\top \simeq \mathbf{l}'^\top [T_1, T_2, T_3] \mathbf{l}''. \tag{3.80}$$

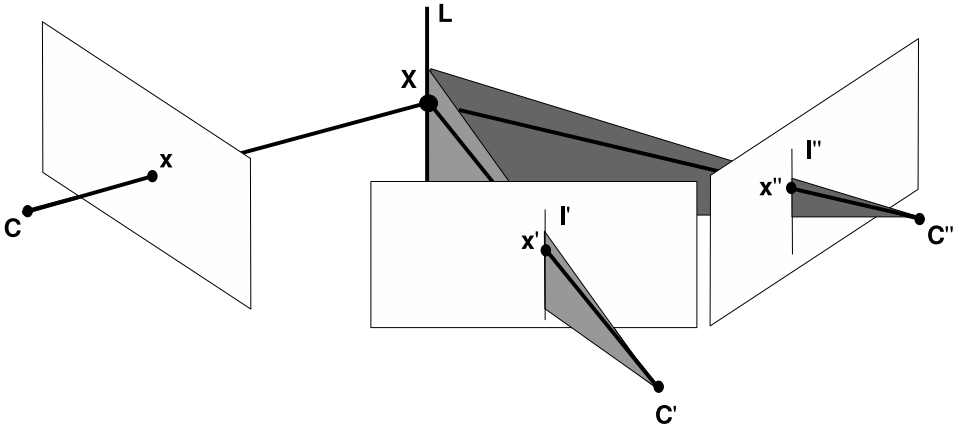


Figure 3.12: Point-line-line incidence relation in three views.

This relationship involves homogeneous quantities and holds only up to a scale. The scale factor can be eliminated by taking the vector cross product of both sides, which must equal to zero:

$$(\mathbf{l}'^{\top}[T_1,T_2,T_3]\mathbf{l}'')\times\mathbf{l} = (\mathbf{l}'^{\top}[T_1,T_2,T_3]\mathbf{l}'')[\mathbf{l}]_{\times} = \mathbf{0}^{\top}, \tag{3.81}$$

where $[\mathbf{l}]_{\times}$ denotes the cross product operation using a skew-symmetric matrix corresponding to $\mathbf{l}$ (see Appendix A for definition of a cross product using skew-symmetric matrices).

Point-line-line correspondence

A point $\mathbf{x}$ lying on the line $\mathbf{l}$ must satisfy

$$\mathbf{x}^{\top}\mathbf{l} = \sum_i x^i l_i = 0. \tag{3.82}$$

Substituting l_i in Equation (3.82) with Equation (3.79) gives

$$(\mathbf{l}'^{\top}(\sum_i x^i T_i)\mathbf{l}'') = 0. \tag{3.83}$$

This is a *point-line-line* correspondence, which holds when a 3-D line $\mathbf{L}$ maps to $\mathbf{l}'$ in the second view, $\mathbf{l}''$ in the third view and a line passing through $\mathbf{x}$ in the first view.

Point-line-point correspondence

A homography from the first to the third image induced by a line $\mathbf{l}'$ in the second image is given by $\mathbf{x}'' \simeq H_{13}(\mathbf{l}')\mathbf{x}$, where

$$H_{13}(\mathbf{l}') = [T_1^{\top}, T_2^{\top}, T_3^{\top}]\mathbf{l}'. \tag{3.84}$$

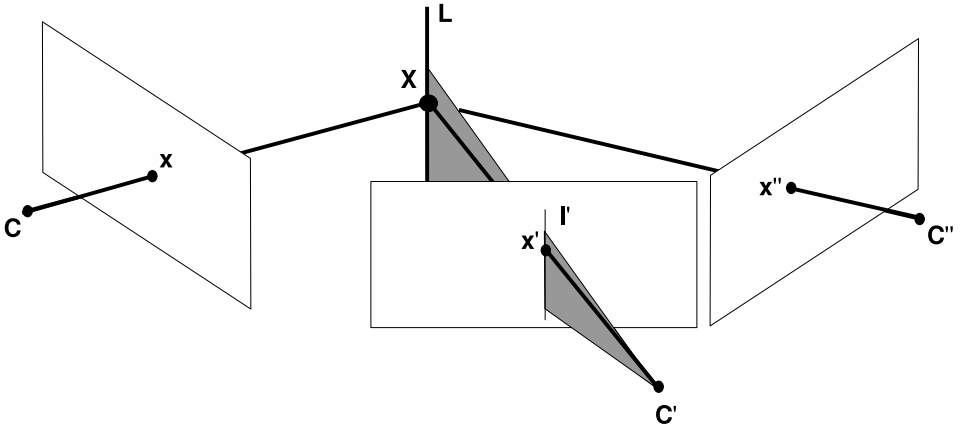


Figure 3.13: Point-line-point incidence relation in three views.

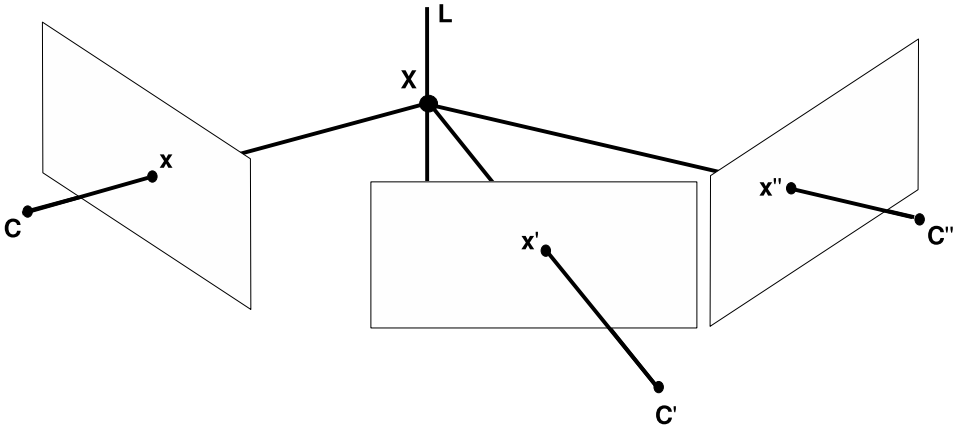


Figure 3.14: Point-point-point incidence relation in three views.

This gives

$$\mathbf{x}'' \simeq H_{13}(\mathbf{l}')\mathbf{x} = [T_1^\top \mathbf{l}', T_2^\top \mathbf{l}', T_3^\top \mathbf{l}']\mathbf{x} = (\sum_i x^i T_i^\top)\mathbf{l}'. \tag{3.85}$$

The homogeneous scale factor is eliminated by multiplying the transpose of both sides from the right by $[\mathbf{x}'']_\times$ to give:

$$\mathbf{l}'^\top (\sum_i x^i T_i)[\mathbf{x}'']_\times = \mathbf{0}^\top. \tag{3.86}$$

Point-point-point correspondence

The line $\mathbf{l}'$ in the second view passes through $\mathbf{x}'$ and can be written as

$$\mathbf{l}' = \mathbf{x}' \times \mathbf{y}' = [\mathbf{x}']_\times \mathbf{y}' \tag{3.87}$$

for some point $\mathbf{y}'$ on $\mathbf{l}'$. Substituting Equation (3.87) into Equation (3.86) gives

$$\mathbf{y}'^{\top}[\mathbf{x}']_{\times}^{\top}(\sum_i x^i T_i)[\mathbf{x}'']_{\times} = \mathbf{0}^{\top}. \tag{3.88}$$

As the relationship in Equation (3.86) is true for all lines $\mathbf{l}'$ through $\mathbf{x}'$, it is independent of $\mathbf{y}'$ and after multiplying both sides of the equation from the left by $(\mathbf{y}'\mathbf{y}'^{\top})^{-1}\mathbf{y}'$, the *point-point-point correspondence* relation becomes:

$$[\mathbf{x}']_{\times}^{\top}(\sum_i x^i T_i)[\mathbf{x}'']_{\times} = 0_{3\times 3}. \tag{3.89}$$

When the corresponding pairs of points between two views are known, the point-point-point correspondence relation conveniently specifies how to recover the corresponding point in the third view. This allows for novel-view synthesis via tensor transfer using two reference views and a corresponding disparity map. To specify the necessary transfer equations, it is helpful to first introduce the tensor notation.

3.9.3 Point-Point-Point Correspondence and Tensor Notation

Let us denote, as before, the image points as $\mathbf{x} = (x^1, x^2, x^3)^{\top}$. Let us also denote the ij-th entry of a matrix A by a^i_j with index i being the contravariant (row) index and j being the covariant (column) index. By convention, the indices repeated in the contravariant and covariant positions imply summation over the range of the index, *e.g.*, $x'^i = a^i_j x^j = \sum_j a^i_j x^j = a^i_1 x^1 + a^i_2 x^2 + a^i_3 x^3$.

In tensor notation, the canonical set of 3 cameras with camera projection matrices P, P' and P'' is now expressed as

$$P = [I|0], \;\; P' = [a^i_j], \;\; P'' = [b^i_j], \tag{3.90}$$

and the elements of the trilinear tensor as

$$T_i{}^{jk} = a^j_i b^k_4 - a^j_4 b^k_i. \tag{3.91}$$

The point-point-point relation is a special case of the point-line-line relation (Equation (3.83), Figure 3.12), which in tensor notation is written as

$$x^i l'_j l''_k T_i{}^{jk} = 0. \tag{3.92}$$

If the choice of the lines $\mathbf{l}'$ through $\mathbf{x}'$ in the second image and $\mathbf{l}''$ through $\mathbf{x}''$ in the third image is restricted to lines which are parallel to the image x-axis, parallel to the image y-axis, or run through the image coordinate origin, the Equation (3.92) becomes a point-point-point relation:

$$x^i(x'^j\varepsilon_{jpr})(x''^k\varepsilon_{kqs})T_i^{pq} = 0_{rs}, \tag{3.93}$$

where ε_{jpr} denotes the cross-product operator in tensor notation (see Appendix C).

For $r = 1,2,3$ and $s = 1,2,3$, this relation gives rise to 9 trilinearities of which only the following 4 are linearly independent:

$$\begin{aligned}
x''^1 T_i^{13} x^i - x''^1 x'^1 T_i^{33} x^i + x'^1 T_i^{31} x^i - T_i^{11} x^i = 0 \\
x''^2 T_i^{13} x^i - x''^2 x'^1 T_i^{33} x^i + x'^1 T_i^{32} x^i - T_i^{12} x^i = 0 \\
x''^1 T_i^{23} x^i - x''^1 x'^2 T_i^{33} x^i + x'^2 T_i^{31} x^i - T_i^{21} x^i = 0 \\
x''^2 T_i^{23} x^i - x''^2 x'^2 T_i^{33} x^i + x'^2 T_i^{32} x^i - T_i^{22} x^i = 0
\end{aligned} \tag{3.94}$$

From Equation (3.94) follows that, given two views in full correspondence and the trilinear tensor relating the two views with the third, virtual view, the entire third view can be synthesised by means of forward warping: from each trilinearity, one can extract either x''^1 or x''^2 and so obtain $\mathbf{x}'' = (x''^1, x''^2)$ for every matching pair $(\mathbf{x}, \mathbf{x}')$ [15].

3.9.4 Novel-View Synthesis Algorithm

Algorithm 3.7 presents the view synthesis algorithm that we use to synthesise novel views shown in Chapters 4 and 8. It uses Equation (3.94) on a stereo pair of images with corresponding camera projection matrices and a dense disparity map. The novel view is specified by a virtual camera projection matrix. As the stereo pair of cameras is arranged on a wide baseline, the two images show the scene from two sufficiently different points of view that there is a noticeable difference in the texture content of the two images. To enhance the realism of the novel view, a heuristic is used where the closest original view to the novel view is identified and the point transfer is first performed from that view. The point transfer from the remaining view is then used to fill in any texture information which was missing in the first view and is available from the second one.

3.9.5 Conclusion

We described the trilinear tensor geometric constraint, its incidence relations, and the novel-view synthesis algorithm using the trilinear-tensor point transfer method to synthesise a virtual view from two given views in full correspondence. With this section we conclude the background theory review.

3.10 Conclusion

The key requirement for a high-quality synthetic view synthesised with the trilinear tensor transfer is that the two given views are in *full correspondence*, *i.e.*, the correspondence information is known for every pixel and the corresponding disparity map is *dense* and *complete*. In a setup like ours, where the cameras are arranged in a wide-baseline stereo configuration, dense disparity maps are difficult to achieve with the stereo correspondence search alone.

In order to be used for high-quality novel-view synthesis, the disparity maps must first be completed with a post-processing algorithm. The post-processing is typically done by interpolating the existing disparity information, however, this method achieves only limited results, as we will show in the next chapter.

Algorithm 3.7: Trilinear-Tensor Novel-View Synthesis, based on [15].

Input: left and right image, I_l and I_r, left and right disparity map, D_{lr} and D_{rl}, left, right and virtual camera projection matrices, P_l, P_r, and P_s, respectively.

- Convert the camera projection matrices to canonical camera set.
- For the left temporary view, L_s:
  - With $P = P_l$, $P' = P_r$ and $P'' = P_s$, construct the tensor T using Eq. (3.91)
  - **Loop:**
    - Using D_{lr}, for every pixel in the left image, $I_l(i, j)$, and the corresponding pixel in the right image, $I_r(i-d, j)$, use Equation (3.94) to calculate the image coordinates of the corresponding point in the left temporary view, $L_s(i_n, j_n)$
    - Transfer the texture of pixel $I_l(i, j)$ to the pixel $L_s(i_n, j_n)$
- For the right temporary view, R_s:
  - With $P = P_r$, $P' = P_l$ and $P'' = P_s$, construct the tensor T using Eq. (3.91)
  - **Loop:**
    - Using D_{rl}, for every pixel in the right image, $I_r(i, j)$, and the corresponding pixel in the left image, $I_l(i-d, j)$, use Equation (3.94) to calculate the image coordinates of the corresponding point in the right temporary view, $R_s(i_n, j_n)$
    - Transfer the texture of pixel $I_r(i, j)$ to the pixel $R_s(i_n, j_n)$
- For the final synthetic view, I_s:
  - For every pixel in the synthetic view, $I_s(i, j)$, copy the texture from the closer of the temporary views, L_s or R_s, if it has texture available, otherwise transfer the texture from the remaining temporary view.

Output: Synthetic view I_s

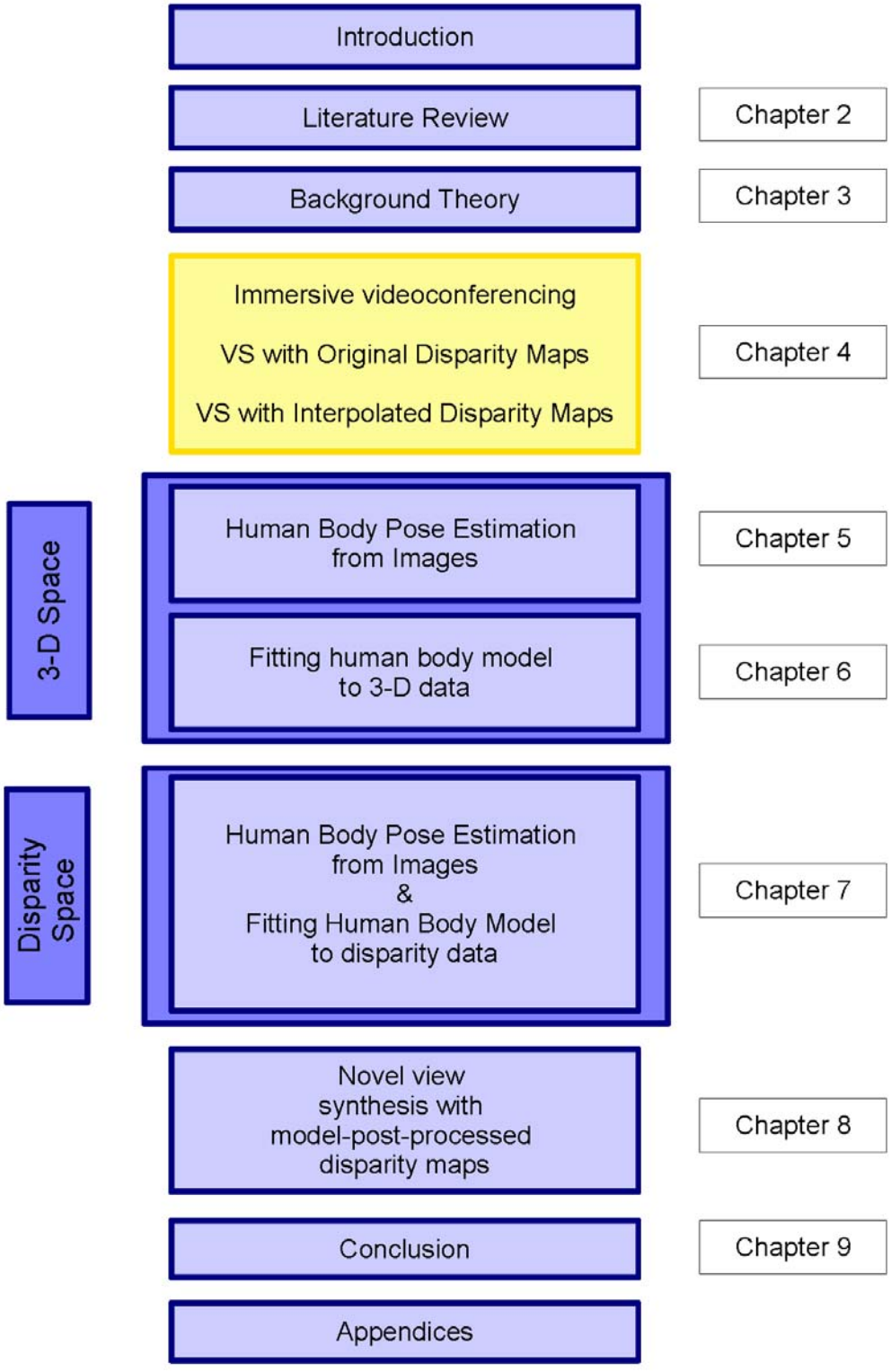


Figure 3.15: The thesis structure diagram.

Chapter 4

View Synthesis for Immersive Videoconferencing

4.1 Introduction

This chapter describes the role of the novel view synthesis in an immersive videoconferencing environment. We present the VIRTUE videoconferencing system [82, 136] which successfully demonstrated the potential of using novel view synthesis in the videoconferencing context. VIRTUE project also highlighted the low quality disparity map problem associated with the wide-baseline stereo which gave the foundation to this thesis.

We proceed by describing our own camera setup that was used to perform experimental work for the purpose of the thesis and motivate the need for the presented research by demonstrating the quality of the view synthesis resulting from the raw, non-postprocessed wide-baseline stereo disparity maps.

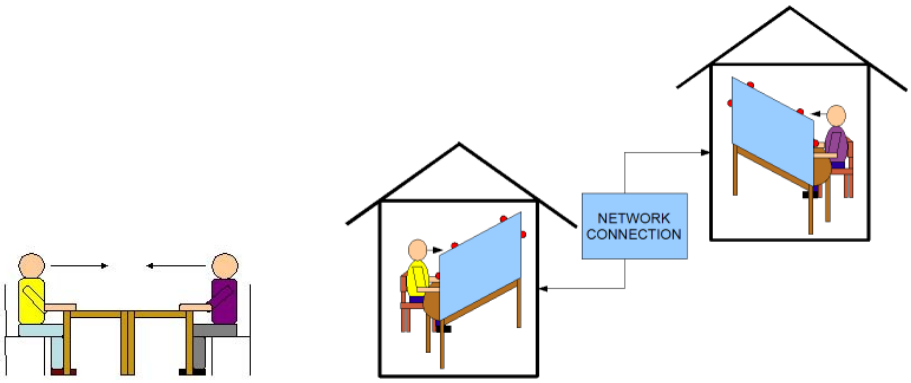
We conclude the chapter by presenting the view synthesis results based on completing the disparity map with different interpolation techniques and motivate the idea of adding *a priori* knowledge in the form of a 3-D model to complete disparity maps, which is the topic of the next chapter.

Figure 4.1: The VIRTUE videoconferencing station. 4 cameras were arranged around the screen in 2 wide-baseline stereo pairs.

4.2 VIRTUE

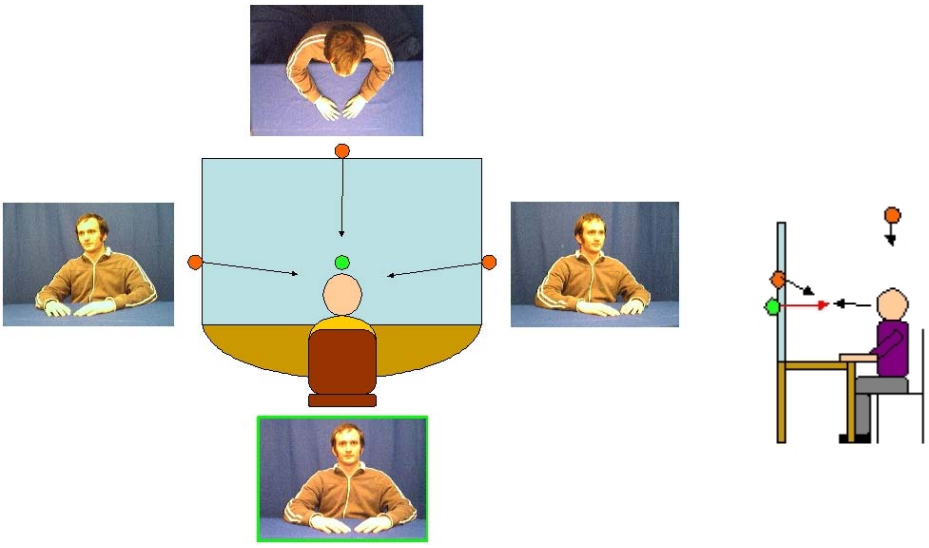
VIRTUE immersive videoconferencing setup [82, 136] made use of a large, non-transparent plasma display, with four cameras arranged as two stereo pairs around the display (see Figure 4.1). The optical axes of the cameras were pointing towards the person sitting at the table in front of the screen. Figure 4.2 illustrates the difference between the line-of-sight in a normal conversation scenario, where people are physically seated at the same table, looking at each other (Figure 4.2(a)), and the line-of-sight in a videoconferencing conversation with remote participants (Figure 4.2(b)). In the latter, the conference participant expects to see on screen what they would normally see in a conventional conversation (such as the scenario shown in Figure 4.2(a)), however this would require a camera to be placed in the middle of the screen to imitate the conversation partner's line-of-sight as closely as possible.

As the screen is non-transparent, placing the camera behind the screen is not an option. Placing it in front of the screen will disrupt the tele-presence effect with the camera appearing in the middle of the image. Immersive systems, such as VIRTUE, solve this problem by mounting the cameras around the screen, where they do not interfere with the immersive experience, whilst still being able to see the person sitting at the table, and by simulating the virtual camera in the middle of the screen by means of view synthesis. Figure 4.3 shows the example views acquired by such a setup where the real cameras, denoted by orange circles, are mounted around the screen. The corresponding virtual camera view is shown as



(a) (b)

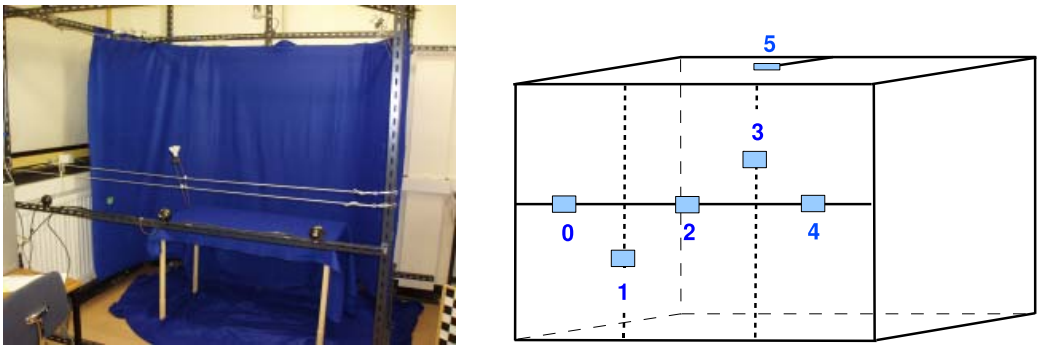
Figure 4.2: (a) Line-of-sight in a face-to-face conversation. (b) Line-of-sight in a videoconferencing conversation.



(a) (b)

Figure 4.3: (a) Example camera positions and their corresponding views in an immersive videoconferencing setup. The green circle denotes the virtual camera and the orange circles the real cameras. (b) If it physically existed, the virtual camera would have to be placed behind the screen to achieve the correct viewpoint without interfering with the immersive experience.

the green-framed view in Figure 4.3 (a). The virtual camera position is denoted by a green circle in Figures 4.3 (a,b).



(a) (b)

Figure 4.4: (a) Our camera setup used for image acquisition. (b) The schematic display of the setup with numbered cameras.

4.3 Our Camera Setup

One of the aims of our research was to produce high-quality view synthesis results with a low-cost equipment. To this end, our acquisition studio setup consisted of 6 low-cost Unibrain Fire-i^{TM}1.2 digital webcams with the IEEE-1394a (FireWire) 400 Mbps interface, mounted on a metal cube-shaped frame which surrounded a table (see Figure 4.4 (a)). The field of view of the setup was sufficient to cover the upper body of a person sitting at the table (see Figure 4.5). Due to the constrained studio space the hands were out of view if the arms were fully extended to either side. The main lighting consisted of low-cost halogen floodlights situated behind the cameras and complemented by an office-type ceiling fluorescent lighting. The resolution of the cameras was VGA 640×480 and the acquisition was performed in the RGB mode.

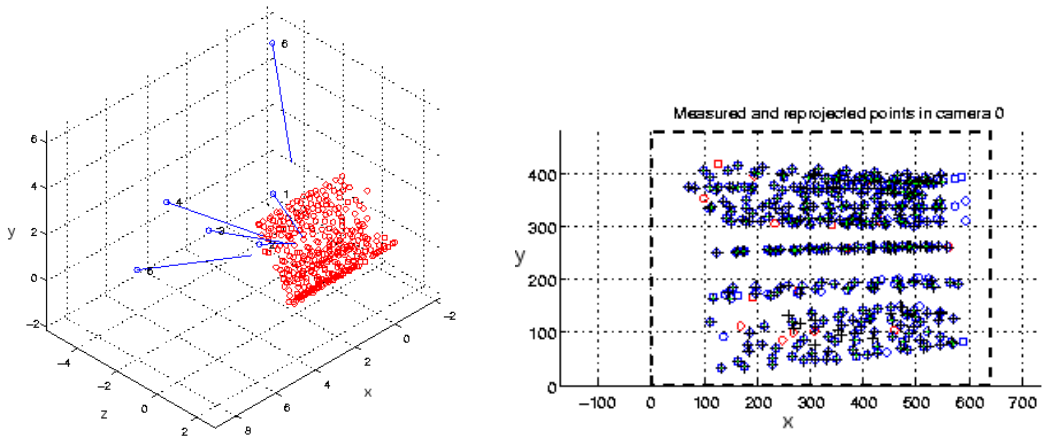
As we will show later in Chapters 5 and 8, the setup was constructed so that it could be used for human body pose estimation as well as novel view synthesis. Figure 4.4 (b) shows a schematic display of the camera setup. The top camera (camera 5) was mounted primarily for pose estimation reasons and did not play a role in later view synthesis. The cameras in front were arranged so that the person seated at the table was covered by a reference stereo pair (cameras 0 and 2) with an approximately 700 mm baseline and 40° rotation between cameras and three ground truth views (cameras 1, 3 and 4) for the novel view synthesis.

Figure 4.5: Example 6-camera snapshots acquired by our camera setup.

The cameras were calibrated using the multi-camera self-calibration method by Svoboda *et al.* [159] (described in Section 3.2.3). Due to the low cost of the setup, the cameras did not have an external trigger input and 6 cameras were connected to only two FireWire cards which saturated the bandwidth and meant that a synchronised acquisition was not possible. As a result, the decision was made to acquire multi-view snapshots of individual poses instead (Figure 4.5) and leave the sequence acquisition until later, when the synchronisation problem can be solved.

The calibration code is provided on the authors' website [159]. The calibration begins by acquiring images of a bright spot in a dark room. The authors suggest using a modified laser pointer. The main requirement is that the light source is small (contained, not a laser beam, for example), bright and well visible. Ideally, one should wave the light source in front of the cameras so as to cover the entire working volume. The bright dot must be visible in at least 3 cameras simultaneously for the particular frame set to be used in calibration. The frames where the light source is visible in 2 or fewer cameras are discarded.

The acquired frames with visible bright dots are then used as an input to the self-calibration algorithm. The algorithm extracts the point positions by fitting a 2-D Gaussian to the thresholded bright spot to find the peak of brightness in every image. Extracted image point positions are then collected in a measurement matrix W, which is factorised into the



(a) (b)

Figure 4.6: (a) Our calibrated camera setup shown with the reconstructed points used for self-calibration. (b) The reprojected points as output by the self-calibration algorithm [159].

projective motion P and structure X. The metric reconstruction, based on the concept of absolute conic, is used to convert the projective estimates to Euclidean structure and motion.

Before running the calibration algorithm, several thresholds have to be set, such as the expected size of the bright spot in pixels, the colour of the light (red, green, blue or intensity) and the number of iterations before the algorithm stops. The lens distortion is estimated with a standard routine from the Bouguet calibration toolbox [25]. The output of the calibration consists of the camera matrices and several figures showing the calibrated setup, the positions of the extracted bright dots in space and reprojected points, as illustrated in Figure 4.6. We describe the main components of the self-calibration algorithm in Section 3.2.3. A further description with implementation details can be found in [159].

4.4 View Synthesis with Original Disparity Maps

The view synthesis algorithm [93] used in the VIRTUE project relied on the dense disparity maps generated from images acquired by a wide-baseline stereo pair of cameras mounted around the videoconferencing screen. The high quality of disparity maps was fundamental to generating high-quality novel views. Real-time processing requirements dictated the use of simple and efficient area-based stereo correspondence approaches which produced in-

(a) (b)

Figure 4.7: Acquired images must be rectified before the correspondence search. (a) The original stereo pairs before rectification. (b) The corresponding rectified stereo pairs.

complete disparity maps and required post-processing before they could be used for novel-view synthesis [14].

The need for disparity map post-processing is not uncommon [14], especially in a wide-baseline setup like ours, where fast correlation-based approaches are not sufficient to produce dense disparity maps due to difficult wide-baseline imaging conditions. In this chapter, we highlight the need for disparity map post-processing in the view synthesis scenarios such as VIRTUE [136] by demonstrating the quality of the view synthesis results when using only original (*i.e.*, not in any way post-processed) disparity maps obtained using a simple and fast correlation-based stereo correspondence search on a wide-baseline (700 mm) stereo pair of images.

A number of steps are required in order to synthesise novel views from the available image data. We first perform the blue screen and background segmentation, keeping only the image information about the person in the foreground. We then rectify the images using the rectification algorithm by Fusiello *et al.* [61], described in more detail in Section 3.3. Examples of rectified stereo pairs are shown in Figure 4.7. The rectified images are used as an input to a pyramidal stereo correspondence search algorithm, described in Section 3.4.

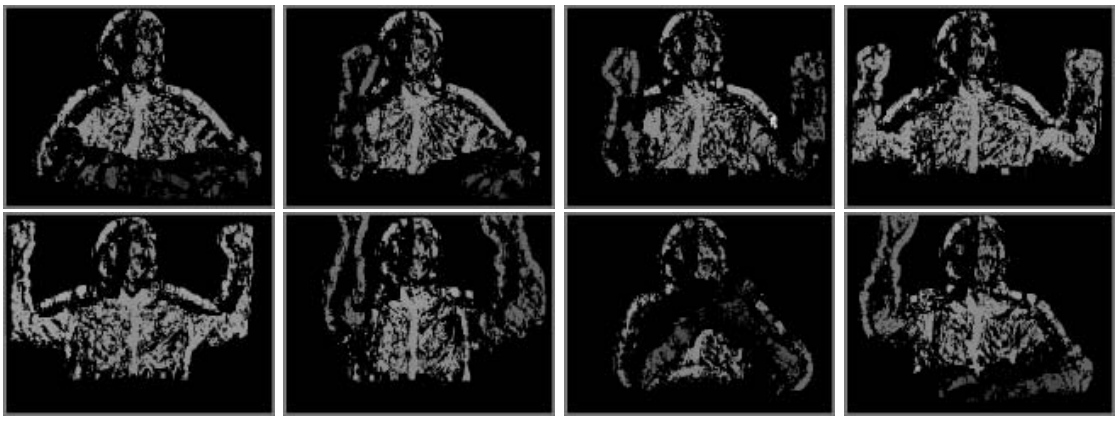


Figure 4.8: Examples of disparity maps resulting from a correspondence search algorithm after the consistency check. Cameras 0 and 2 (Figure 4.4(b)) were used as a stereo pair.

The correspondence search is performed independently on the left image with respect to the right image and on the right image with respect to the left image, producing the left-right (D_{LR}) and right-left (D_{RL}) disparity maps, respectively. The D_{LR} and D_{RL} disparity map then undergo a consistency check where only those disparity values are kept which satisfy the consistency threshold:

$$|D_{LR}(i,j) + D_{RL}(i-d,j)| \leq 1, \quad i = 1 \ldots M, j = 1 \ldots N, \tag{4.1}$$

where M and N are the disparity map row and column dimensions, respectively, and d is the disparity value in pixels.

Figure 4.8 shows the resulting disparity maps after the consistency check. Using these disparity maps in a trilinear-tensor-based novel-view synthesis gives novel views such as those shown in Figures 4.9 and 4.10. As expected, the synthesised views are very patchy, a consequence of the poor quality disparity maps.

In order to improve the results of novel view synthesis, the missing disparity information must be recovered. In the next section we look at different ways of post-processing the available disparity information to complete the missing regions. In particular, we look at different ways of interpolating the available disparity data.

Figure 4.9: Examples of synthesised views using the original disparity maps from Figure 4.8. Camera 1 (see Figure 4.4(b)) is used as the virtual camera.

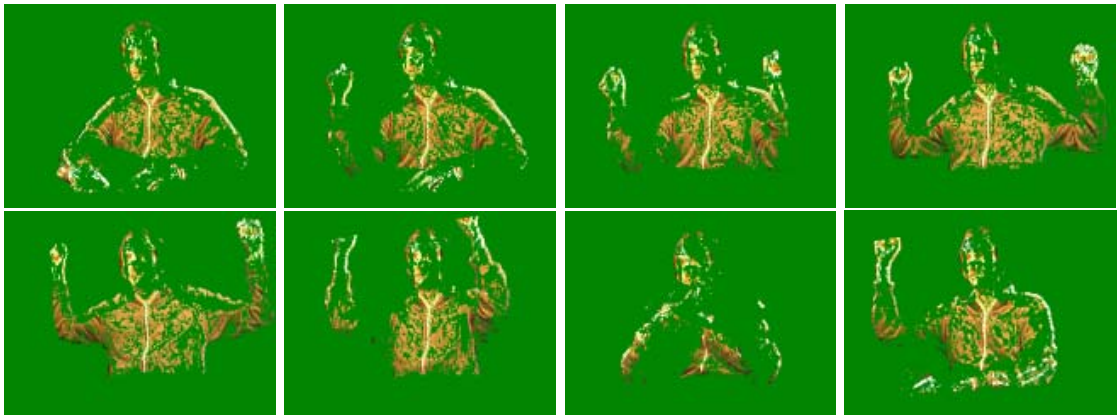


Figure 4.10: Examples of synthesised views using the original disparity maps from Figure 4.8. Camera 4 (see Figure 4.4(b)) is used as the virtual camera.

4.5 View Synthesis with Interpolated Disparity Maps

We have shown in the previous section that the correspondence search followed by the consistency check produces incomplete disparity maps which require some form of post-processing to improve their quality. The aim of post-processing is to close the gaps and provide the missing disparity information to be used for novel-view synthesis.

If no additional knowledge about the contents of the scene is available, the obvious approach to completing the patchy data is by means of interpolation.

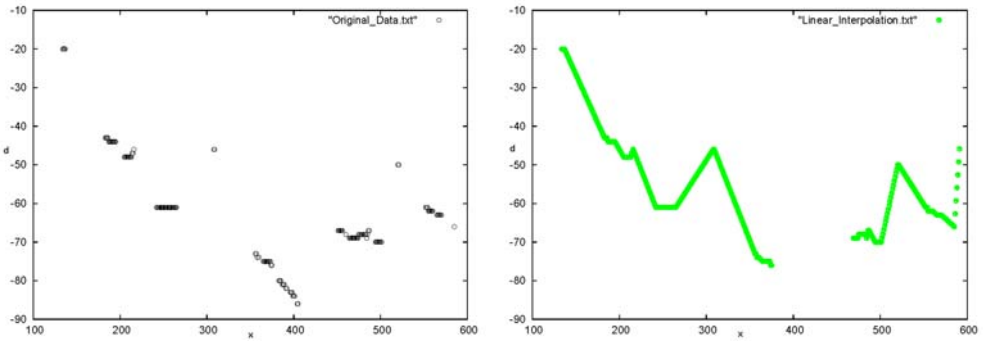
In this section we examine three different ways of interpolating the available disparity data to produce a complete disparity map. We begin with a simple row-based linear interpolation, followed by the bilinear interpolation, and finally the cubic B-spline interpolation.

We first present an overview of each interpolation technique, followed by the interpolation experiments and finally the view-synthesis results obtained by using the interpolated disparity maps.

4.5.1 Linear Interpolation

The linear interpolation algorithm proceeds by fitting a line to two data points and then sampling the line to provide missing data in between the original two data points. Given a 3-D line in parametric form, $\mathbf{x} = \mathbf{x}_0 + \mathbf{d}t$, where:

$$\begin{aligned} x &= x_0 + d_x t \\ y &= y_0 + d_y t \\ z &= z_0 + d_z t, \end{aligned} \tag{4.2}$$



(a) Original data. (b) Interpolated data.

Figure 4.11: Illustration of linear interpolation on a row of disparity data. Foreground mask restricts the interpolation to the two intervals shown.

and two input data points, $D_1(x_1, y_1, z_1)$ and $D_2(x_2, y_2, z_2)$, the direction vector $\mathbf{d} = (d_x, d_y, d_z)$ is computed as a difference vector between the two data points:

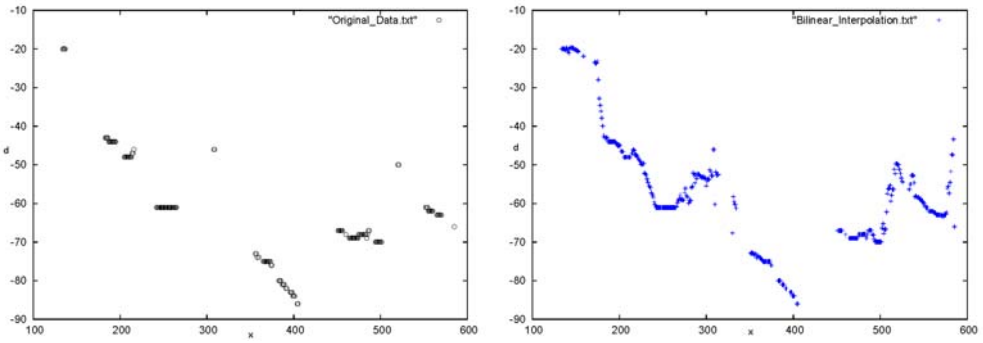
$$\begin{aligned} d_x &= x_2 - x_1 \\ d_y &= y_2 - y_1 \\ d_z &= z_2 - z_1. \end{aligned} \tag{4.3}$$

The line is then sampled between D_1 and D_2 by setting $\mathbf{x}_0 = D_1$, and evaluating the parametric equation for $t \in (0,1)$. Figure 4.11 illustrates the result of a linear interpolation on one row of disparity data. The section between the pixel positions 380 and 480 is outside the foreground mask defining the interpolation intervals and is thus not interpolated.

4.5.2 Bilinear Interpolation

Bilinear interpolation is an extension of the linear interpolation scheme to a two-dimensional case. Let $I(x,y)$ be the unknown image value at position (x,y). Given the four surrounding integer pixel positions, where the image value is known, the bilinear interpolation formula for $I(x,y)$ is given by:

$$I(x,y) = c_1 x + c_2 y + c_3 xy + c_4, \tag{4.4}$$



(a) Original data. (b) Interpolated data.

Figure 4.12: Illustration of bilinear interpolation on a row of disparity data.

where the coefficients c_1, c_2, c_3 and c_4 are given by solving the following system of equations [115]:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & x_1y_1 & 1 \\ x_2 & y_2 & x_2y_2 & 1 \\ x_3 & y_3 & x_3y_3 & 1 \\ x_4 & y_4 & x_4y_4 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}, \tag{4.5}$$

and $I_1(x_1,y_1)$, $I_2(x_2,y_2)$, $I_3(x_3,y_3)$, $I_4(x_4,y_4)$ are the 4 known surrounding image values. Figure 4.12 illustrates the result of bilinear interpolation on one row of disparity data. The row of data is the same as the one in Figure 4.11.

4.5.3 Cubic B-Spline Interpolation

Cubic B-spline interpolation is appropriate to use when the data originates from curved surfaces, where straight-line interpolation would not produce realistic results. We first look at the cubic B-spline curve interpolation, followed by the bi-cubic B-spline surface interpolation. For an overview of the B-spline theory, we refer the reader to Section 3.6.3.

Curve Interpolation. Let $C(u)$ be a 2-dimensional parametric curve $\mathbf{C}(u) = (X(u), Y(u))$ with control points $\mathbf{P}_i = (p_i^x, p_i^y)$, cubic basis functions $B_{i,3}$ and let the samples of this curve be given as $\mathbf{D}_j = (x_j, y_j)$.

We then have

$$\mathbf{C}(u_j) = \sum_{i=0}^{m} \mathbf{P}_i B_{i,3}(u_j) = \mathbf{D}_j, \tag{4.6}$$

which, in terms of individual vector components, gives:

$$X(u_j) = \sum_{i=0}^{m} p_i^x B_{i,3}(u_j) = x_j \tag{4.7}$$

$$Y(u_j) = \sum_{i=0}^{m} p_i^y B_{i,3}(u_j) = y_j, \tag{4.8}$$

where $j = 3 \ldots m+1$ while $m+1$ is the number of control points and basis functions.

The unknown values of control points $\mathbf{P}_i = (p_i^x, p_i^y)$ can be found by solving two identical linear systems of equations, one for p_i^x and one for p_i^y:

$$\begin{bmatrix} B_{0,3}(u_2) & \cdots & B_{m,3}(u_2) \\ \vdots & & \vdots \\ B_{0,3}(u_{m+2}) & \cdots & B_{m,3}(u_{m+2}) \end{bmatrix} \begin{bmatrix} p_0^x \\ \vdots \\ p_m^x \end{bmatrix} = \begin{bmatrix} x_2 \\ \vdots \\ x_{m+2} \end{bmatrix}, \tag{4.9}$$

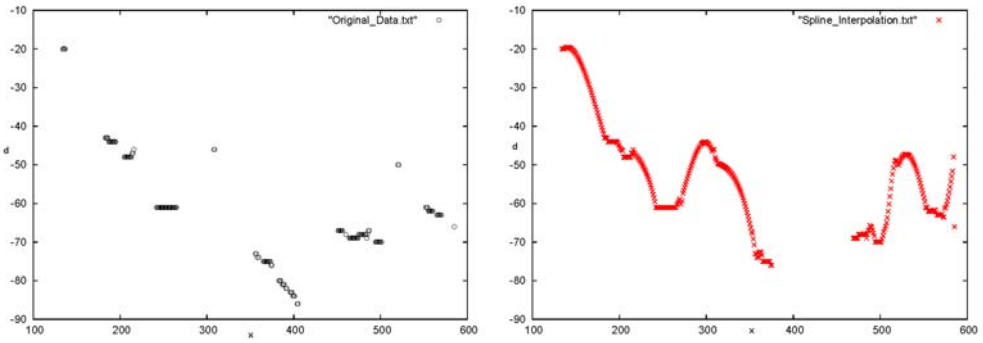
$$\begin{bmatrix} B_{0,3}(u_2) & \cdots & B_{m,3}(u_2) \\ \vdots & & \vdots \\ B_{0,3}(u_{m+2}) & \cdots & B_{m,3}(u_{m+2}) \end{bmatrix} \begin{bmatrix} p_0^y \\ \vdots \\ p_m^y \end{bmatrix} = \begin{bmatrix} y_2 \\ \vdots \\ y_{m+2} \end{bmatrix}. \tag{4.10}$$

The two extra data points required to complete the above system, $\mathbf{D}_2$ and $\mathbf{D}_{m+2}$, can be chosen in various ways with the details given in Bartels *et al.* [22]. Letting $\mathbf{D}_2 = \mathbf{D}_{m+2} = \mathbf{0}$, for example, makes the B-spline behave as a natural cubic interpolating spline.

Figure 4.13 illustrates the result of spline interpolation on one row of disparity data. Again, the row of data is the same as the one in Figure 4.11.

Surface Interpolation. The bi-cubic B-spline parametric surface $S(u,v)$ is defined as follows:

$$S(u,v) = \sum_{s} \mathbf{P}_s B_s(u,v), \tag{4.11}$$



(a) Original data. (b) Interpolated data.

Figure 4.13: Illustration of cubic B-spline interpolation on a row of disparity data.

where $B_s(u,v)$ is the bi-cubic B-spline basis function defined on a rectangular patch spanned by the u and v parametric segments.

If the control vertices are arranged in a rectangular topology, $\mathbf{P}_{i,j}$, we can write the surface as a double summation instead [22]:

$$\begin{aligned} S(u,v) &= \sum_i \sum_j \mathbf{P}_{i,j} B_{i,j}(u,v) \\ &= \sum_i \sum_j (p^x_{i,j} B_{i,j}(u,v), p^y_{i,j} B_{i,j}(u,v), p^z_{i,j} B_{i,j}(u,v)), \end{aligned} \tag{4.12}$$

where the surface basis functions $B_{i,j}(u,v)$ are now defined as a product of the B-spline curves $B_i(u)$ and $B_j(v)$, called the *tensor product B-spline:*

$$B_{i,j}(u,v) = B_i(u) B_j(v). \tag{4.13}$$

This formulation allows the B-spline surface interpolation of given sample data to be treated as two consecutive sequences of B-spline curve interpolations, one in the direction of parameter u, and the other in the direction of parameter v.

Figure 4.14 shows a comparison between the three described interpolation approaches on one row of disparity data. Linear interpolation provides a very clear-cut result which is not very realistic as the surfaces in the scene are not strictly planar. The bilinear interpolation is more in line with the nature of the data but also contains values which appear noisy.

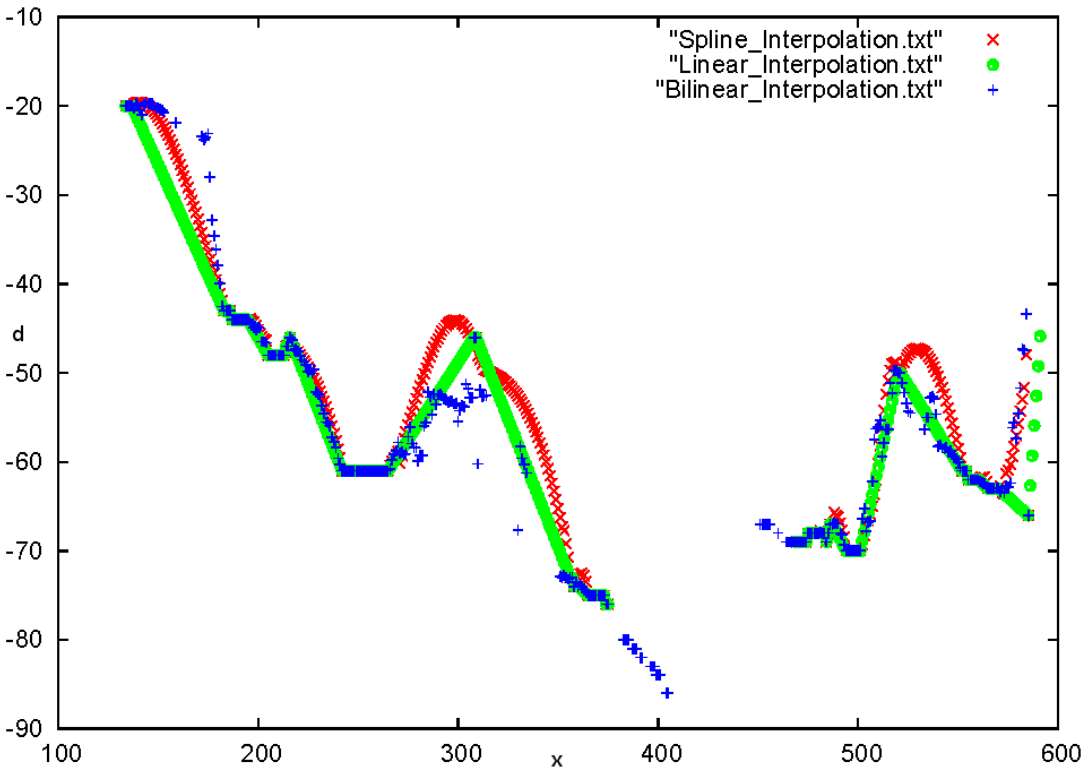


Figure 4.14: Comparison of all three interpolation algorithms on a row of disparity data.

The cubic B-spline interpolation combines the strengths of the first two methods and provides a smooth, noise-free interpolation across the gaps while also interpolating the original data and thus seems to be the most appropriate choice for disparity map interpolation.

In the next section we apply the three interpolation methods to consistency-checked disparity maps and use the resulting disparity maps to synthesise novel views.

4.5.4 Disparity Map Interpolation Results

Each method is tested on several different poses and novel views are synthesised to demonstrate the actual quality of the disparity map. Algorithm 4.1 details the steps of the interpolation algorithm, illustrated in Figure 4.15. The sequence of steps is the same for all three tested interpolation methods.

Algorithm 4.1: Disparity Map Interpolation Algorithm.

Input: Consistency-checked patchy disparity map

1. Median filter the disparity map.
2. Segment the disparity map into two distinct disparity intervals to separate arms and torso.
3. Use hand mask based on skin segmentation to label disparities belonging to hands.
4. Use foreground-background mask to constrain the interpolation to foreground only.
5. Interpolate disparities within each of the segments (hands, arms, head and torso) without crossing the segment boundaries.
6. Complete hands with simple extrapolation, using appropriate hand masks.
7. Interpolate across the torso and arms segments to close the gaps, leave the hands intact.
8. Fill in any remaining gaps with simple extrapolation.

Output: Complete disparity map.

The segmentation of disparities into two distinct intervals, separating the torso from the arms, is necessary to preserve depth discontinuities. Figure 4.16 shows examples of such segmented disparity maps.

Only when the disparity data has been interpolated within the individual segments (the hands, the arms, and then the head and torso segment), the algorithm then proceeds to interpolate across the segment boundaries. In this way we intend to preserve as much as possible of the depth discontinuity information in the disparity map, although the interpolation across the segments does introduce spurious transitions as will be seen from the

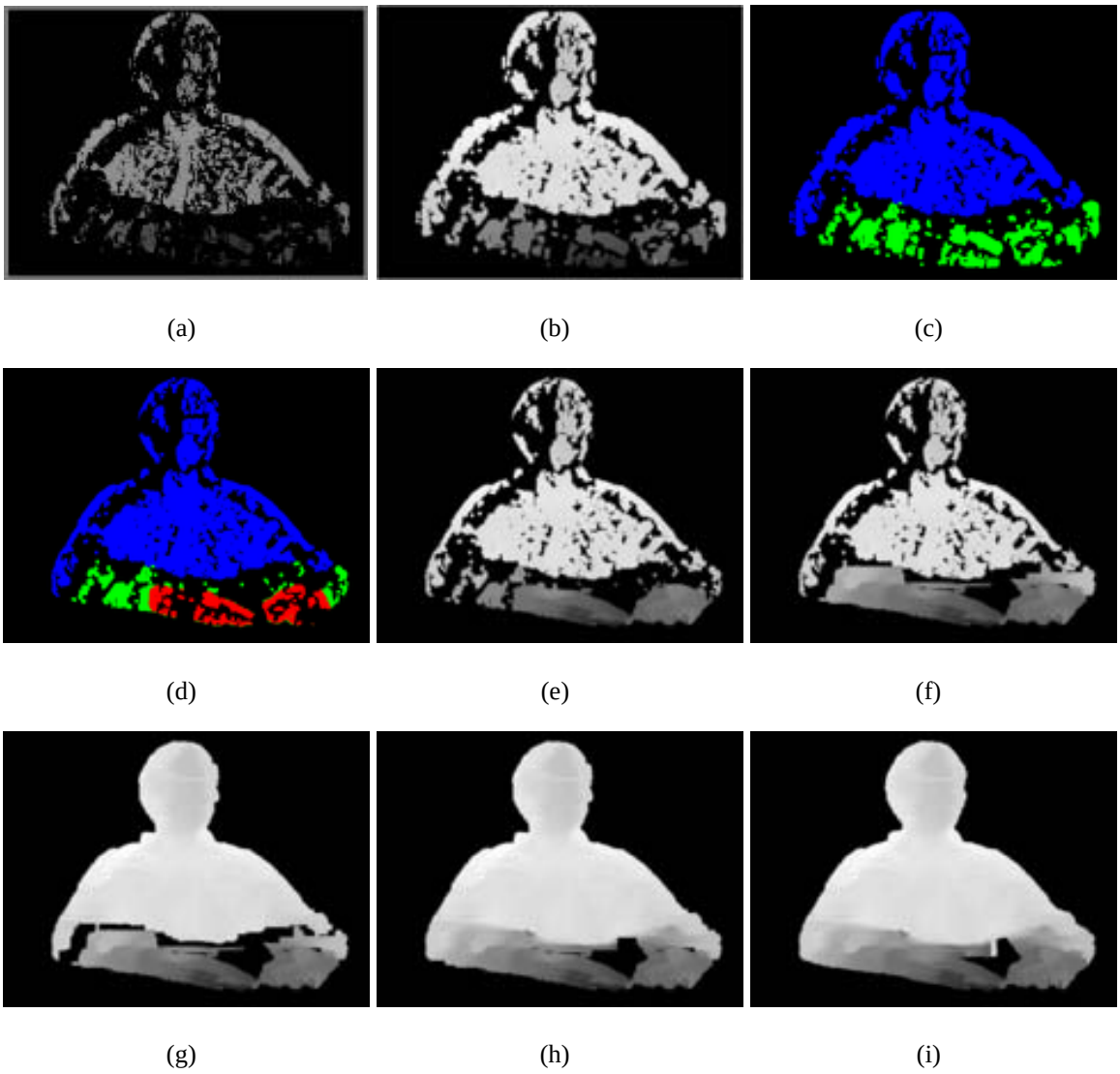


Figure 4.15: The steps of the disparity map interpolation algorithm. (a) the original disparity map; (b) median-filtered disparity map; (c) segmentation into two disparity intervals to separate hands and torso (d) hand mask is added (e) hands are interpolated (f) arms are interpolated (g) torso and head are interpolated (h) the segments are interpolated across (i) the disparity map is finalised with extrapolation.

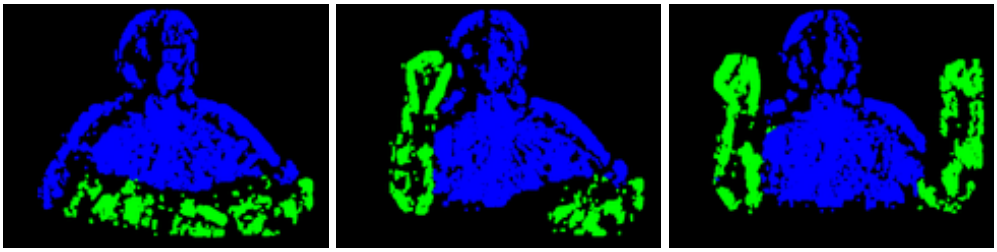
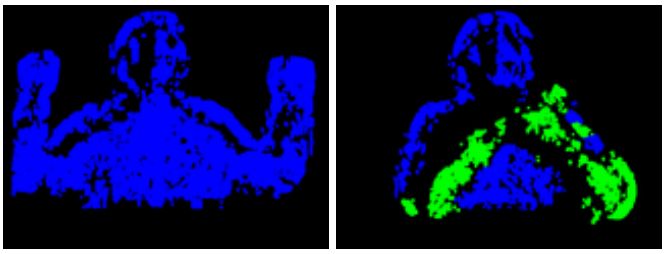


Figure 4.16: Examples of disparity maps successfully segmented into two different disparity regions, separating the torso from the hands.



(a) (b)

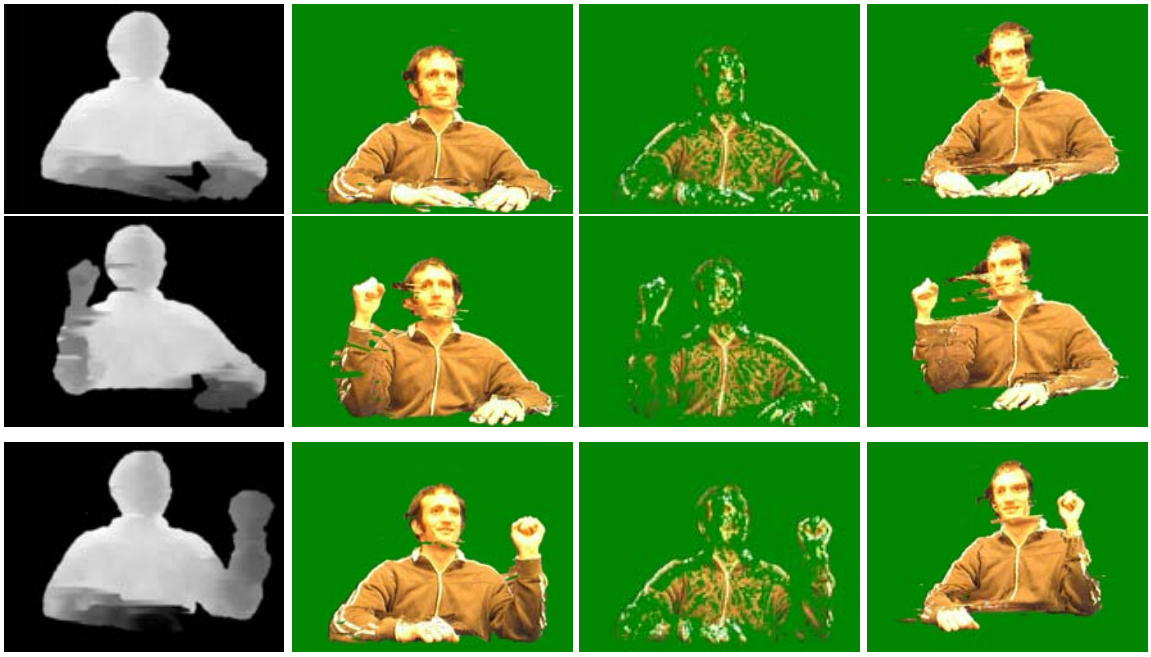
Figure 4.17: Examples of unsuccessful segmentation. (a) arms and torso share the same disparity interval - this case is not problematic as there are no occlusions causing depth discontinuities (b) arms occluding the torso - although the segmentation looks plausible, the threshold is unclear and the disparity values are not very accurate due to occlusion.

view-synthesis results.

A drawback of the presented segmentation approach is that the segmentation is not always accurate, as shown in Figure 4.17, and an incorrect segmentation adversely affects the quality of the interpolated disparity map.

Linear Interpolation

Figure 4.18 shows disparity maps for different pose examples after a simple row-based linear interpolation, and their corresponding view-synthesis results. Visual inspection confirms that the interpolation generates disparity data which improves the result of novel view synthesis compared to the original disparity map, however, the artefacts of the linear interpolation are clearly visible in the synthesised views.



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

Figure 4.18: Examples of disparity maps after linear interpolation and novel views generated using them. (a) left-right linearly interpolated disparity map; (b) novel view for virtual camera 1; (c) novel view for virtual camera 1, synthesised using the original disparity map; (d) novel view for virtual camera 4.

Bilinear Interpolation

Figure 4.19 shows the result of applying bilinear interpolation to the original disparity maps and the corresponding view synthesis results. Again, visual inspection confirms that the view synthesis is better using the interpolated disparity map, but it also improves on the result of the simple linear interpolation shown in Figure 4.18. The synthesised results do not contain as many artefacts and are in general more consistent.

Cubic B-spline interpolation

Figure 4.20 shows examples of synthesised views using the spline-interpolated disparity maps. The view synthesis results indicate that the cubic spline interpolation produces the least noisy and most consistent result from the three interpolation algorithms tested.

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

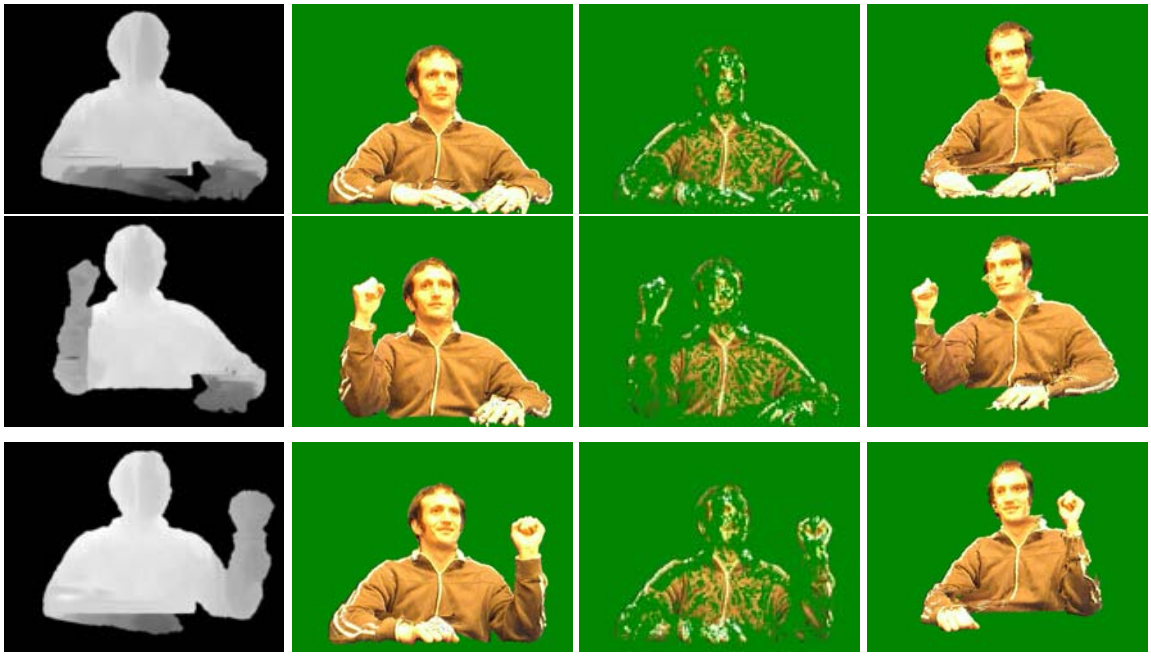
Figure 4.19: Examples of disparity maps after bilinear interpolation. (a) left-right bilinearly interpolated disparity map; (b) novel view for virtual camera 1; (c) novel view for virtual camera 1, synthesised using the original disparity map; (d) novel view for virtual camera 4.

Is Interpolation Enough?

Although an improvement has clearly been made by interpolating the available disparity data, the magnified view synthesis results shown in Figures 4.21 and 4.22 make the drawbacks of interpolation more obvious.

Virtual camera 1 lies between the reference stereo pair of cameras, slightly below the baseline. The corresponding view synthesis results shown in Figure 4.21, although with some artefacts, look realistic and believable.

Virtual camera 4 is positioned to the right of the reference stereo pair, the distance of one reference stereo pair baseline away from camera 2. The virtual view is considerably different in comparison with the reference stereo pair and the artefacts due to incorrect interpolation become much more obvious.



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

Figure 4.20: Examples of disparity maps after cubic B-spline interpolation. (a) left-right spline-interpolated disparity map; (b) novel view for synthetic camera 1; (c) novel view for synthetic camera 1, synthesised using the original disparity map; (d) novel view for synthetic camera 4.

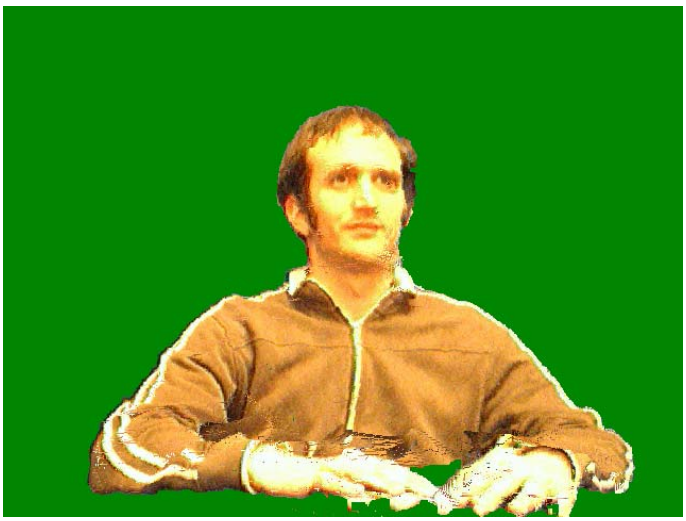
Although the cubic spline interpolation produces the best results, it is still only an interpolation trying to make sense of the available data. In particular, the interpolation across segment boundaries (step 7 in Algorithm 4.1), seems to produce unwanted artefacts such as the link between the torso and the right arm in Figure 4.22.

Removing such artefacts in a fully automatic and consistent manner is a difficult task without any prior knowledge of the scene geometry. Interpolation across segments could, for example, be avoided if the two segments clearly did not represent parts of the same surface or, in our case, segments of the same body part. Without knowing what we are actually looking at, *i.e.*, having some prior knowledge about the shape and structure of the scene, these decisions are difficult to make and do not generalise very well.

It is for these reasons that we turn our attention to post-processing the disparity data with the use of scene models, in particular human body models, and we will discuss this

(a) Linear interpolation.

(b) Bilinear interpolation.

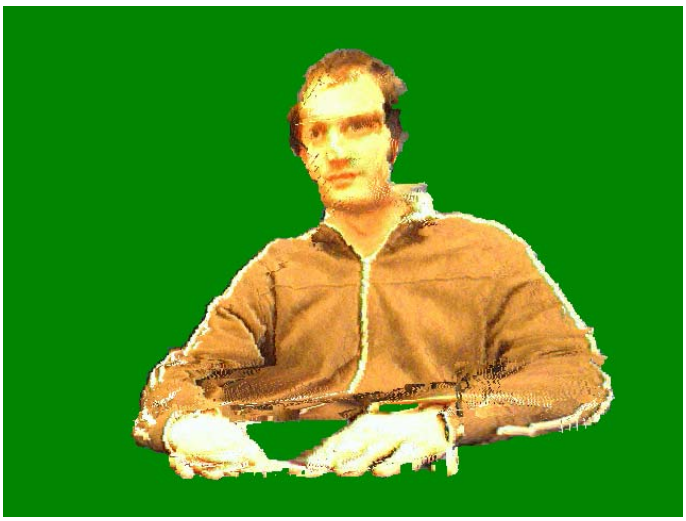


(c) Cubic spline interpolation.

Figure 4.21: The view synthesis results for three different interpolation algorithms shown magnified side-by-side for clarity. Camera 1 has been used as the virtual camera.

(a) Linear interpolation.

(b) Bilinear interpolation.



(c) Cubic spline interpolation.

Figure 4.22: The view synthesis results for three different interpolation algorithms shown magnified side-by-side for clarity. Camera 4 has been used as the virtual camera.

(a) (b) (c)

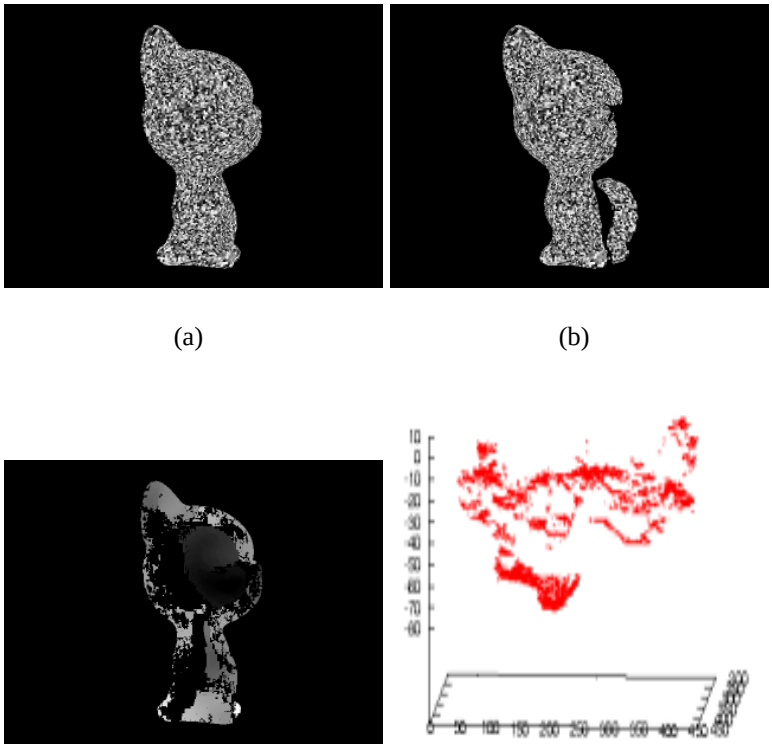
Figure 4.23: 3 views of the kitten model generated for the quantitative analysis: (a) left view; (b) centre view; (c) right view.

approach in more detail in the following chapters. In the next section, we first demonstrate on a synthetic example the view synthesis improvement achieved by using a 3-D model of the scene.

4.6 The Advantage of *A Priori* Knowledge

In the previous section, we completed the patchy disparity maps using three different interpolation techniques. We showed that by adopting this approach to disparity map completion, some improvement of the view synthesis quality is possible. We also illustrated the drawbacks of interpolation-based disparity completion. In this section, we demonstrate the difference that an *a priori* scene model makes to the view synthesis quality. We show results on a synthetic set of images and perform quantitative analysis of the view synthesis results.

To enable quantitative analysis, we generated a synthetic set of images with a known ground truth disparity. We used a 3-D mesh model of a kitten provided courtesy of UU, INRIA, by the AIM@SHAPE Shape Repository of range scan models [9] and acquired 3 views of the kitten model using a wide baseline stereo setup, as shown in Figure 4.23. Left and centre view were used as the reference stereo pair, and the right view as the ground truth for novel view synthesis.



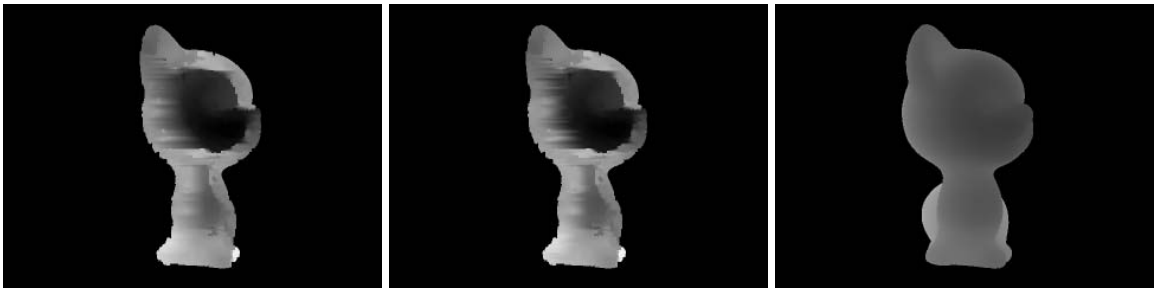
(a) (b)

(c) (d)

Figure 4.24: (a,b) Left and centre view of the checkerboard-textured kitten. Only mesh faces visible in both views simultaneously are rendered. (c) Disparity map corresponding to the left and centre view, shown as an image. (d) Disparity map shown as points in (x, y, d) space.

For the purpose of stereo correspondence search we textured the kitten model with a synthetic checkerboard texture. To reduce the number of false matches, we only rendered the faces of the kitten mesh which were simultaneously visible in both views, as shown in Figure 4.24(a,b). We used a pyramidal, window-based stereo correspondence algorithm to generate the disparity map. The resulting left-centre consistency-checked disparity map is shown as an image in Figure 4.24(c) and as data points in the (x, y, d) space in Figure 4.24(d), where x and y are the image row and column dimensions and d is the disparity. For clarity we henceforth refer to the disparity map generated by the stereo correspondence algorithm as the *original* disparity map.

For comparison with the post-processing methods discussed in Section 4.5, we completed the original disparity map with linear and cubic B-spline interpolation (Figure 4.25(a,b)).



(a) (b) (c)

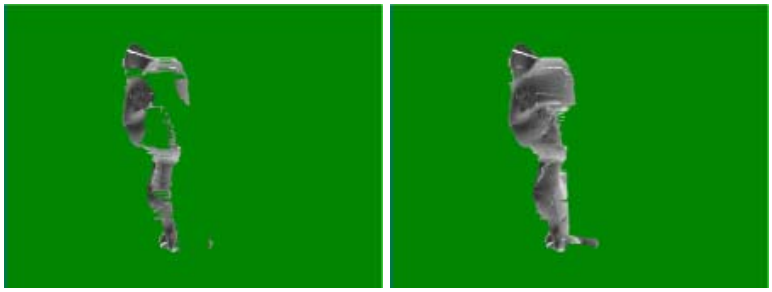
Figure 4.25: Kitten disparity maps used for quantitative analysis: (a) completed by linear interpolation; (b) completed by cubic B-spline interpolation; (c) ground truth disparity map.

We also generated a ground truth disparity map using the kitten model (Figure 4.25(c)).

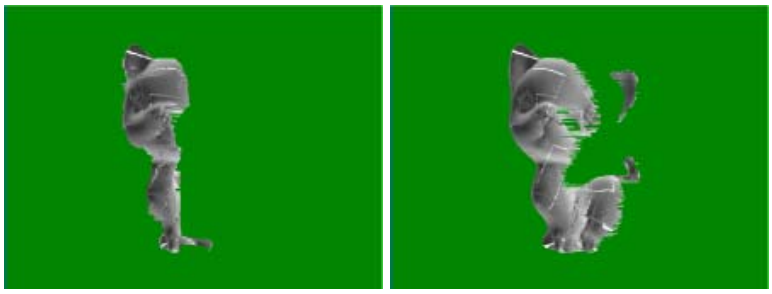
We used the original disparity map, the ground truth disparity map and the disparity maps completed by linear and cubic B-spline interpolation to synthesise a novel view of the kitten. The right view in Figure 4.23 was used as the ground truth view to which the synthesised views were compared. Figure 4.26 shows the resulting synthesised views.

The novel view synthesised using the original disparity map (Figure 4.26(a)) is patchy due to the lack of data in the original disparity map. As expected, the two novel views using the linearly interpolated disparity map (Figure 4.26(b)) and cubic B-spline interpolated disparity map (Figure 4.26(c)) are visually of a better quality as the interpolated disparities provided the missing information.

Figure 4.26(d) shows the novel view synthesised using the ground truth disparity map. Although the ground truth disparity map is shown as an image in Figure 4.25(c) to enable visual comparison with other disparity maps, the actual ground truth disparity map is a 3-D model in the $\{x, y, d\}$ space. The advantage of such ground truth disparity map is that it contains disparity values for the entire kitten model and not only disparity values for parts of the kitten visible in the left and centre view simultaneously, as is true for the other 3 disparity maps (original, linear, and cubic B-spline interpolated). This is why, using the ground truth disparity map, we can synthesise a novel view which makes use of all the tex-



(a) (b)



(c) (d)

Figure 4.26: Synthesised view of the kitten. The viewpoint is equivalent to the right view shown in Figure 4.23 (c). (a) original disparity map, (b) linearly interpolated disparity map, (c) cubic-spline interpolated disparity map, (d) model-based disparity map.

ture information available in the left and centre kitten stereo pair (Figure 4.23). This could not be done for the other 3 disparity maps and thus the ground truth disparity map produces a visually superior novel view.

We also performed a quantitative analysis of the view synthesis results, as shown in the diagram in Figure 4.27. We chose 500 3-D points on the kitten model and projected them into the left view. The points were chosen so that their disparities could be traced from the post-processed disparity maps (they were visible in the left and centre view simultaneously) but were not necessarily available in the original disparity map due to its patchy condition. For each of the 500 points we measured the distance between the point coordinates obtained by directly projecting the chosen 3-D points into the right view, and the point coordinates obtained by synthesising the right view using the post-processed disparity maps. The resulting mean pixel difference and its standard deviation are shown in Table 4.1.

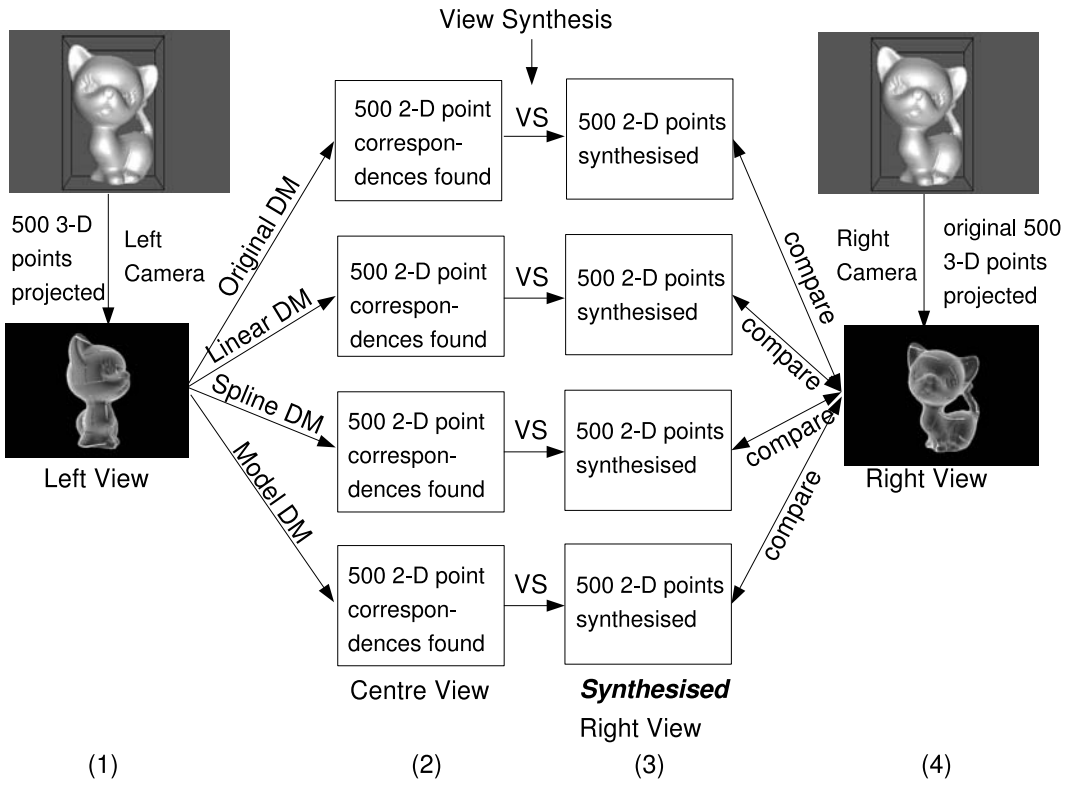


Figure 4.27: The view synthesis accuracy experiment. (1) 500 3-D points are projected in the left view. (2) Each of the 4 disparity maps is used to find corresponding points in the centre view. (3) The corresponding point pairs are used to synthesise the points in the right view. (4) The image location of the synthesised points is then compared to the image location of the 3-D points directly projected into the right view. The mean difference in the image pixel position and its standard deviation are calculated. The results are presented in Table 4.1.

Disparity map	Mean difference	Standard deviation	Missing matches
Original	2.507091	10.431232	247
Linear	7.874941	15.513824	0
Cubic	7.348909	15.989594	0
Model	0.306740	0.263713	0

Table 4.1: Disparity map post-processing error statistics. The mean pixel difference and its standard deviation were only measured for the points with known correspondences.

The following conclusions can be drawn from the results in Table 4.1:

- **Original disparity map vs the interpolated disparity maps.** The accuracy of the original disparity map is higher than that of the interpolated disparity maps. This is because the points with missing matches in the original disparity map have been left out of the calculation, whereas in the interpolated disparity maps those matches become available. If the accuracy was compared only for the points which are available in the original disparity map, there would be no difference between the original disparity map and the interpolated versions, as the latter exactly interpolate the values available in the original disparity map.

- **Interpolated disparity maps vs the ground truth disparity map.** Comparison of the interpolated disparity maps with the ground truth disparity map reveals that the interpolated values are not very accurate. The slightly higher accuracy of the cubic spline interpolation is the consequence of the cubic polynomial shape better describing the real surface than the straight line. Both interpolations only fill in the missing data in line with the nature of the interpolating function, effectively inventing values when far from the existing data points.

- **Original disparity map vs the ground truth disparity map.** The original disparity map is a result of a correspondence search algorithm using an intensity-based similar-

Figure 4.28: An example frame and its unsuccessful hand segmentation and disparity map segmentation.

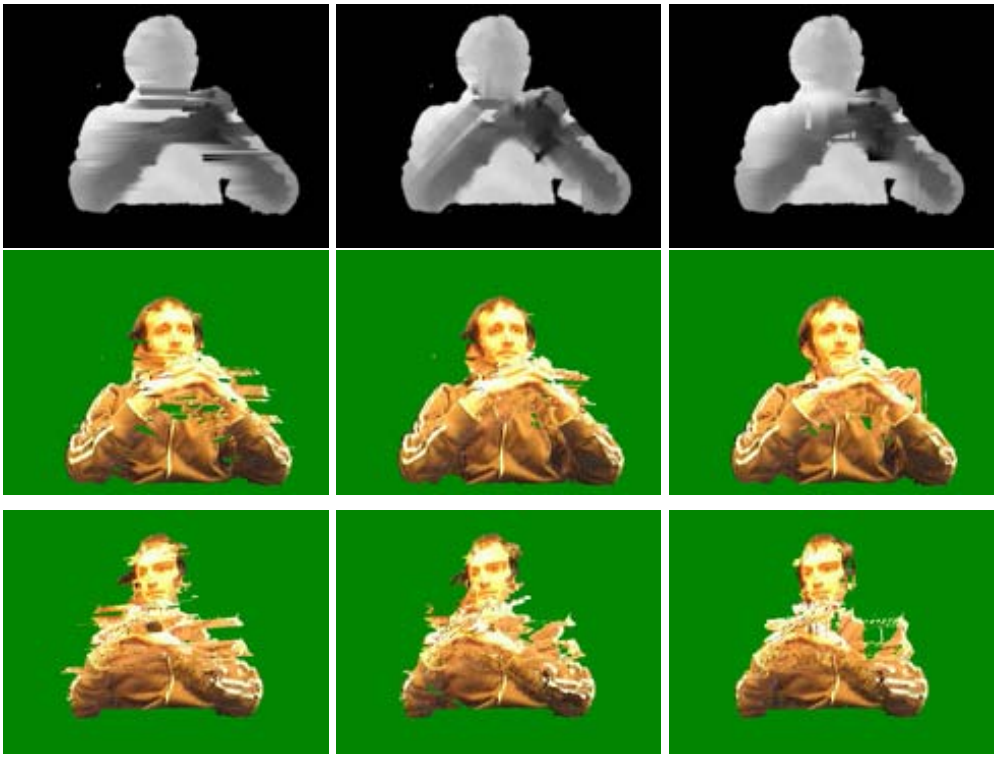
ity measure to calculate the matching points. While most of the resulting erroneous matches do not pass the consistency check, some do. Consequently, the accuracy of the original disparity map is inevitably worse than that of the ground truth disparity map.

Results in Table 4.1 suggest that using *a priori* knowledge in the form of a model of the scene, when such a model is available, improves the visual quality of the novel view synthesis and also has the potential to increase its accuracy. We say "potential" because our experiment used an exact model of the scene for the ground truth and therefore the superior accuracy was not surprising. When working with the real data, the model of the scene will be generic and not known with the same accuracy.

Despite the approximate nature of the generic model, the considerable difference in the view synthesis accuracy as well as visual quality between the model-based disparity map and the two interpolated disparity maps shown in Table 4.1 warrants the use of a model, when a reasonably accurate generic model of the scene is available.

4.7 Conclusion

The disparity map interpolation results demonstrated that interpolating the disparity data improves the quality of the disparity map and consequently the novel view synthesis. They also showed that the cubic B-spline interpolation performance is better than simple linear interpolation or bilinear interpolation. Despite the improvement demonstrated, the proposed completion algorithm has several important drawbacks, common to all suggested



(a) (b) (c)

Figure 4.29: An example of disparity map interpolation and corresponding view synthesis, where the quality of the interpolation was affected by the quality of the segmentation. (a) linearly interpolated disparity map and synthetic views for cameras 1 and 4; (b) bilinearly interpolated disparity map and the synthetic views; (c) spline-interpolated disparity map and the synthetic views.

interpolation methods.

The first drawback is that all interpolation techniques merely "guess" the missing data. Apart from the available disparity data, no *a priori* information is used to guide the interpolation. The data is simply completed according to the nature of the interpolation algorithm, as demonstrated in Figure 4.14, which also explains the superiority of the spline method, as the splines ensure that the data is interpolated with a smooth cubic curve, much more realistic than, for example, a straight line. However, given that we know that our scene consists of a person sitting at a table, it seems natural to try and incorporate this knowledge into the completion process to improve the results.

The second drawback is the reliance on the quality of the disparity map segmentation.

When the segmentation performs badly, the interpolation algorithm loses its flow and the results of the view synthesis only amplify the problem. An example of such a disparity map is shown in Figure 4.29, where not only the torso was not properly split from the arms, but also the hand segmentation did not work as the hands and head are connected into one big blob in the image (see Figure 4.28). As a consequence, the result of the post-processing and the corresponding view synthesis are not very realistic either.

Another problem stems from the fact that the original disparity data is believed to be accurate and interpolated as such. In cases where there are serious occlusions, such as the one shown in Figure 4.29, the disparity values surrounding the occluded and occluding areas are unlikely to be entirely correct. Interpolating the incorrect data creates additional artefacts in the view synthesis results. This is further exacerbated when extrapolating the missing disparities in the final step, to create a complete disparity map.

The problems described are difficult to resolve without including *a priori* knowledge about the scene. It is impossible to judge whether the interpolation is correct or not, if we have no way of telling what the data ideally should look like. We demonstrated the advantage of using a model of the scene on a synthetic example in Section 4.6. Given that our scene is fairly constrained, consisting of a person sitting at a table, we conclude that a sufficiently accurate generic model of the scene can be generated and used to improve the view synthesis quality. We explore this idea in the following chapters.

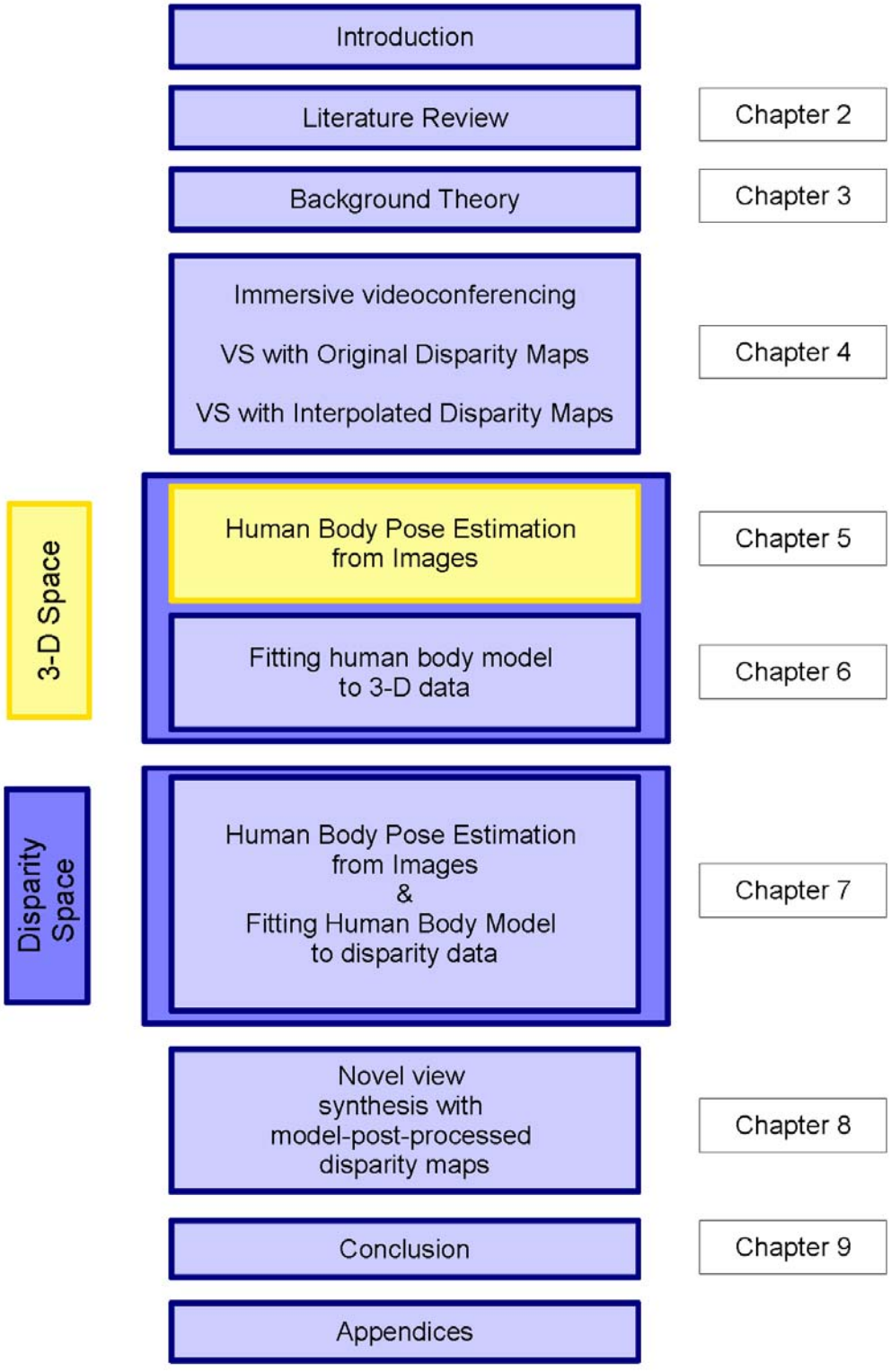


Figure 4.30: The thesis structure diagram.

Chapter 5

Articulated Human Body Model and Articulated Body Pose Estimation

5.1 Introduction

We have shown in the previous chapter that a straightforward interpolation of the available disparity data does not produce a disparity map of a high enough quality to achieve a convincing novel-view synthesis. We concluded that *a priori* knowledge in the form of a generic model of the scene had to be used to make the completion process more informed and reliable and demonstrated the advantage of using a model on the synthetic example of a kitten model.

In the remainder of this thesis we focus on the use of a generic model for completion of wide-baseline disparity maps originating from videoconferencing scenes. We assume that the scene consists of a person sitting at a table. In the virtual view, the original table is replaced by a virtual table (see Figure 4.1 in Chapter 4) and we therefore focus on modelling only the person. The required generic model of the scene is an articulated model of the human body.

In order to use the articulated human body model for disparity map completion, the body pose of the model must replicate the body pose of the imaged person. To achieve this, we have developed a pose estimation algorithm based on the Particle Swarm Optimisation

(PSO). Once the correct articulated body pose has been estimated, the resulting model can then be used to complete the patchy disparity data and improve the quality of novel view synthesis.

We postpone the description of disparity map completion with the articulated model to Chapter 6. In this chapter, we focus on the articulated pose estimation algorithm, a prerequisite for the work described in Chapter 6.

In Section 5.2, we begin by presenting our subdivision surface human body model that we use as a generic model of the scene. In Section 5.3, we then describe our articulated body pose estimation algorithm which makes use of the presented generic body model and is based on PSO. Finally, we show experimental results on images acquired by our camera setup and present a comparison of our pose estimation algorithm with an equivalent approach using simulated annealing and gradient descent in Section 5.4. We demonstrate that our PSO-based pose estimation outperforms the other two optimisation techniques, both in consistency of the pose estimates and the efficiency of the estimation process.

5.2 Subdivision Surface Generic Articulated Body Model

In computer vision, working with visually appealing models is not always the most important issue. Instead, the focus is often on simplicity, functionality and efficiency. In view of this, the models used for pose estimation can generally be divided in two broad groups. In the first group, focusing solely on articulated pose analysis, are the simpler models. These are usually collections of sticks representing the skeleton, fleshed out with geometric primitives such as, *e.g.*, ellipsoids, cylinders or conical sections (see Section 2.3.2 for details).

In the second group, the importance of the model goes beyond the articulated body pose analysis and the models used are more accurate replicas of the actual human bodies. Examples are implicit surface models, mesh models, voxel models and subdivision surface models (see Section 2.3.2). As we would like to use our generic model to complete patchy disparity map data, our work falls into this category as well.

We decided to construct the generic human body model using the subdivision surface modelling approach. The reasons for this choice are the following:

- **Smoothness under deformation.** Subdivision rules are derived from spline theory (see Section 3.6.4) and designed to generate smooth surfaces on regular base meshes unless specifically defined otherwise - special rules exist which can generate cusps and sharp edges, for example. The smoothness property is important when modelling a deformable surface, such as a human body, which changes shape through articulated movement. Unlike a standard mesh which, when deformed, is prone to "cracks", the subdivided surface maintains its smoothness even under severe deformation.

- **Multi-resolution.** As explained in Section 3.6, subdivision iteratively approximates the limit surface, with every iteration generating a smoother surface, represented by a more complex (denser) mesh. This property can be exploited to enable visualisation or network transfer even with limited computational resources.

- **Compact representation.** The model is completely specified by the coarse base mesh and the corresponding set of subdivision rules.

- **Fitting to unstructured data.** An efficient method to fit subdivision surfaces to unstructured data was developed by Litke *et al.* [96] for scientific visualisation and animation of accurate data sets. We identified this method as a good candidate for our problem of fitting a model to disparity data and successfully extended it to work with noisy and incomplete data, as we will demonstrate later in Chapter 6.

5.2.1 The Generic Upper Body Model

In a videoconferencing scenario, the lower body of the conference participant is usually occluded by the table (see Figure 5.1). We are therefore primarily interested in modelling the human body from the waist up. Note that the methods presented in this and the subsequent chapters are not in any way upper-body specific and can be extended to work with a full body model in a straightforward manner.

Figure 5.1: In a videoconferencing scene, only the upper body is normally visible as the lower body is occluded by the table.

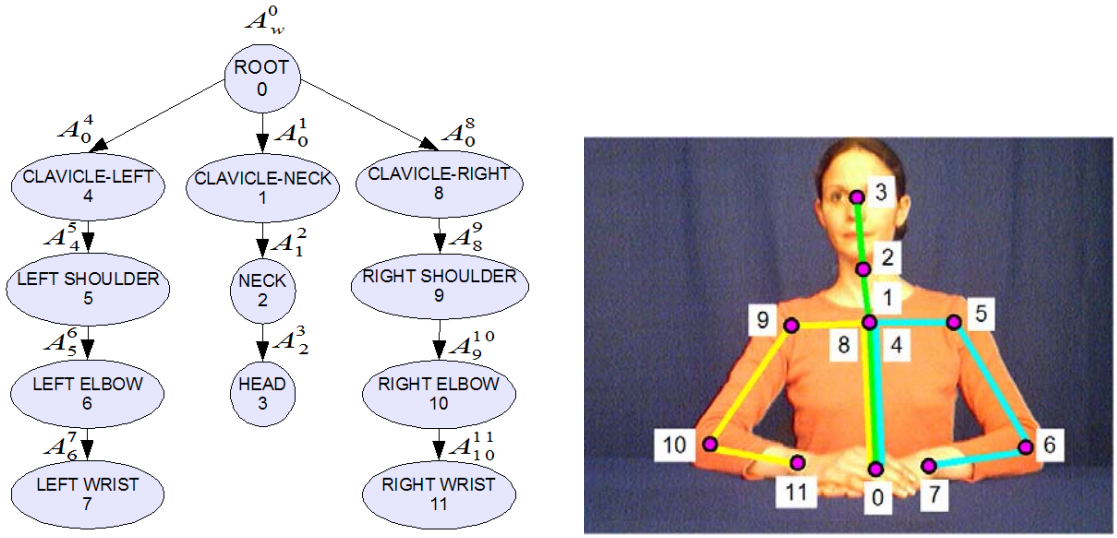
The generic upper-body model consists of two layers: the *skeleton* and the *skin*. The skeleton layer drives the articulated movement and is represented by a kinematic chain approximating the degrees of freedom (DOF) of the human body. The skin is modelled as a Catmull-Clark subdivision surface[32, 186] which is attached to the skeleton and moves and deforms with skeleton articulation. We now present each layer in more detail.

Skeleton

The skeleton layer imitates the bone structure of the human body. It consists of one 6-DOF root joint, with 3 rotational and 3 translational DOF, and 11 pure rotational joints with varying numbers of DOF, connected by rigid links. In total, we model the upper body with 20 DOF. The hierarchical structure of the skeleton layer is modelled with a kinematic tree, a collection of simple open kinematic chains with a common root joint.

A kinematic chain is defined as an assembly of links connected in a series through joints, the output motion of which at any chosen point only depends on the motion parameters of the joints. A kinematic chain is said to be open if there is no joint connecting the first and last links inside the chain. When the role of the root joint and the end joint of an open kinematic chain is fixed, as is the case in our upper body structure, such a chain is called a simple open kinematic chain [179].

The motion parameters of the joints are conveniently expressed as 4×4 homogeneous transformation matrices encoding the information about the position and orientation of every joint with respect to its parent joint in the kinematic hierarchy. The structure of the



(a) (b)

Figure 5.2: (a) Kinematic tree structure of the upper body model and the corresponding transformations. (b) Kinematic chains shown overlaid on the upper body.

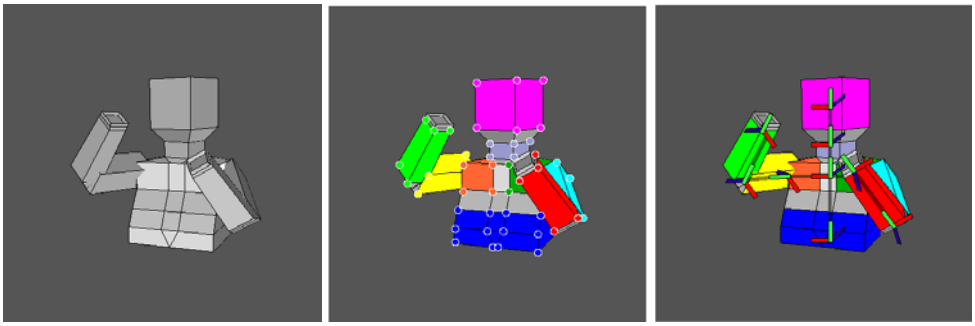
kinematic tree dictates the necessary transformations and the order in which they must be multiplied together to generate the required articulated motion.

Let pairs of indices (i, j) denote the parent-child pairs of nodes in the kinematic tree, $Q = \{(i, j)\}$ denote a set of all such pairs for a given kinematic tree and $N = |Q|$ denote the cardinality of Q. We define the skeleton layer as a set of transformations:

$$Skeleton = \{A_w^0, A_{i_1}^{j_1}, A_{i_2}^{j_2}, \ \ldots \ , A_{i_N}^{j_N}\}, \ \ \forall (i, j) \in Q, \tag{5.1}$$

where A_i^j is a homogeneous transformation matrix encoding the position and orientation of the child joint coordinate system j with respect to the parent joint coordinate system i. The transformation A_w^0 denotes the transformation between the world coordinate system and position and orientation of the root joint coordinate system.

Figure 5.2(a) shows the kinematic tree for our upper body model. It contains 12 nodes corresponding to 12 different joints. At the root of the tree is the root joint. Its rotational and translational movement with respect to the world coordinate system affects the position and orientation of all other joints in the hierarchy. From the root joint, the tree branches out into three simple open kinematic chains, one for each arm and one for the neck and head. Each



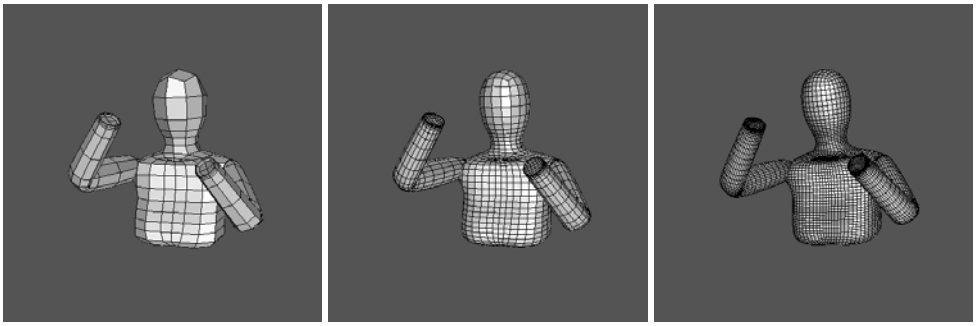
(a) (b) (c)

Figure 5.3: The skin is modelled as a subdivision surface. (a) The base mesh. (b) Different colours denote parts of the base mesh influenced by different joints. (c) Approximate skeleton position for this particular pose and the corresponding local coordinate systems.

kinematic chain passes through its corresponding clavicle joint. The three clavicle joints have the same location but different orientations with respect to the root joint. From the clavicle joint, the kinematic chains proceed independently towards their corresponding end joints (head joint, left wrist joint, right wrist joint). Figure 5.2(b) shows the corresponding three different kinematic chains overlaid on top of the upper body. Each chain is shown in a different colour.

Skin

The skin layer represents the second layer of the generic upper-body model, connected to the skeleton through the joints' local coordinate systems. It is modelled as a subdivision surface with the base mesh shown in Figure 5.3(a). Each of the joints controls the movement of a specific part of the base mesh by influencing the position and orientation of the base mesh vertices through its local coordinate system. Different colours in Figure 5.3(b) illustrate parts of the base mesh influenced by different joints. Figure 5.3(c) shows the approximate position of the underlying skeleton with local coordinate systems belonging to individual joints. The coordinate system x, y and z axes are colour coded as red, green and blue, respectively. To avoid confusion, only the coordinate systems belonging to joints visible from this viewpoint are shown. The clavicle local coordinate system is also only displayed for the neck-head kinematic chain. When the base mesh is subdivided to produce a smooth skin surface, each joint influences an area of the skin associated with the base



(a) (b) (c)

Figure 5.4: The skin subdivision levels. (a) 1 level. (b) 2 levels. (c) 3 levels.

mesh that it controls.

Formally, the skin can be defined as a set of transformation matrices A_i^j from the skeleton layer (see Equation 5.1) combined with the corresponding sets of base mesh vertices $\{X_{A_i^j}\}$ directly influenced by each of these transformations:

$$Skin = \{(A_w^0, \{X_{A_w^0}\}), (A_{i_1}^{j_1}, \{X_{A_{i_1}^{j_1}}\}), (A_{i_2}^{j_2}, \{X_{A_{i_2}^{j_2}}\}), \ \ldots \ , (A_{i_N}^{j_N}, \{X_{A_{i_N}^{j_N}}\})\}, \tag{5.2}$$

where

$$X_{A_w^0} \cap X_{A_{i_1}^{j_1}} \cap X_{A_{i_2}^{j_2}} \cap \cdots \cap X_{A_{i_N}^{j_N}} = \emptyset. \tag{5.3}$$

In order to generate a smooth skin surface, all vertices must first be transformed from their corresponding local coordinate system, where they are defined, into a common global coordinate system, such as the world coordinate system. This is achieved by constructing the appropriate kinematic chain transformation for each group of vertices $X_{A_i^j}$. For example, the set of vertices $X_{A_{i_k}^{j_k}}$ is transformed into the world coordinate system by concatenating all parent-child transformations which, according to the kinematic tree, lead to the joint j_k:

$$X^w_{A_{i_k}^{j_k}} = A_w^0 \cdot A_0^{j_1} \cdot \ \ldots \ \cdot A_{i_k}^{j_k} \cdot X_{A_{i_k}^{j_k}}. \tag{5.4}$$

The base mesh M_0, *e.g.*, the one shown in Figure 5.3 (a), then consists of all vertices V, expressed in a common coordinate system, connected with edges into quadrilateral faces F:

$$M_0 = \{V, F\}, \tag{5.5}$$

where

$$V = \{X^w_{A^j_i}\}, F = \{(m^w_a, m^w_b, m^w_c, m^w_d)\}, \text{ and } m^w_{\{a,b,c,d\}} \in V. \tag{5.6}$$

Finally, to obtain the smooth limit surface M_∞ of the base mesh, *i.e.*, the skin in Figure 5.4, the base mesh is repeatedly subdivided:

$$M_\infty = \lim_{k\to\infty} S^k M_0, \tag{5.7}$$

where S is the Catmull-Clark subdivision operator described in Section 3.6 and S^k denotes its k-th application.

Figure 5.4 shows the first three levels of subdivision starting from the base mesh in Figure 5.3 (a). In practice, only a small finite number of subdivision levels k is required. For the purpose of this work, $k = 2$ levels were sufficient for the pose estimation described in this chapter and $k = 3$ for the model fitting described later in Chapter 6.

5.2.2 Hands

Hand gestures form an important part of the conversation and as such have an important role in the videoconferencing situations. When we chose to use an *a priori* model of the human body to provide information about the missing data, the hands proved a very complex articulated object, difficult to estimate and model accurately within the presented setup.

Instead of approximating them with a model of a simpler shape, which provided only a partial solution that often caused additional difficulties, we decided to not include them in the model at this stage and instead concentrate on modelling and completing the rest of the disparity map. Accurate modelling and articulated pose estimation of the hands is an established research area in itself and was within the scope of this work identified as a future challenge.

5.3 Human Body Articulated Pose Estimation

In the previous section, we presented the generic model of the upper body used to illustrate an articulated upper-body pose. In this section we present our algorithm which utilises this

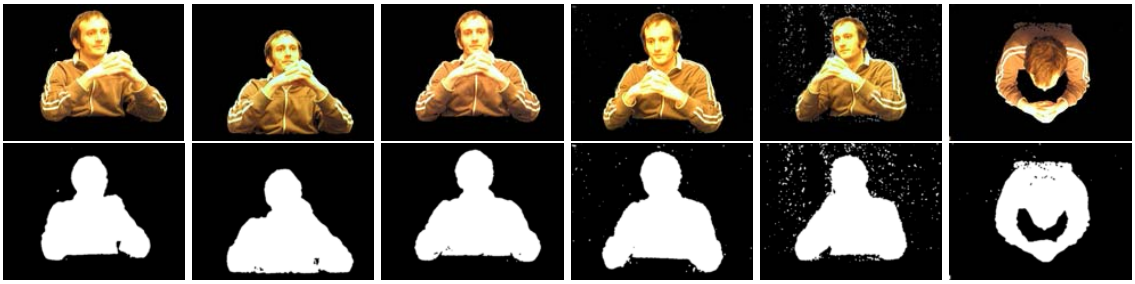


Figure 5.5: A set of 6 images acquired from 6 different viewpoints. The images are foreground-background segmented (the upper row) and binarised to obtain silhouettes used as constraints in the optimisation (the lower row).

model for articulated pose estimation from multiple views using PSO. We first describe the cost function and PSO parameter settings and then provide a detailed description of the pose estimation algorithm.

5.3.1 Optimisation Cost Function

The input to the articulated pose estimation algorithm is a set of still images taken from multiple viewpoints. The images are foreground-background segmented using blue-screen segmentation combined with background subtraction. The foreground is then converted into silhouettes which serve as constraints for the pose estimation (see Figure 5.5).

The individual pose estimates are demonstrated through the generic 3-D body model. The assumption is that when the pose expressed by the model closely matches the pose of the person in the original set of images, the silhouettes generated by the model also closely match the silhouettes extracted from the original images.

The process of estimating the body pose is therefore formulated as an optimisation problem, *i.e.*, finding the pose which maximises the overlap of the two silhouette sets. The cost function compares the silhouettes extracted from the original images with the silhouettes generated by the model in its current pose. Let us assume that the original images can be acquired from N different viewpoints. Each image is foreground-background segmented and binarised to obtain a silhouette. Let the images containing the *original* silhouettes be denoted as I_i^o, $i = 1...N$. Similarly, let I_i^m, $i = 1...N$ denote images of the

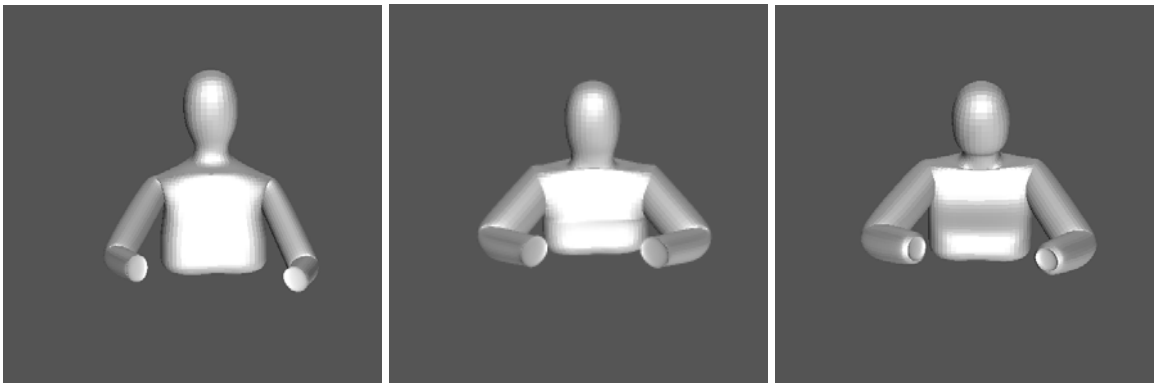


Figure 5.6: Three different generic models used to estimate the body pose for three different test subjects. The limb lengths as well as the mesh dimensions have to be adjusted.

model silhouettes. The cost function can then be written as follows:

$$E = \sum_{i=1}^{N} \frac{\omega_i}{Z_i} \sum_{1}^{row} \sum_{1}^{col} (I_i^o \,\&\, I_i^m), \tag{5.8}$$

where *row* and *col* denote the image dimensions, *i.e.*, number of rows and columns, respectively, and & denotes the logical *AND* operation. Coefficients Z_i are the normalisation constants obtained by counting the number of silhouette pixels in every original image. Weights ω_i control the contribution of every view to the total error count.

5.3.2 Pose as a PSO Particle

In PSO, each particle represents a potential solution in the search space. Our search space is the space of all plausible skeleton configurations, expressed by the motion parameters of individual joints. The individual particle's position vector in the search space is specified as a vector containing the joint parameters:

$$X_i = (root_x, root_y, root_z, \alpha_x^0, \beta_y^0, \gamma_z^0, \alpha_x^1, \beta_y^1, \gamma_z^1, ..., \gamma_x^N), \tag{5.9}$$

where $root_x, root_y, root_z$ denote the position of the root joint (first joint in the kinematic tree) with respect to the reference (world) coordinate system, and $\alpha_x^i, \beta_y^i, \gamma_z^i$ refer to rotational degrees of freedom of joint *i* around the *x*, *y*, and *z*-axis, respectively.

Custom Model Dimensions

In order to assist the pose estimation, the dimensions of the generic model have to be adjusted to individual people for which the pose is being estimated. This involves adjusting

the limb lengths and the shape of the mesh to better capture the variation in people's body shapes and sizes. The model adjustments have to be made only once for every individual. For the purpose of this work, we adjust the model to each person manually before attempting pose estimation. Figure 5.6 shows models used for pose estimation of different test subjects.

5.3.3 PSO Parameter Setup

In Section 3.8 we presented the generic PSO algorithm and described the role of its parameters. We now discuss the actual parameter values, used in our pose estimation experiments.

Inertia weight parameter

Shi and Eberhart [143] addressed the influence of different inertia values on the exploratory abilities of the swarm. They tested inertia values in the interval $[0, 1.4]$ and found that, for a constant inertia value, a medium value of w, *i.e.*, $0.8 < w < 1.2$, had the best chance of finding the global optimum while also requiring a moderate number of iterations. Large values of w, *i.e.*, $w > 1.2$, made PSO behave more like a global search method always trying to exploit new search areas.

In this work, we use a time-varying inertia weight which gradually changes the behaviour of the swarm from global to local search. We model the change over time with an exponential function which allows us to use a constant sampling step to guide the swarm from exploration to exploitation:

$$w(x) = \frac{A}{e^x}, \quad x \in [0, \ln(10A)], \tag{5.10}$$

where A denotes the starting value of w when $x = 0$. The optimisation terminates when $w(x)$ falls below 0.1. The sampling variable x is incremented by $\Delta x = ln(10A)/N$, where N is the desired number of inertia weight changes.

The swarm is allowed to explore the search space with a particular inertia value for so long as every move of the swarm improves the current global optimum estimate. As soon as an iteration fails to improve the estimate, the value of the sampling variable increases and

JOINT (index)	DOF
Root location (root)	3
Root orientation (0)	3
Clavicle-neck orientation (1)	2
Clavicle-left orientation (2)	2
Left Shoulder orientation (3)	3
Left Elbow orientation (4)	1
Clavicle-right orientation (5)	2
Right Shoulder orientation (6)	3
Right Elbow orientation (7)	1
TOTAL	20

Table 5.1: Degrees of freedom (DOF) of the upper body model

the inertia weight value decreases accordingly. This forces the swarm to identify possible optimum regions at the very beginning, then focus on the best few, and eventually settle down in the most promising region and converge to the global optimum.

Social and Cognition Parameters

In our experiments, the values of the constants c_1 and c_2 for the social and cognition parameters $\varphi_1 = c_1 rand_1(0,1)$ and $\varphi_2 = c_2 rand_2(0,1)$ were both set to integer 2, as first recommended by [143, 88], which on average made the weights for social and cognition components of the swarm equal to 1. We did not experiment with the social and cognition bias that can be introduced by manipulating the values of φ_1 and φ_2.

5.3.4 The Hierarchical and Non-Hierarchical Search

We developed two variants of the pose estimation algorithm: the *non-hierarchical* pose estimation and the *hierarchical* pose estimation.

Our upper-body pose was defined using 20 parameters shown in Table 5.1. The non-

hierarchical algorithm optimised all parameters simultaneously. Optimising the entire configuration at once amounts to searching for an optimum of a multi-modal function in a 20-dimensional search space, which is a challenge for any optimisation method and our initial experiments with the PSO confirmed that this was indeed the case. The complexity of the cost function would at the very least require a large swarm size which was not feasible in our approach as the cost function became prohibitively expensive with the increase in the number of particles and consequently the time needed for PSO to produce a pose estimate became unacceptably long.

We decided to instead exploit the nature of the problem and formulate the search in a hierarchical manner, taking advantage of the hierarchical structure of the kinematic tree and its chains (see Figure 5.2). The hierarchy means that the positions of the joints lower in the chain are constrained by the configurations of the joints higher in the chain and we can optimise for the joint rotations in a hierarchical manner, thereby reducing the complexity of the search. Depending on the level of articulation of individual joints and dependency between them, the optimisation can be done one joint at a time or by grouping several joints together.

Although the non-hierarchical search is too complex for estimating the pose from scratch, it is useful as a post-processing step to hierarchical optimisation. Due to its sequential structure, the hierarchical optimisation is prone to error propagation when the pose of the joints early on in the hierarchy is not optimised accurately enough. In such cases, the non-hierarchical search can be initialised around the result of the hierarchical search and used to correct for the error propagation. The complexity of the non-hierarchical search is much smaller in this case, as it is initialised relatively close to the correct solution.

In the remainder of this section we first present the hierarchical approach in more detail and then describe the algorithm for both hierarchical and non-hierarchical search.

Hierarchical Approach

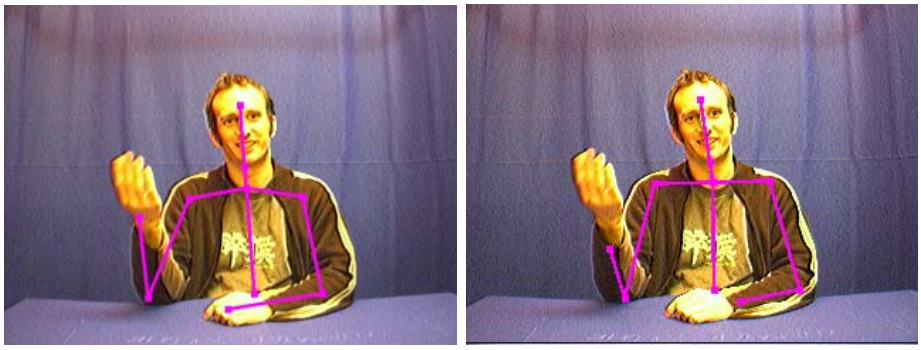
We perform the hierarchical optimisation in 7 steps, detailed in Table 5.2. First, we optimise the location of the skeleton in space, *i.e.*, the location of the root joint, followed by the

TORSO (**1**) Root location 3DOF: $root_x, root_y, root_z$ (**2**) Root orientation 3DOF: $\alpha_x^0, \beta_y^0, \gamma_z^0$	**RIGHT UPPER ARM** (**5**) Clavicle-right orientation + Right shoulder orientation 4DOF: $\alpha_x^5, \gamma_z^5, \alpha_x^6, \gamma_z^6$
NECK & HEAD (**3**) Clavicle-neck orientation 2DOF: α_x^1, γ_z^1	**LEFT LOWER ARM** (**6**) Left shoulder orientation + Left elbow orientation 2DOF: β_y^3, α_x^4
LEFT UPPER ARM (**4**) Clavicle-left orientation + Left shoulder orientation 4DOF: $\alpha_x^2, \gamma_z^2, \alpha_x^3, \gamma_z^3$	**RIGHT LOWER ARM** (**7**) Right shoulder orientation + Right elbow orientation 2DOF: β_y^6, α_x^7

Table 5.2: 7 steps in the hierarchical optimisation

root joint orientation. These are both 3 DOF optimisations. Once the skeleton is positioned in space, we optimise the neck and head sub-chain, for which we only use 2 DOF in the clavicle neck joint to model the tilt of the head. The movement of the clavicle-left and clavicle-right joint on their own does not produce enough variation in the silhouette shape to be optimised individually. Therefore, in the next step, we combine the left clavicle joint with two rotational dimensions of the shoulder joint and optimise the parameters of the left upper arm, a 4 DOF optimisation. Likewise, we then optimise the right upper arm, again 4 DOF. At the end we are left with the left and right lower arm, each modelled with 2 DOF.

When optimising joint parameters hierarchically, the joints lower in the hierarchy mislead the silhouette overlap count as they contribute to it despite of not having been optimised yet. We avoid this by deforming the subdivision body model so that at a particular stage of the optimisation only those body parts which are currently optimised, or have already been optimised, are visible.



(a) (b)

Figure 5.7: (a) Pose estimation result under the influence of error propagation in the hierarchical approach. (b) Corrected estimate after initialising the particles around the solution found in (a).

Using the hierarchical approach as described has a disadvantage in cases where the pose estimation is not very accurate early on in the hierarchy. In those cases the erroneous estimate propagates throughout the hierarchy and influences the quality of the subsequent estimates (see Figure 5.7). When the silhouette information is sufficiently articulated and unambiguous, this can be corrected by running another non-hierarchical optimisation on the whole kinematic chain at once, by initialising the particles around the hierarchical estimate and letting them settle into a new optimum. We found that, most of the time, the slight misalignment of the torso which triggered the error propagation, did not really need correcting itself and we instead focused the effort on the arms only. To make this step more efficient, we optimised each arm separately as a 6 DOF optimisation.

5.3.5 PSO Algorithm

Algorithms 5.1 to 5.5 illustrate the step-by-step hierarchical and non-hierarchical pose estimation with PSO, used to obtain results shown in this chapter. Both hierarchical and non-hierarchical approach start with the initialisation step, where the lengths and weights of joints which are not being optimised at this stage are set to zero. The particles are initialised in those dimensions of the search space, which are being optimised at this stage. The non-hierarchical optimisation only performs one loop, optimising all parameters simultaneously (all weights are set to 1.0 and all limbs are shown), while the hierarchical optimisation goes through 7 stages, as described in Table 5.2.

Algorithm 5.1: Hierarchical Pose Estimation.

Initialisation: Algorithm 5.4

Set joint group j = 0.

Set number of joint groups $= J + 1$.

repeat

Configure particle weights and model limb lengths for joint group j.

Initialise inertia weight to $w = A$ as in Equation (5.10).

repeat

PSO optimisation step: Algorithm 5.5.

until $w < 0.1$

$j++$.

until $j = J$

Result: Global best particle of the last iteration.

Algorithm 5.2: Non-Hierarchical Pose Estimation.

Initialisation: Algorithm 5.4.

repeat

PSO optimisation step: Algorithm 5.5.

until *inertia* < 0.1

Result: Global best particle of the last iteration.

Algorithm 5.3: Particle Fitness Evaluation.

1. Construct the subdivision body model for the particle.
2. Generate silhouettes for the subdivision body model.
3. Estimate the silhuette overlap using Equation (5.8).

Algorithm 5.4: Initialisation.

1. Set the starting inertia value $w = A$, as in Equation (5.10).
2. Set the inertia counter $x = 0$ and $\varphi_1 = \varphi_2 = 2.0$.
3. Initialise the weights and limb lengths.
4. Initialise the swarm in the search space dimensions specified by weights.
5. for $(1 \leq i \leq$ Swarm Size)

   Evaluate fitness of particle i: Algorithm 5.3.
6. Set all particles' initial position as their personal best.
7. Identify the best particle in the swarm as the global best.

Algorithm 5.5: Optimisation Step.

1. Move the swarm according to Equation (3.76).
2. for $(1 \leq i \leq$ Swarm Size)

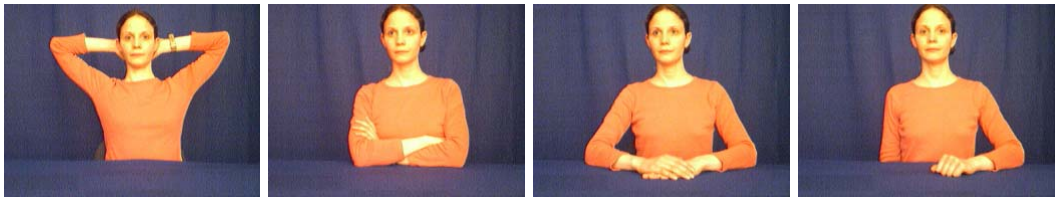
   Evaluate fitness of particle i: Algorithm 5.3.
3. Identify the best particle in this iteration as current best.
4. Update global best:

   if *current best better than global best* **then**
   global best $\leftarrow$ current best

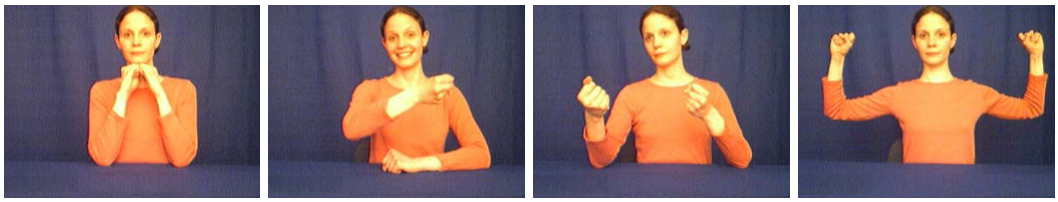
   else
   Increase inertia counter and decrease inertia weight.

   end

(a) Pose 1 (b) Pose 2 (c) Pose 3 (d) Pose 4



(e) Pose 5 (f) Pose 6 (g) Pose 7 (h) Pose 8



(i) Pose 9 (j) Pose 10 (k) Pose 11 (l) Pose 12

Figure 5.8: 12 different poses used in the experimental evaluation.

5.4 Experiments

In this section we present the results of running PSO-based pose estimation on real images. We acquired a set of 6 images (6 different viewpoints) for 12 different body poses for this purpose. The front view for each of the 12 poses is shown in Figure 5.8.

5.4.1 Cost Function Experiments

The cost function defined in Equation 5.8 can be interpreted in different ways, depending on the choice of weights ω_i. The simplest choice is to set all ω_i to be equal. However, when certain views contain occlusions, it would be intuitively useful to give more weight to views which are not occluded or less occluded and can guide the optimisation better. In the test set, the top view contains the lowest number of occlusions and could be given a greater emphasis to disambiguate other views. Another choice is to change the importance of the views dynamically in each iteration, by checking the quality of the overlap in every view

Configuration	Overlap Mean	Overlap Std. deviation
Configuration 1	0.887156	0.050874
Configuration 2	0.885688	0.052481
Configuration 3	0.887059	0.052004
Configuration 4	0.887792	0.051282

Table 5.3: Different weight configurations of the cost function and the corresponding pose estimation results calculated over 10 repetitions of the hierarchical pose estimation on 12 different pose examples.

and assigning more importance to the view where the optimisation is doing worst, thereby forcing it to explain it better.

We experimentally compared the following 4 different weight configurations:

1. All views equally important, $\omega_i = 1.0$;
2. More importance was given to the top view with the weight $\omega_{top} = 0.25$, while the other views had weights set to $\omega_i = 0.15$
3. One higher weight, $\omega_{worst} = 0.25$, was dynamically assigned to the view with the worst fit, while all the other views had weights set to $\omega_i = 0.15$;
4. Out of six views we manually chose three which contained fewer or no occlusions and gave them a higher weight $\omega_j = 0.25$, while the remaining three had weights set to $\omega_i = 0.0833$, $i \neq j$.

Each weight configuration was tested on 12 different poses by running hierarchical pose estimation with PSO 10 times per pose. The quality of the fit over all 120 runs was then evaluated for each configuration and the resulting mean and standard deviation statistics are shown in Table 5.3. As we do not have the ground truth information (such as optical motion capture data) available for the poses in the test set, and setting the ground truth manually is rather subjective, we instead evaluate the quality of the fit by counting the number of pixels in the final overlap between the model's silhouettes and the silhouettes extracted from the

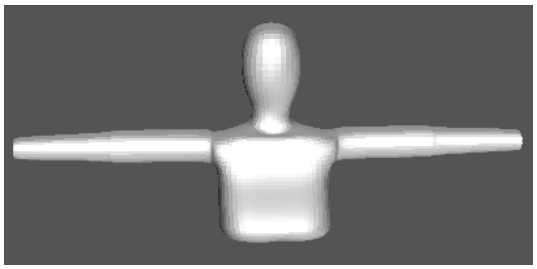
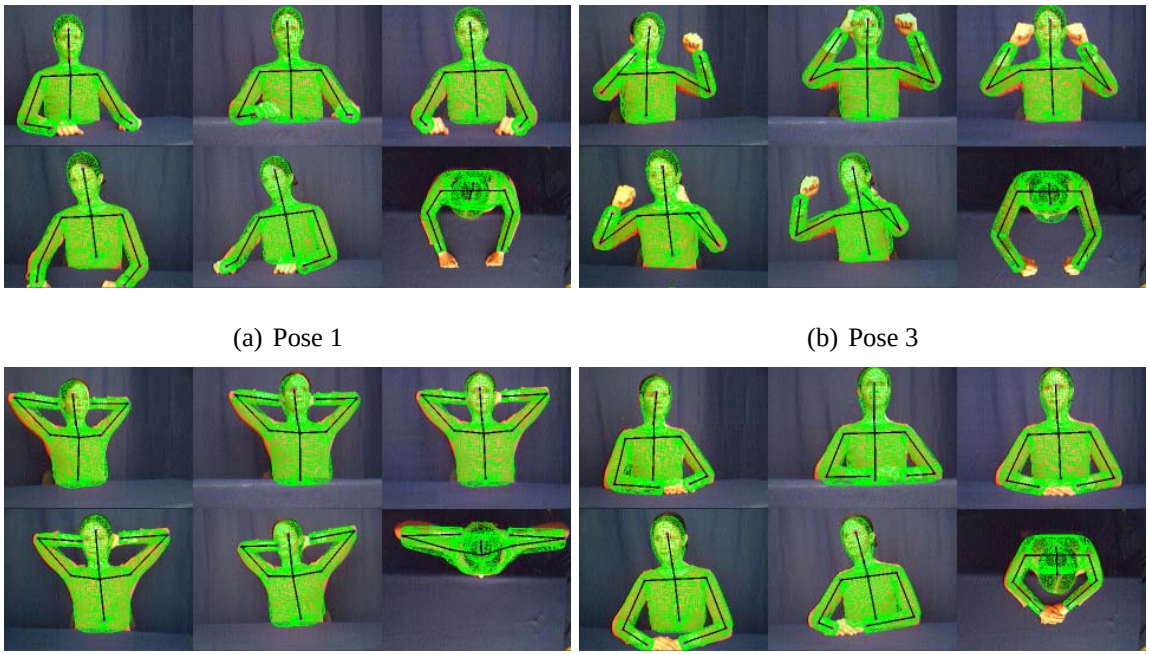


Figure 5.9: Default model pose. All optimisation experiments were initialised randomly around the default pose estimate.

original images. The bigger the overlap, the better the agreement between the true pose and that of the model.

Each pose estimate was evaluated by comparing the size of the model-silhouette overlap with the size of the original silhouette, averaged over all 6 views. The entries in Table 5.3 show what percentage of the original silhouettes the overlap achieved - the higher the percentage, the better the original silhouettes were explained by the estimated pose. An average silhouette size computed over all 12 poses used in the experiment is approximately 77588 pixels. This shows that, on average, the overlap size using the configuration 4 was approximately 49.3 pixels larger than the overlap size using the configuration 1.

Experimental results show that configuration 4, where more important views were chosen manually, performed best, followed by the configuration 1, where all views were given equal importance. Although the top camera does contain more information in poses with front-view occlusions, giving solely the top view a higher weight did not work as well as simply allowing all views to have the same influence. As we do not have a way of estimating the importance of the views automatically at this point, using configuration 4 would require labelling views manually in advance of every pose estimation. Our goal is a fully automatic pose estimation method and we have therefore decided to use the second best configuration instead and treat all views as equally important.



(a) Pose 1 (b) Pose 3

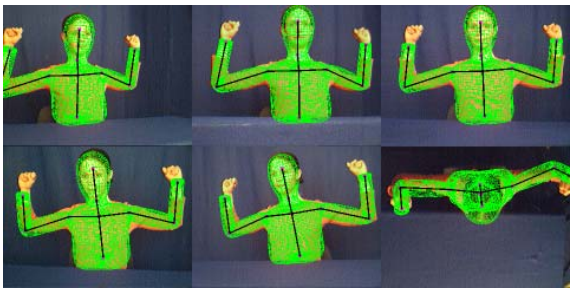
(c) Pose 5 (d) Pose 7

Figure 5.10: Successful results of the pose estimation with PSO. For every pose example, the model is shown overlaid on top of the original image in the corresponding six views. The views in this figure are sufficiently discriminative to allow for a good interpretation of the pose.

5.4.2 Pose Estimation Results

We now show the results of running the PSO algorithm using the cost function with configuration 1 on a test set shown in Figure 5.8. Due to the complexity of the evaluation function, we limited the swarm size to only 10 particles and set the starting inertia value to $w = 1.2$. The swarm was initialised around the default pose shown in Figure 5.9, which made an assumption about the approximate root position, as the person in the image was sitting on a chair, but made no assumptions about the body pose itself. The swarm was scheduled to move once per each inertia value, which changed if a better solution was found. In that case, the corresponding inertia value was kept the same until a swarm move without improvement on the global optimum forced it to decrease again.

Results in Figures 5.10 and 5.11 are the examples of poses where the estimate matches the real pose rather accurately as the silhouette constraints were informative enough and did not contain significant occlusions. Figure 5.12 shows the results of running the pose



(a) Pose 12

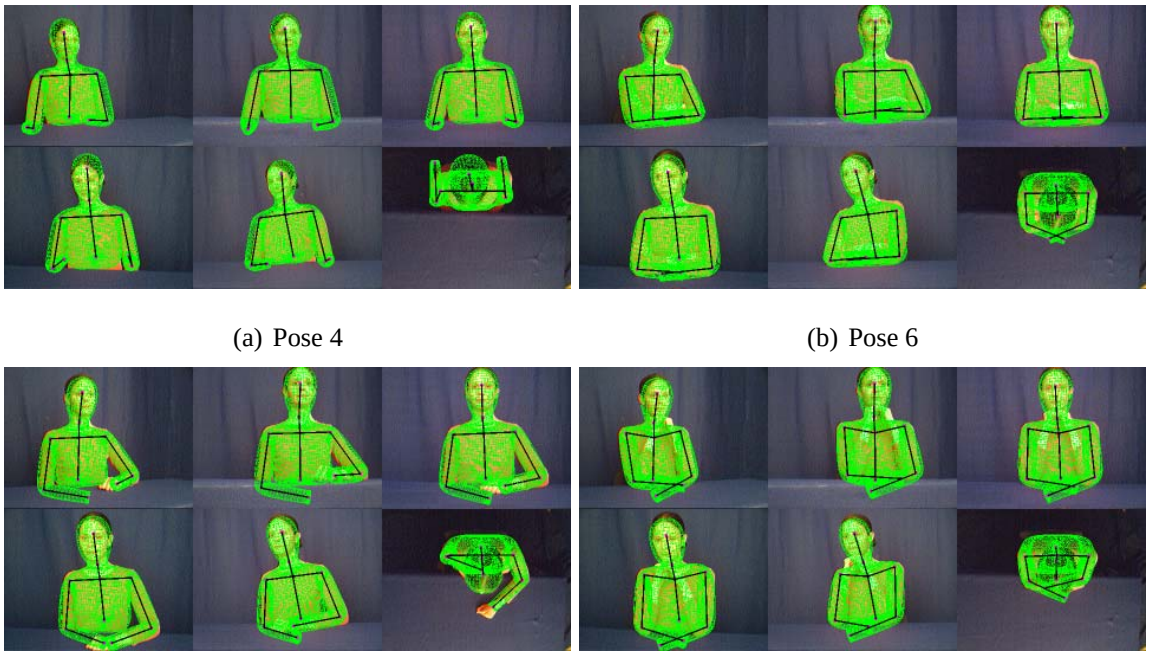
Figure 5.11: Another example of a successful result (continuation of Figure 5.10).

estimation on poses with missing or folded lower arms (invisible in the silhouette), a realistic scenario in a videoconferencing application. Although all views were given equal importance, the optimisation often managed to fold the arms in front or under the body to achieve a better overlap in the top view (see the lower arms in top camera view for pose 4, 6 and 9). This makes sense as by the time the optimisation reached the lower-arm step (steps 6 and 7 in Table 5.2), the top view was the only one with bits of silhouette left unexplained, a consequence of the rather approximate generic model dimensions.

Figure 5.13 shows the pose examples where the optimisation failed to recover the correct pose. In pose 2, the not-so-ideal shape and size of the model made the correct pose less obvious and the upper arm estimate almost completely occluded the lower arm in two of the views. In pose 10, the right arm was occluded in most of the views, while the lower right arm estimate also managed to lock onto the left hand to increase the overlap count. In pose 11, the left lower arm was occluded by the body in three views and by the upper arm in two views. The resulting pose explains the top camera very well and provides a minimally visible left lower arm in the rest of the views. The corresponding pose estimate overlaps with the silhouettes are also shown in Figure 5.13 to illustrate the ambiguity in silhouette constraints.

5.4.3 Pose Estimation for Different People

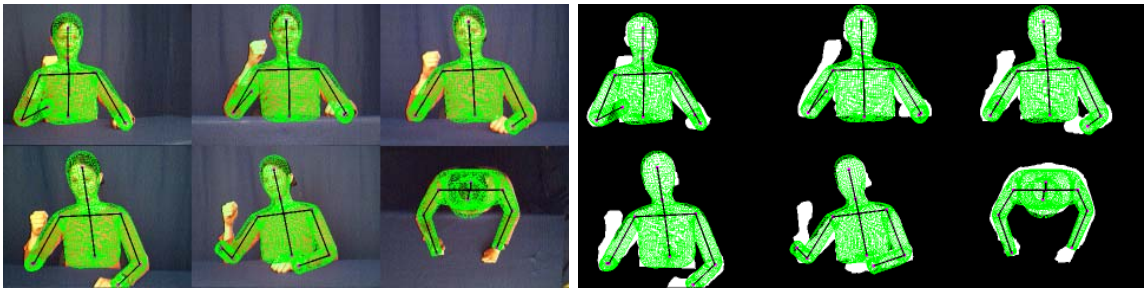
The pose estimation can be performed on different people by adjusting the model to better represent a particular person. Figure 5.6 shows the variation in the generic body model for



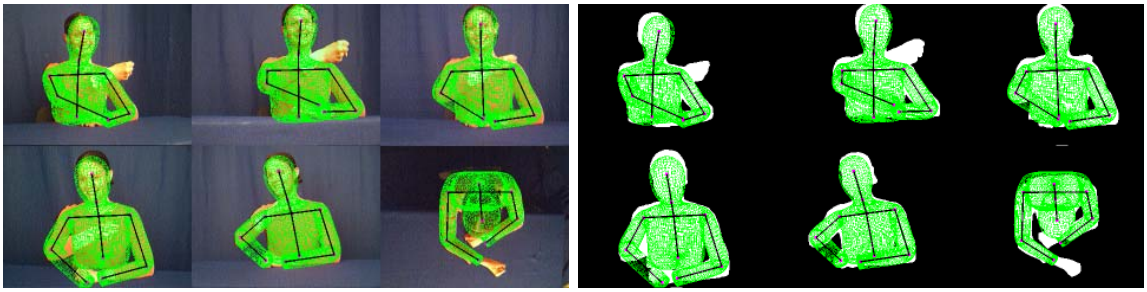
(a) Pose 4 (b) Pose 6

(c) Pose 8 (d) Pose 9

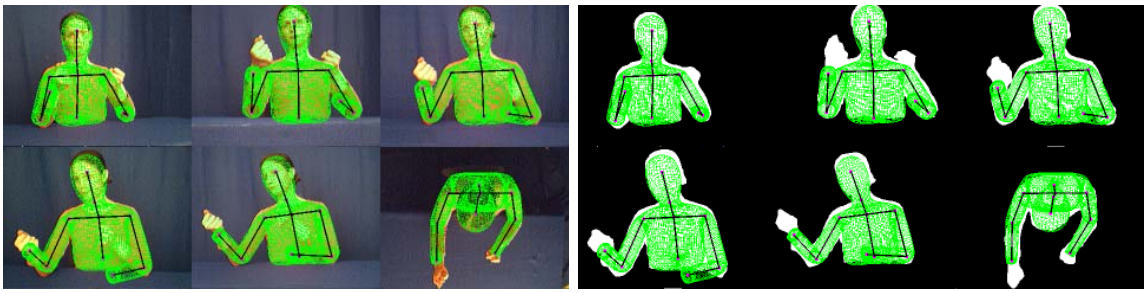
Figure 5.12: Results on deliberately occluded poses. PSO attempted to explain the lower arms by folding them onto or under the body. No default pose was enforced in these cases, the optimisation was left to explain the pose the best it could.



(a) Pose 2



(b) Pose 10



(c) Pose 11

Figure 5.13: Examples where PSO failed to recover the correct pose. These failures seem mainly the result of a lack of constraints as limbs are occluded in sufficiently many views to make the interpretation of the pose from the silhouettes very ambiguous. The overlap of the model and the silhouette is also shown to better illustrate where the pose optimisation has gone wrong.

Figure 5.14: Different fronto-parallel poses used to adjust model dimensions for a particular test subject.

three different test subjects. The limb lengths as well as the mesh dimensions were adjusted for every test subject. We adjust the model dimensions manually, once for each test subject.

A fronto-parallel body pose is used as a canonical body pose to aid the model customisation process. Ideally, the person in a canonical pose should be in a fronto-parallel pose with arms extended to both sides, however, due to the limitations of our setup, we resorted to using available frontoparallel poses such as those shown in Figure 5.14.

Once the model has been adjusted, the pose estimation continues as described before. Figure 5.15 shows results of pose estimation on different test subjects from the one in the original experiment.

5.4.4 Comparison Experiments

We performed a gradient descent (GD) optimisation with a constant step size parameter [59] to demonstrate that the pose estimation problem was complex enough to warrant a global optimisation approach such as PSO. The optimisation was initialised randomly around the canonical pose estimate shown in Figure 5.9. Despite some of the runs producing a good pose estimate, the resulting pose estimates were sufficiently inconsistent to indicate that the cost function landscape was too complex for a simple downhill optimisation such as gradient descent (see Figure 5.16).

We also compared the performance of PSO with the downhill-simplex simulated an-

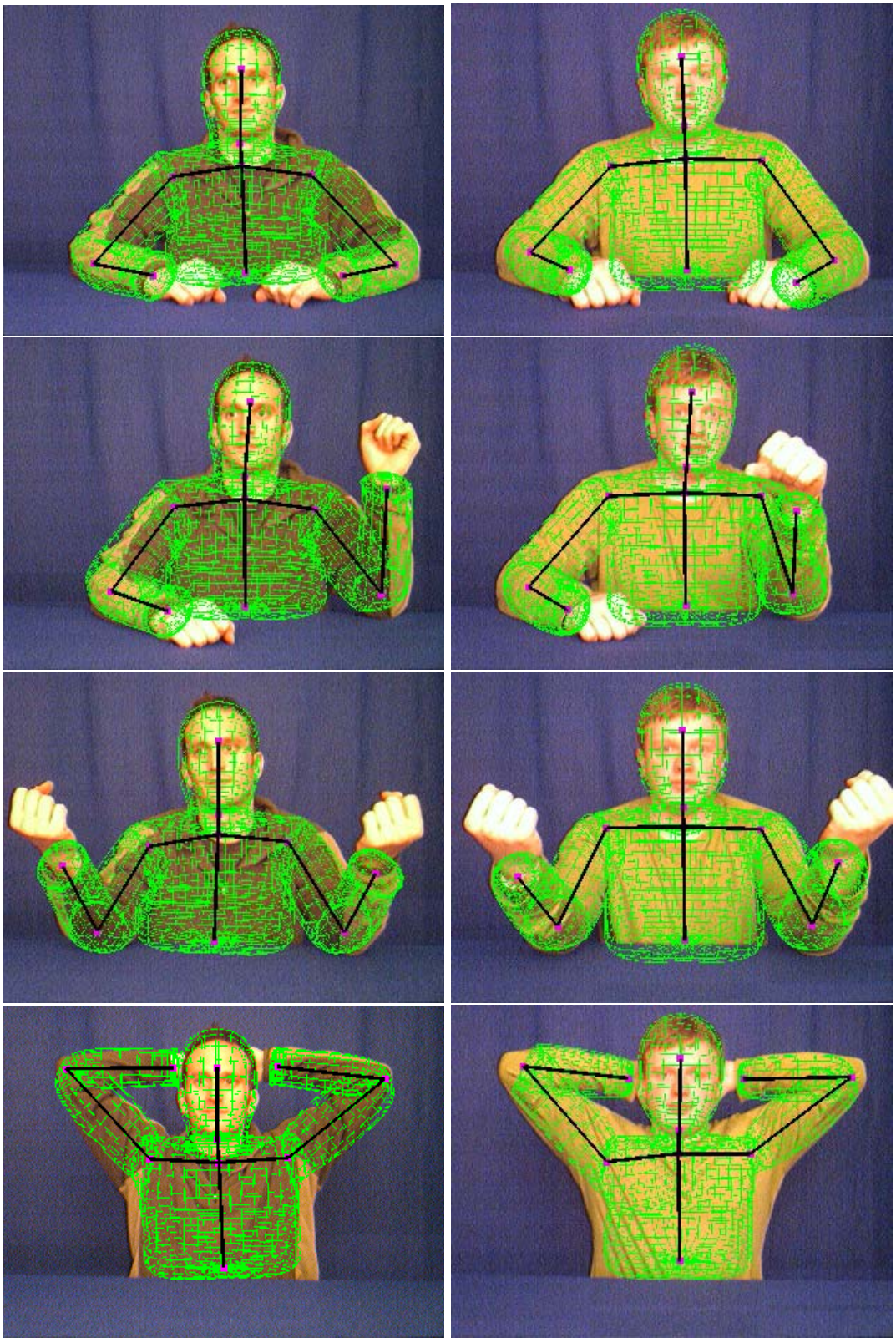


Figure 5.15: Pose can be estimated for different people, the only requirement is that the model dimensions are adjusted to those of the imaged person.

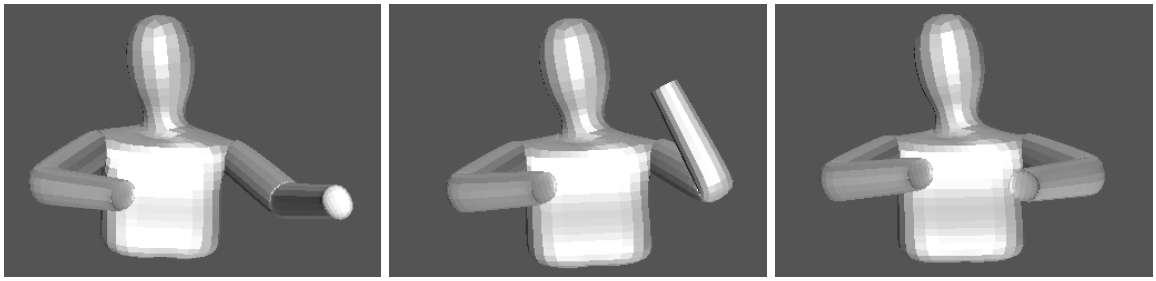


Figure 5.16: Three different pose estimates for pose number 3, generated by three sequential randomly initialised runs of gradient descent.

nealing [122]. Simulated annealing (SA) with its annealing schedule and PSO with the decreasing inertia parameter intuitively behave in a similar way when exploring the search space. Both start with a high degree of randomness and global search and gradually slow down to explore a smaller area of the search space. Despite the apparent similarity in the general behaviour of the two methods, this is not true for the parameters which they use. In order to enable a sensible comparison of the two methods, care must be taken when setting the parameter values.

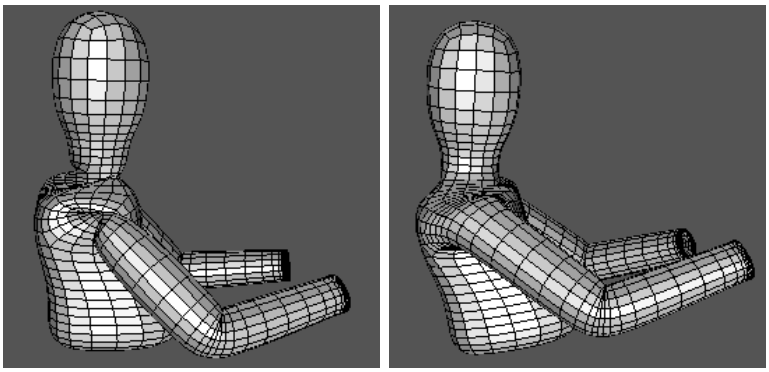
We configured the PSO parameter values through experimental trials. The starting inertia value had to be high enough to enable global exploration of space. In our experiments it was set to $w = 1.2$. The exponential function in Equation 5.10, used to model the inertia change over time, was sampled with a sampling step equal to 0.05. The optimisation was terminated when the inertia value fell below 0.1. The swarm consisted of 10 particles and was allowed one move per inertia value, unless it found a better solution, in which case it was allowed to stay at the current inertia value for another move.

For the simulated annealing, the starting and stopping conditions were set according to the guidelines in [131]. The starting temperature was set to $T_0 = 10.0$ which meant that, at the start, every move was accepted and the global exploration was encouraged. The annealing schedule was chosen to resemble the inertia change function in PSO as closely as possible. The temperature decrease schedule was set to $T_{new} = 0.95 * T_{old}$. The algorithm had 10 iterations available at every temperature level, corresponding to the 10 particles in the PSO. Like PSO, if a better solution was found at a particular temperature level, the algo-

rithm was allowed to remain at that temperature for another 10 iterations. The optimisation was terminated either when the temperature fell below 10^{-4} or when the improvement step fell below the tolerance threshold of 10^{-5}.

We attempted a comparison of both PSO and SA on a constrained search space. For PSO this meant that whenever a particle crossed the boundary of the search space in a particular dimension, the position of the particle in that dimension was set back to the boundary value and the corresponding velocity vector reversed. In SA, the cost function was designed to return a high error value (the pose estimation was formulated as a minimisation) to discourage the optimisation from searching in that area. Experiments showed that with only 10 iterations per energy level this strategy did not allow the optimisation to make a sensible recovery once it crossed the border of the allocated search space and consequently that optimisation step failed. Constraining the search to the allocated search space was much more easily achieved using PSO as the direction of the particles was specified by their velocity vector and could be easily reversed when the boundary was reached. To allow for a fairer comparison, we present the results of the unconstrained SA instead, with the drawbacks illustrated in Figure 5.17.

Figure 5.18 shows the error-bar comparison of the three different optimisation methods. The error bars indicate the mean and standard deviation of the overlap, calculated for each individual pose. Although some SA and GD estimates did exceed PSO (see poses 11 and 9), the overall performance of PSO was better and more consistent. In terms of efficiency, PSO came top, followed by GD and SA. On the set of 12 poses, PSO took on average 136.11 ± 5.7 seconds to complete and required 414.92 ± 16.7 cost function iterations, GD took on average 323.31 ± 66.4 seconds to complete and required 9562.5 ± 1964.5 cost function evaluations, while SA took on average 503.44 ± 31.06 seconds to complete and required 15035.58 ± 864.1 cost function evaluations.



(a) (b)

Figure 5.17: Two different results of the unconstrained SA. The twist of the skin indicates that the rotation of the shoulder in the result (a) is overestimated by at least one period. In another run, (b), the same shoulder rotation was explained by the identical rotation in its original period. While unconstrained optimisation still produces a plausible pose estimate, even if outside the primary search space, it takes longer to converge, which is another significant drawback.

5.5 Conclusion

In this chapter we have presented a particle swarm optimisation (PSO) method for articulated human body pose estimation. The pose has been systematically estimated on real images and the quality of pose estimates compared with the equivalent approach using gradient descent and simulated annealing.

Results demonstrate that the PSO-based method is more consistent and efficient than the other two methods. It performs well on unambiguous views as well as views where limbs are not visible at all. Although the explanation in those cases is purely cost function driven, it is actually entirely acceptable for the scope of this work, *i.e.*, the novel-view synthesis, as the limbs folded onto the body to conceal their silhouette do not significantly mislead the disparity values generated later in Chapter 8.

The PSO technique is slightly less successful on views where the silhouette constraints are ambiguous, which is to be expected, as the silhouettes and realistic parameter intervals were the only constraints used for the optimisation. The results could be further improved by including the probabilistic prior knowledge about the feasibility of a particular pose,

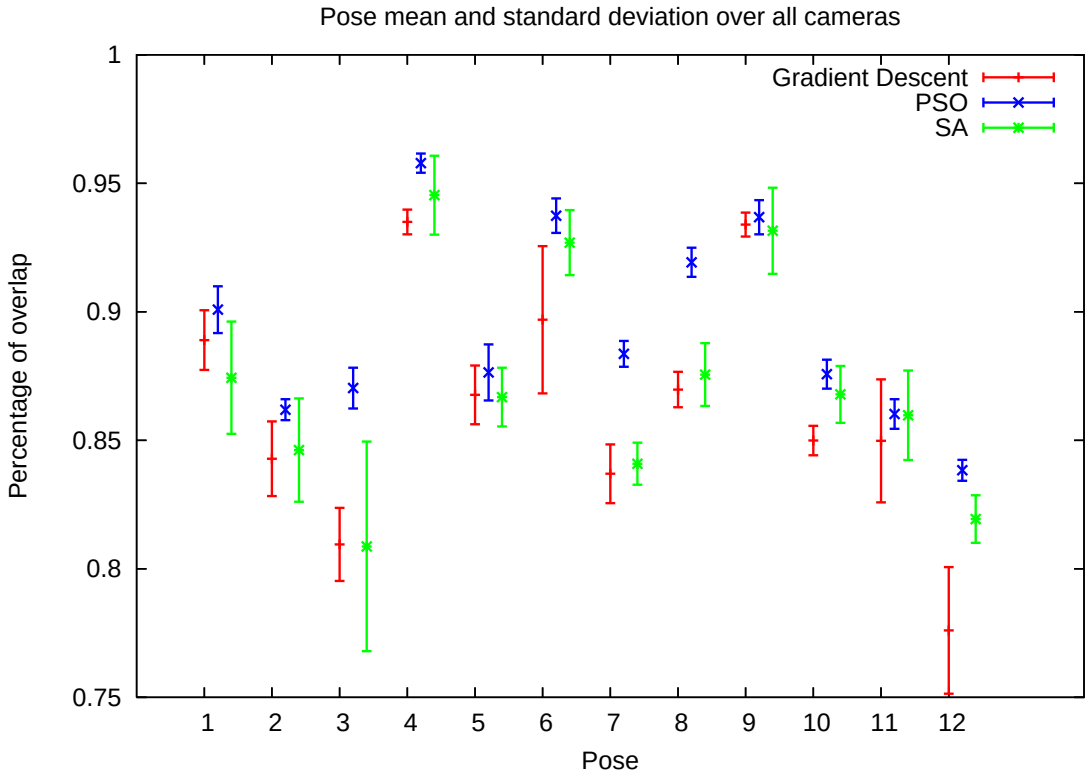


Figure 5.18: The results of the pose estimation comparison using GD, PSO, and SA. Results are shown in triplets for every pose, starting with GD, PSO and SA result for Pose 1. The GD error bar is always positioned above the corresponding pose number and the PSO and SA to the right of it. Across the entire test set, the pose estimates produced by PSO are more consistent than those of SA, while GD produces the worst performance. The variability in the SA results for pose 3 are a consequence of the lack of constraints as the search several times failed to correctly identify the root position and then produced significantly differing explanations of the remaining parts of the model. The SA performance on other poses was better, although the amount of variability in the results was larger than that of PSO.

however, we must emphasise that the solutions found in the ambiguous cases were in fact not at all improbable, but rather sensible explanations of the silhouette constraint, even if not quite correct. In view of this, we believe that disambiguating the silhouette constraints by adding more views as well as including the texture information to guide the search would further improve the results.

The pose estimates obtained in this chapter represent an input to our fitting algorithm, which fits a generic model to an unstructured cloud of points. We discuss this algorithm in the next chapter.

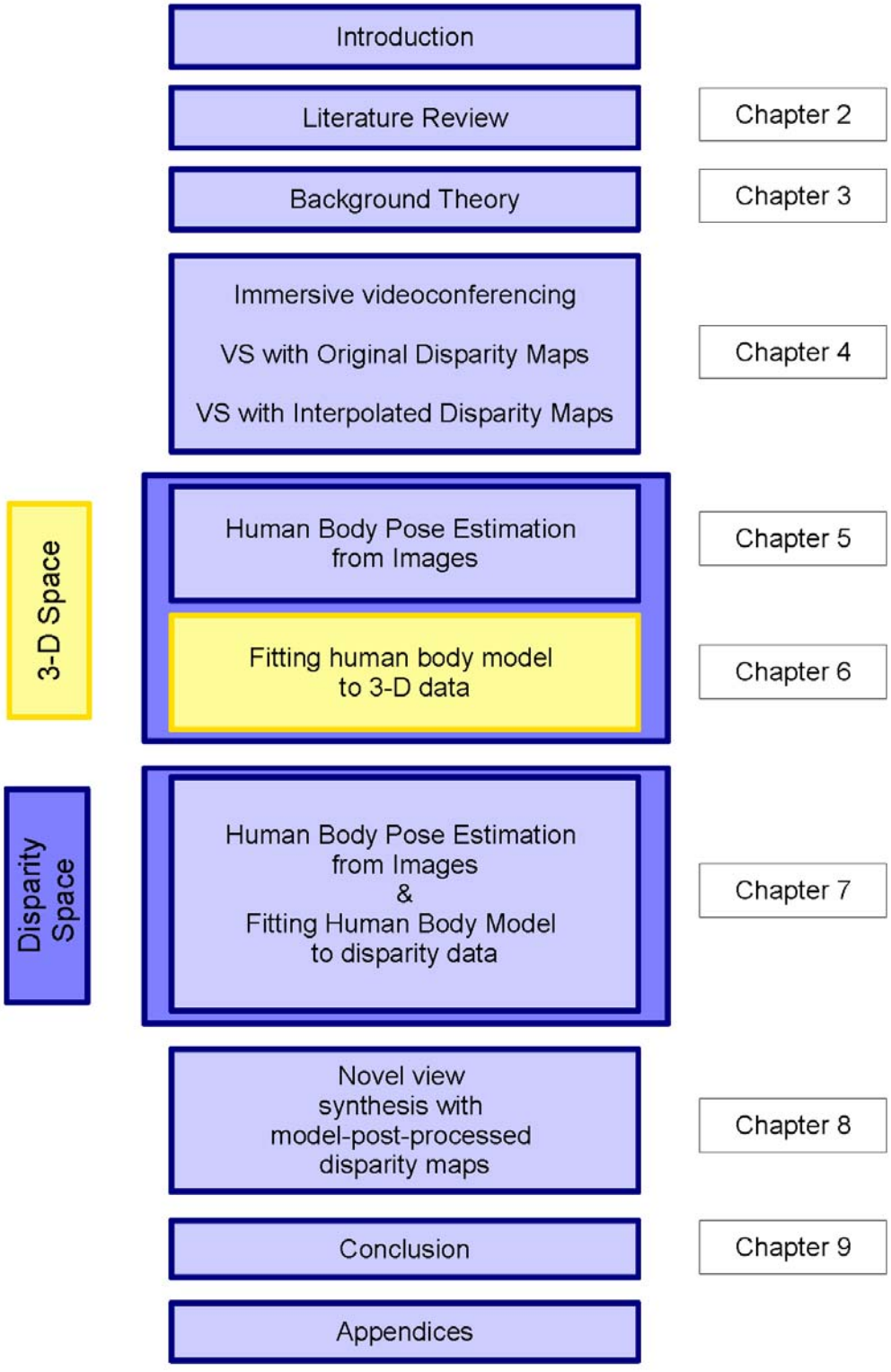


Figure 5.19: The thesis structure diagram.

Chapter 6

Model-Based Disparity Map Completion

6.1 Introduction

In the previous chapter we presented the PSO-based method for articulated body pose estimation, using a generic 3-D model of the human body. We now proceed with the assumption that the model described in Section 5.2 is readily available and that its pose has been estimated using the algorithms given in Section 5.3.5.

In this chapter, we address the model-based disparity map completion. We fit the generic model in the correct body pose to the reconstructed disparity data and use the result of the fit to generate the complete disparity map for novel view synthesis.

By fitting the model to the disparity data, we achieve the following:

- The generic model after the fit more closely resembles the actual disparity data.
- The generic model provides the information missing in the original disparity map.

The structure of this chapter is as follows: we begin by presenting the method for fitting a generic subdivision surface 3-D model to an unstructured and incomplete cloud of 3-D points in Section 6.2. To demonstrate the potential of the fitting method, we then show results on an incomplete but accurate range data set of a human head containing a high level

of detail in Section 6.3, followed by results on noisy and incomplete 3-D data obtained by 3-D reconstruction from disparity data in Section 6.4. We conclude with Section 6.5.

6.2 Fitting a 3-D Model to Unstructured and Incomplete Data

In this section, we present our method for fitting a generic 3-D subdivision surface model to an unstructured, noisy and incomplete cloud of 3-D points. Our method is based on the quasi-interpolation approach of Litke *et al.* [96], originally developed for scientific visualisation and animation of accurate and complete data.

In the original method [96], the authors adaptively subdivide and deform the model mesh to better approximate the data. In every iteration, only the mesh faces which do not pass a pre-defined approximation criterion are marked for further subdivision and subsequently modified. While this approach is sensible when fitting to accurate and complete data with high level of detail, it is not suitable for working with noisy and patchy data for reasons explained in the next section.

6.2.1 Unsuitability of the Original Method for Our Problem

Missing data. The original method assumes that every mesh vertex has a corresponding data point which can be used to evaluate the approximation criterion on every mesh face. This is not true for our patchy disparity data. In our scenario, mesh faces exist for which no data correspondences are available. For such faces, the approximation criterion cannot be applied, although these might be faces which, if the data was complete, would indeed require further subdivision.

Noisy data. Disparity data originates from a stereo correspondence search and contains noise in the form of false matches which survived the consistency check. It is therefore difficult to define an appropriate approximation criterion which would allow the model to be adaptively fit to a high level of detail while approximating only the correct data and not

the noise.

Patchy data. The fitting method modifies the model mesh locally. With patchy data this means that parts of the mesh with available data correspondences would deform to better describe the data, while interspersed parts of the mesh without correspondences would remain fixed on their initial level of subdivision, creating unwanted artefacts on the limit surface.

Mesh connectivity. Adaptive subdivision interrupts the mesh connectivity on the boundaries between different subdivision levels. This results in gaps which are typically closed by inserting additional triangular or quadrilateral faces. These additional faces create a locally denser mesh which produces a different limit surface shape to the one defined by the original base mesh. In a complete data set, such mesh modifications typically happen in the transition areas where the level of detail gradually changes from lower to higher. In a patchy data set, these modifications are scattered over the mesh in a manner which is not consistent with the level of detail transition areas but rather consistent with regions of data availability. The resulting changes to the limit surface are therefore undesirable.

6.2.2 Modifications of the Original Method for Our Problem

In the previous section we listed a number of reasons which rendered the original method unsuitable for use on our patchy disparity data. However, its efficient iterative nature which allows the fit to be performed to a desired level of detail provides an interesting aspect which, with modifications, can be exploited in favour of our noisy data set.

We modified the original method in several ways. We replaced the adaptive subdivision with a uniform subdivision method. This meant that the base mesh became uniformly denser and closer in shape to the limit surface after every iteration.

The use of uniform subdivision removed the need for the approximation criterion and allowed us to modify the shape of the limit surface without introducing unwanted artefacts due to local changes in mesh connectivity. Consequently, the limit surface could be used to

provide an *a priori* shape information in the regions with missing data, *e.g.*, the back of the model, as well as model the data in more detail where it was available.

While an adaptively subdivided surface approximates the detail in the data more efficiently than the uniformly subdivided surface, the latter better approximates noisy data by increasing the modelled level of detail uniformly across the surface. This allowed us to set a pre-defined level of iterations and avoid explicitly modelling the noise in the data as our aim was to make the model resemble the shape of the data without modelling it in minute detail.

Our fitting algorithm, presented in the next section, has been tested on stereo data derived from a stereo pair of still images. Stereo data can also be generated from a video sequence, frame by frame. We wanted to allow for incremental fit of the model to such data, where the shape of the model is updated through time as new data becomes available. This requires updating the position of the vertices in the already deformed base mesh and propagating the change down the hierarchy of subdivision surfaces and is more readily achievable with a uniformly rather than adaptively subdivided surface.

Last but not least, we wanted a fitting method that was simple and easy to reproduce. In comparison with uniform subdivision, adaptive subdivision requires paying attention to a number of ways in which the mesh connectivity must change to close the gaps between the faces on different levels of subdivision. Deciding where to subdivide adaptively requires a much more thorough assessment of the quality-of-fit which makes sense when working with detailed and clean data but is not of the utmost importance when dealing with noisy and patchy data such as ours.

6.2.3 The Model Fitting Algorithm

The fitting algorithm uses the *quasi-interpolation* method, described in detail in Section 3.7. The input is the base mesh B_0 of a generic Catmull-Clark subdivision model and an unstructured set of M data points $\{P_j\},\ j = 1 \ldots M$ (see Figure 6.1). We assume that the correspondences between the data and the mesh surface of the model can be established during the fitting process and are not known in advance.

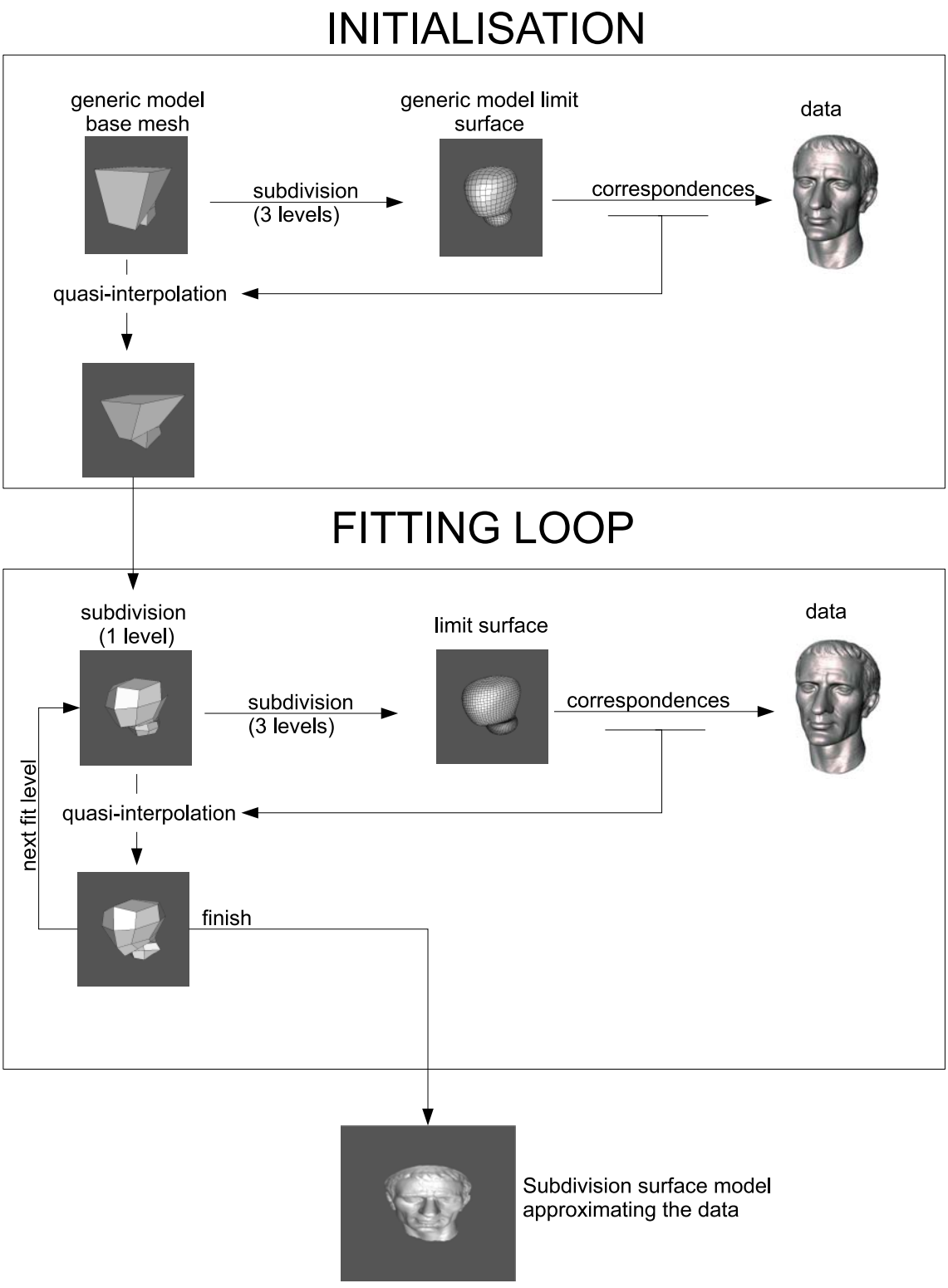


Figure 6.1: The illustration of the model fitting algorithm. The data is treated as an unstructured cloud of points (see Figure 6.2(c) for an example).

Let us first introduce the necessary terminology. The base mesh B_0 is the coarse mesh which, when subdivided infinitely many times, takes on the shape of the smooth subdivision surface, the *limit surface* S^∞:

$$S^\infty = \lim_{n\to\infty} S^n B_0, \tag{6.1}$$

where S is the subdivision operator and S^n its n-th application. Every vertex of the base mesh $b_k^0 \in B_0$ has its limit position on the limit surface, denoted by $v_k^0 = \lim_{n\to\infty} S^n b_k^0$, where $k = 1,\ldots,K$ and K is the number of base mesh vertices.

In practice, it suffices to subdivide the base mesh a finite number of times to obtain an acceptable approximation to the limit surface. We will denote this approximation with $\tilde{S^\infty}$:

$$\tilde{S^\infty} = S^N B_0, \quad \textit{where} \quad N < \infty, \tag{6.2}$$

In all our fitting experiments we set $N = 3$ because 3 levels of subdivision approximate the desired limit surface sufficiently well for our purposes. When referring to our fitting algorithm, the limit surface $\tilde{S^\infty}_i$ denotes the base mesh B_i subdivided 3 times and $\tilde{v}_k^i$ the limit position of the k-th base-mesh vertex $b_k^i \in B_i$ on the approximating limit surface, $\tilde{v}_k^i \in \tilde{S^\infty}_i$. The number i denotes the number of iterations of the algorithm.

The fitting process consists of two stages: the *initialisation stage*, where the base mesh is deformed while its connectivity remains unchanged, and the *fitting loop*, where the base mesh is repeatedly subdivided and deformed until satisfactory approximation is achieved (see Algorithm 6.1 and Figure 6.1).

Algorithm 6.1 has been designed to deal with incomplete data. The vertices of the base mesh for which no correspondences with data points can be found remain unchanged during the quasi-interpolation step (Initialisation step 3 and Loop step 4). In this way, the parts of the model corresponding to missing data regions converge towards the limit surface defined by the generic model and thus complete the missing regions with the *a priori* knowledge.

Unlike in computer graphics, fitting subdivision models to unstructured data is not yet widely used in computer vision. We identified an approach by Ilic [80] who fits triangular

Algorithm 6.1: Model Fitting Algorithm

Initialisation $i = 0$

1. Generate the limit surface of the initial base mesh $\tilde{S^{\infty}}_0 = S^3 B_0$.
2. Find correspondences between $\tilde{v}_k^0 \in \tilde{S^{\infty}}_0$ and the data points $\{P_j\}$.
3. Apply the quasi-interpolation operator Q to all vertices $b_k^0 \in B_0$ of the base mesh for which the correspondences could be found.

Loop: $i = i+1$, *repeat until satisfactory approximation is achieved.*

1. Uniformly subdivide the base mesh: $B_i = S^1 B_{i-1}$.
2. Generate the limit surface of the new base mesh $\tilde{S^{\infty}}_i = S^3 B_i$.
3. Find correspondences between $\tilde{v}_k^i \in \tilde{S^{\infty}}_i$ and the data points $\{P_j\}$.
4. Apply the quasi-interpolation operator Q to all vertices $b_k^i \in B_i$ of the base mesh for which the correspondences could be found.

Output: Subdivision model deformed to fit the data.

subdivision surfaces to unstructured range and stereo data of a human face. His model fit is formulated as a vertex-based least-squares optimisation method with a regularisation factor constraining excessive model surface deformation.

Compared to [80], our method is not an optimisation but an iterative data approximation process which can be stopped as soon as the fit is accurate enough for our purposes. We start the fit with a coarse base mesh and gradually increase its density and add more detail. In this way we avoid the need for a regularisation factor to ensure smooth mesh deformation, as this is naturally achieved through the multi-resolution fitting process.

(a) front view (b) side view (c) point cloud

Figure 6.2: (a,b) Julius Caesar range data set. The data on the back of the head is missing. (c) The data set is originally provided as a mesh model, however, for the purpose of our fitting experiment we only use the mesh vertices as a raw data point cloud containing 19572 points.

6.3 Fitting to Range Data

Although we primarily use the fitting method in Algorithm 6.1 for noisy stereo data, the same method can also be used on more accurate and detailed data with missing regions that need completing. To show this, we first demonstrate the method on a range data set of the Julius Caesar head mask, provided courtesy of INRIA by the AIM@SHAPE Shape Repository [9], illustrated in Figure 6.2.

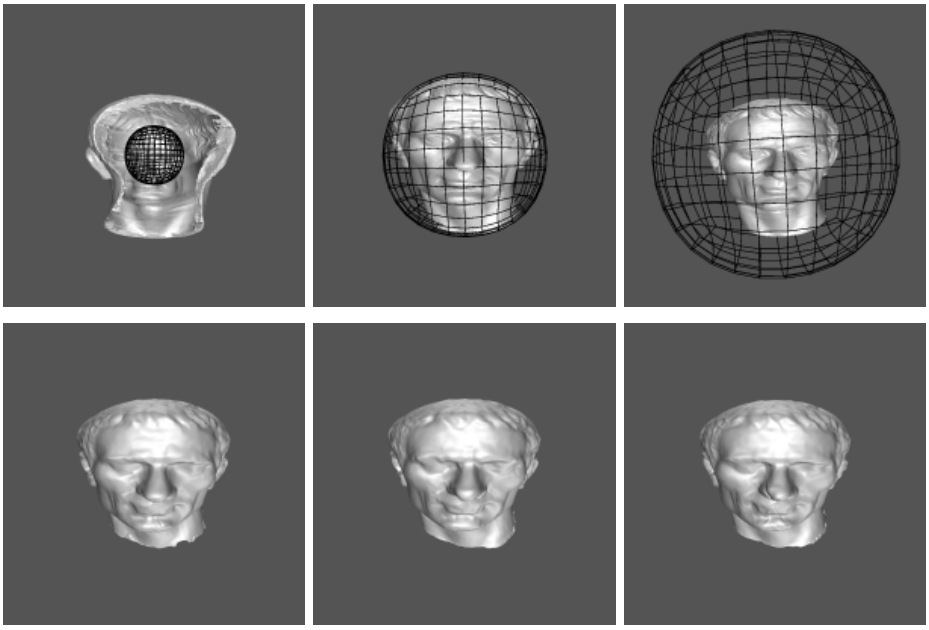
The reason for choosing this particular data set is two-fold. Firstly, the data on the back of the head is missing, as shown in Figure 6.2(b), which allows us to test how well we can complete the missing region with a generic shape. Secondly, the data is accurate enough to allow a quality-of-fit assessment by measuring the distance of the surface from the data as the fit evolves.

We performed several experiments analysing the behaviour of the fitting method. In the first experiment, a simple sphere model was used as the generic shape. The purpose of the experiment was to establish how sensitive the method is to different initial configurations and how well it performs given a very simple, although topologically appropriate, generic shape. In the second experiment a more informative generic shape in the form of a simple head model was used to show how a better generic model changes the end result. Finally, in the last experiment, we used the most detailed generic model to show how a careful

placement of vertices in the base mesh facilitates faster convergence of the fitting process by guiding the mesh deformation in the early stages of the fit.

Sphere Model and Varying Initial Configurations

In this set of experiments we used a simple sphere model to test the sensitivity of the final result to the starting distance and relative position of the generic model with respect to the data. We used three different scenarios: the sphere was scaled to fit inside the data, as close as possible to the data, and around the data, as shown in the upper row of Figure 6.3.



(a) Inside (b) At data (c) Outside

Figure 6.3: Upper row: different starting sphere positions with respect to the data. Lower row: The final results for the three sphere positions after 5 iterations of the fitting algorithm. All 3 spheres almost replicate the data.

We iterated the main loop of the fitting algorithm five times for each of the three starting configurations. The final results are shown in the lower row of Figure 6.3. Visually, all 3 configurations practically replicate the data in 5 iterations.

Finding correspondences. The correspondences between the model and the data are established by searching for the closest data points along the vertex normals of the model

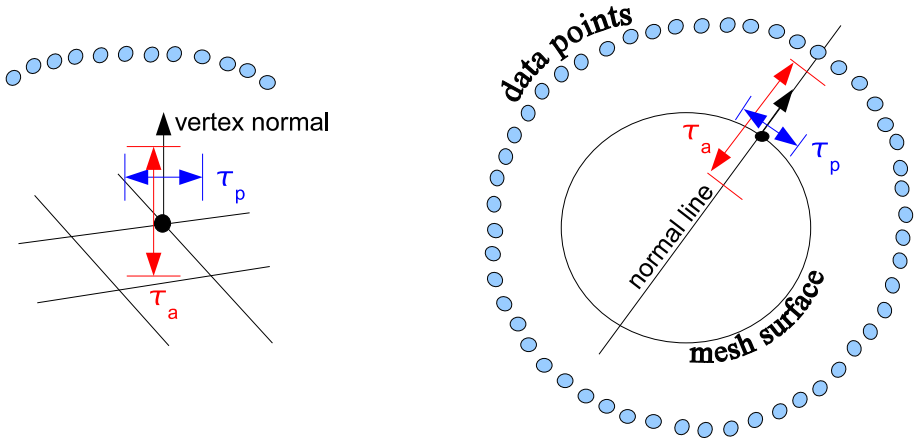


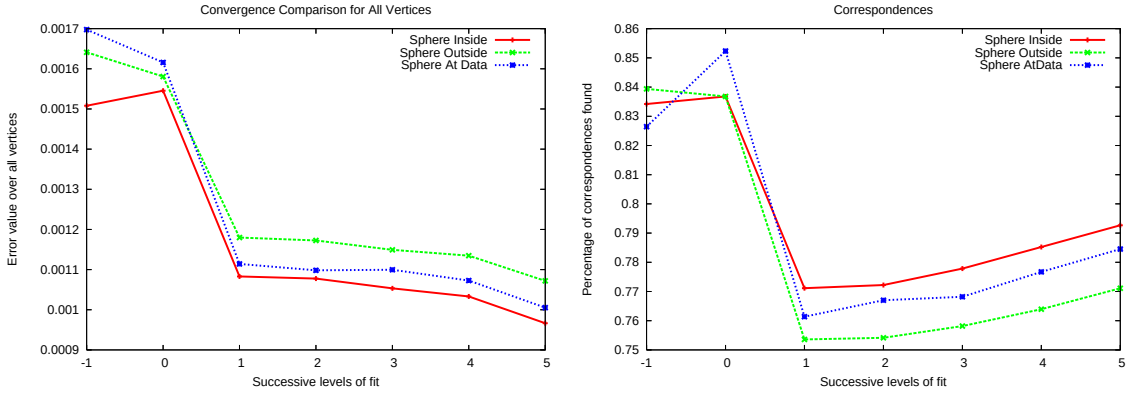
Figure 6.4: The thresholds used to find the corresponding pairs of mesh vertices and data points. Threshold τ_p prevents the cross-talk between neighbouring vertices and τ_a the erroneous matches when the normal line passes through more than one set of data points.

mesh. Two thresholds are used in the search, τ_p to control how far perpendicular to the normal, and τ_a how far along the normal the search extends for every vertex. Thresholds are necessary to prevent false matches and cross-matches between neighbouring vertices (see Figure 6.4). The first threshold, τ_p, is defined as the midpoint between two vertices in the mesh and its size changes with mesh density. The second threshold, τ_a, is set experimentally and must be much larger in the initialisation phase of the fit to successfully attract the model to the data. After the initialisation, τ_a is reduced and linearly decreases with every iteration while the model mesh density, *i.e.*, the number of mesh vertices and faces, increases, which explains the sudden change in the behaviour of the error function from iteration 0 to iteration 1 in Figures 6.5 and 6.7.

Quantitative assessment. For a quantitative assessment of the fit quality, the error function evaluated the RMS distance between the model surface and the data where correspondences were successfully established and weighted the result by the percentage of vertices for which the correspondences could not be found:

$$E = \frac{N_{nc}}{N} \sqrt{\frac{1}{N_c} \sum_{i=1}^{N_c} (v_i - p_{v_i})^2}, \tag{6.3}$$

where v_i is a limit surface vertex, p_{v_i} is the corresponding data point, N is the total number of vertices in the limit surface, N_c is the number of vertices for which correspondences were



(a) (b)

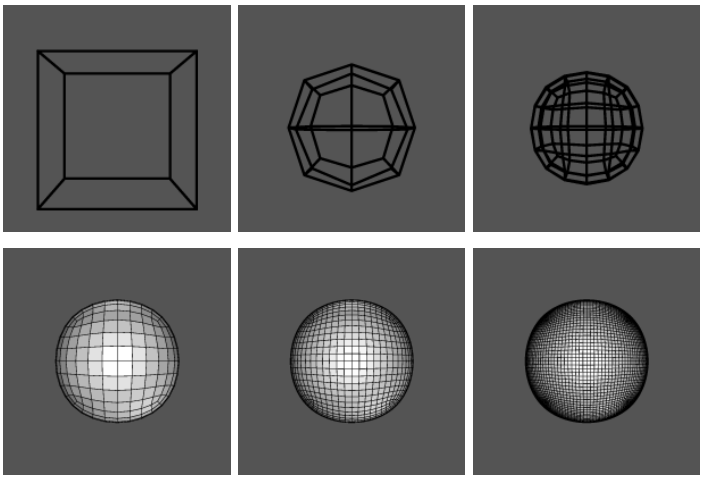
Figure 6.5: The behaviour of the fitting process for three different starting points. (a) Error function convergence. (b) Percentage of correspondences, N_c/N, found on every level of the fit.

established and $N_{nc} = N - N_c$.

The resulting error graph for the 3 different configurations is shown in Figure 6.5. As the data units were not known, the data points were normalised to $[0,1]$ to facilitate quantitative analysis. Results shown in the graph confirm that the quantitative difference in the final result between the 3 different configurations is indeed very small, less than 0.02%.

The iterations in Figure 6.5 are labelled from -1 to 5 to differentiate between the initialisation stage of the algorithm and the fitting loop (see Algorithm 6.1 and Figure 6.1). Iteration -1 illustrates the starting point of the fit and iteration 0 describes the model after the initialisation step. Iterations 1 to 5 repeat the main loop of the fitting algorithm.

As the generic shape is a subdivision model which can be used at different levels of mesh density, we performed another experiment investigating how the density of the starting base mesh influences the flow of the fit. We chose the sphere model scaled to fit the data from the previous experiment and modified it to obtain two additional levels of density as shown in Figure 6.6. For reasons of computational complexity and for fair comparison, we used 5 iterations of the algorithm on the lowest density model, 4 iterations on the medium one and 3 iterations on the highest density model. In this way all three models had the same base mesh density on the final level and could be compared more fairly.



(a) (b) (c)

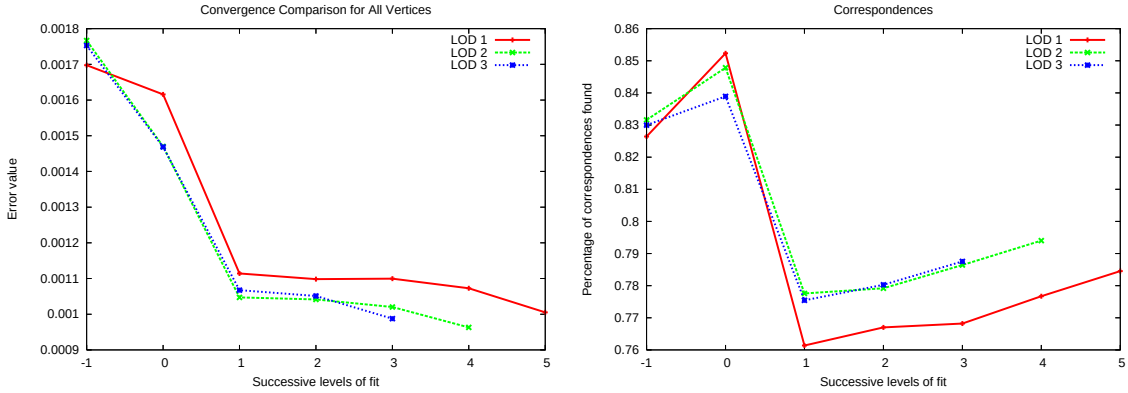
Figure 6.6: Initial sphere model with an increasingly dense mesh. (a) Lowest density base mesh and corresponding subdivision model. (b) Medium density base mesh and corresponding subdivision model. (c) Highest density and corresponding subdivision model.

Figure 6.7 illustrates the behaviour of each model. The error plot over all vertices shows that the model with a denser base mesh achieves the same quality of the fit in fewer iterations as it samples the data in more detail, establishes more correspondences and recovers the detail faster. All three models achieve comparable fit quality if allowed to iterate to the same level of base mesh complexity. This is a useful property if an approximation on several levels of detail is required. The two denser models have a very similar performance which also indicates that there is a limit to the useful mesh density.

Simple Head Model

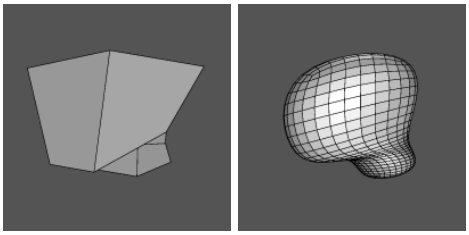
In this section, we show how we can use a more informative generic shape to complete large regions of missing data with our fitting algorithm. Our aim is to keep the model as simple as possible. In place of the sphere from the previous experiment, we fit a very basic head and neck model, shown in Figure 6.8, which contains more information about the actual shape of the data.

Results of the fit with the basic head model are shown in Figure 6.9. Although the



(a) (b)

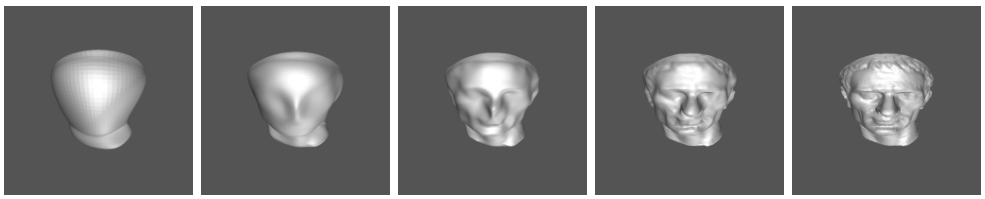
Figure 6.7: Behaviour of the fit for three different model mesh densities. (a) Error function convergence. (b) Percentage of correspondences N_c/N found on every level.



(a) (b)

Figure 6.8: Basic head model used in the second set of experiments. (a) The base mesh. (b) The corresponding limit surface.

shape of this model is much more informative than the sphere, the basic model turns out to be too simple to convincingly model the head and a more elaborate base mesh is necessary to better capture the head shape. In the next experiment we show how a slightly more complex base mesh achieves the desired result.

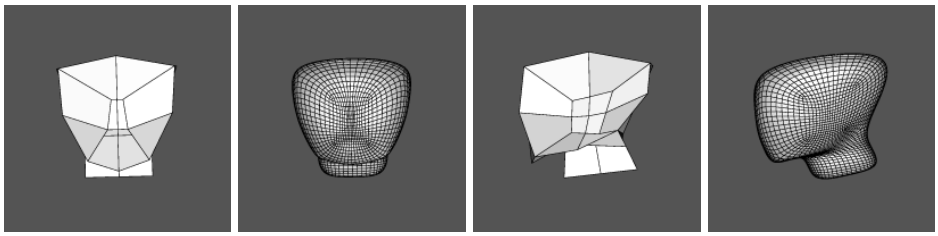


(a) I_1 (b) I_2 (c) I_3 (d) I_4 (e) I_5

Figure 6.9: The fitting stages with the basic head model (Figure 6.8). I_i denotes the iteration of the fitting algorithm.

Head Model with Strategic Control Point Placement

While performing experiments with the simple head model, we observed that it made a noticeable difference if the data points belonging to the prominent face features, such as nose and ears, were sampled first. We therefore constructed a new base mesh for the generic head and neck model, which contained control points exactly where the important face features were expected to be (see Figure 6.10). This proved to be a much more successful approach. When the new control points were added, the overall mesh density increased slightly as well. This allowed us to create a more convincing head shape which very realistically completed the missing data on the back of the head (see Figure 6.11).

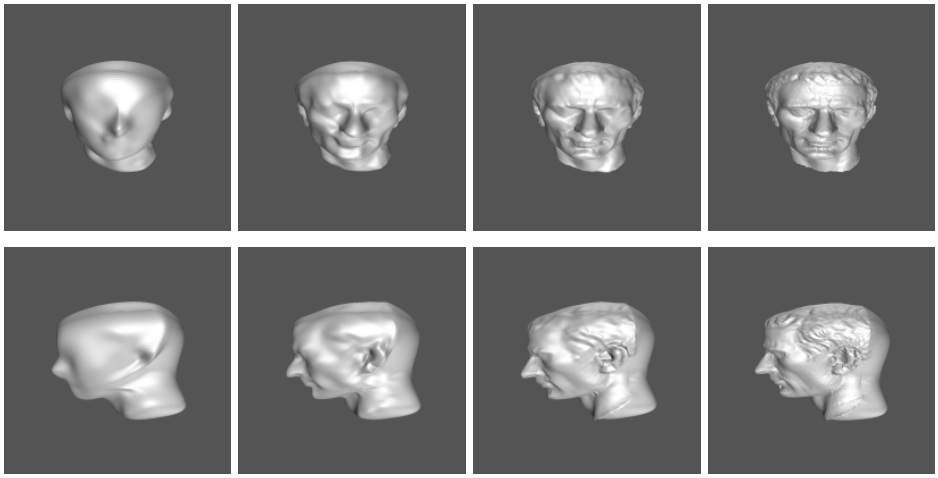


(a) (b) (c) (d)

Figure 6.10: Head model with control points placed to match prominent face features.

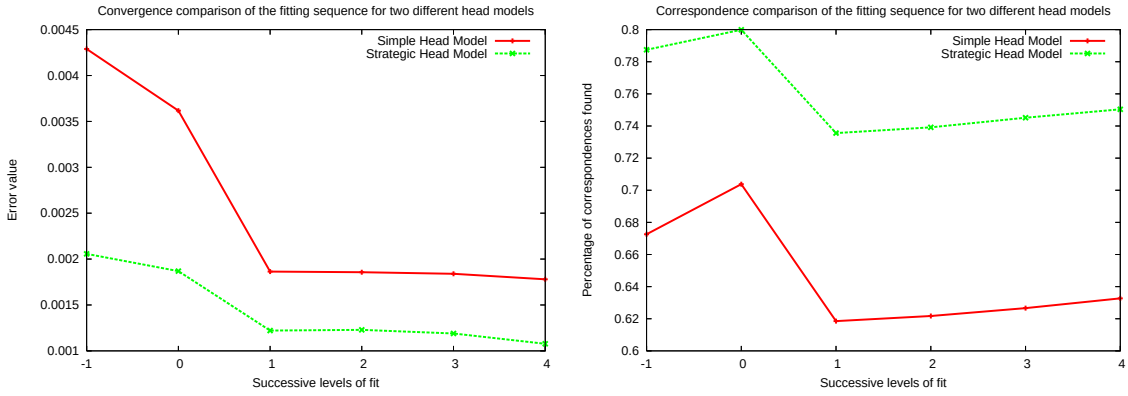
The experiments with the last model show that it is important to use a generic model with control points placed at "strategic" locations, if there are prominent features present in the data. As the fitting is performed hierarchically, it makes sense to schedule the correspondence search to some extent, *i.e.*, force the most important correspondences to be established very early in the fitting process. This would not be such a big issue if the correspondences were established manually and in advance, however, since we are aiming at a fully automatic process, the base mesh vertex placement is an important consideration.

To illustrate the difference in the fit quality between the simple head model and the model which samples the prominent features first, we evaluated the fit error using the error function in Equation 6.3. Figure 6.12 shows the resulting graphs with noticeable difference in the speed of convergence for the two models.



(a) I_1 (b) I_2 (c) I_3 (d) I_4

Figure 6.11: Upper row: The fitting stages with the model where control points have been placed at strategic locations (compare with equivalent levels of Figure 6.9). Lower row: view from the side to show the complete model even where data is missing



(a) (b)

Figure 6.12: Comparison of the fit convergence for two different head models. (a) Error function convergence. (b) Percentage of correspondences found on every level.

In this section, we have illustrated the fitting method on detailed and accurate data. We now show how our fitting method can be applied to noisy stereo data for the purpose of disparity map completion.

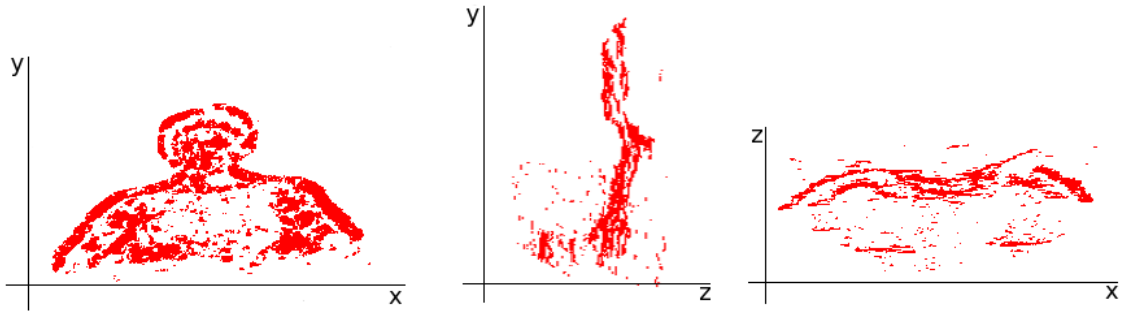


Figure 6.13: Reconstructed disparity data after consistency check, shown as 3-D points from three different viewpoints: front, side and top. The patchy and noisy nature of the disparity data presents a difficult challenge for model fitting.

6.4 Fitting to Stereo Data

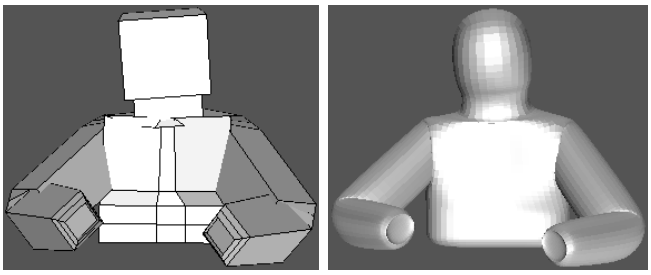
In this section, we present the results of applying our fitting method (Algorithm 6.1) to noisy and incomplete wide-baseline stereo data using a generic model of the upper human body.

In comparison with the results on the synthetic data presented in the previous section, fitting an upper body model to stereo data presents a different challenge. As the generic model has an articulated pose, this must first be estimated before the fit can be attempted. In our case, this has been achieved by using the PSO pose estimation algorithm described in Section 5.3.5.

In comparison with range data from 3-D scanners, typically used for surface based modelling [44, 112, 12], fitting to stereo data presents a more demanding problem because the data is patchy and noisy, as can be seen in Figure 6.13, and can only be trusted to a limited extent. The aim of the fitting process is not to produce a visually pleasing result with a high level of detail but instead to deform the model so that it overall better represents the data without replicating the noisy values. The appropriate level of detail is dictated by a tradeoff between allowing the subsequent process (here, view synthesis) to achieve a given quality level and simultaneously discarding the information coming from the outliers in the data.

The stereo data used in our experiments comes from the images of the upper human

Figure 6.14: Examples of patchy disparity map data resulting from a correspondence search algorithm. The data shown in this figure passed the consistency check and the regions with no available disparity data are shown in black.



(a) (b)

Figure 6.15: (a) The base mesh of the upper body subdivision surface model. (b) The limit surface approximation after 3 levels of subdivision.

body, acquired by our camera setup described in Section 4.3. Disparity data (see Figure 6.14) was generated by the correspondence search algorithm described in Section 3.4 and the corresponding 3-D points reconstructed using the 3-D reconstruction algorithm described in Section 3.5. Our generic model is a subdivision model of the upper body shown in Figure 6.15. The base mesh of the generic model is subdivided 3 times to generate an approximation of the limit surface which is sufficiently smooth for our purposes. The model dimensions are adjusted to match individual test subjects as shown in Figures 6.18, 6.19 and 6.20.

6.4.1 Finding Correspondences

As stated in Algorithm 6.1, the correspondences between the data and the model are established during the fitting process. Stereo data is view-dependent, comes from an articulated structure with self-occlusions and contains outliers. Finding correspondences between the

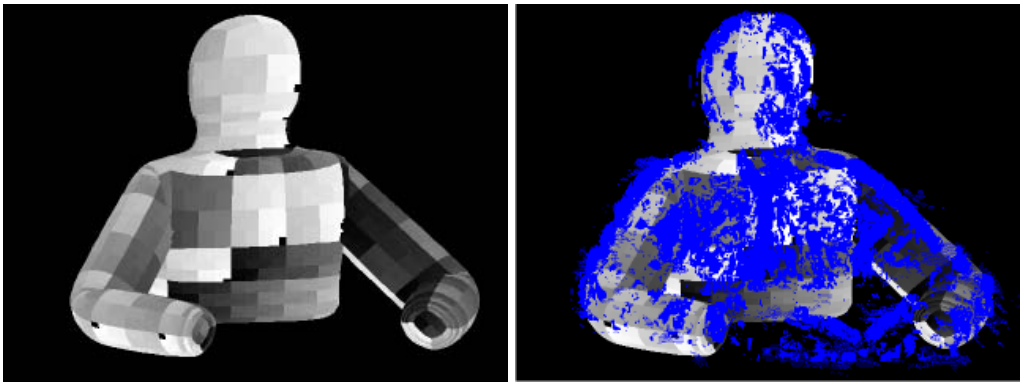


Figure 6.16: The labelled model projection used to find correspondences between mesh faces and the disparity data points. The numerical labels have been converted to a grayscale value between 0 and 255 for visualisation.

model and the data therefore presents a much more difficult challenge than the equivalent process in the range data experiments presented in Section 6.3 and must be handled differently.

To ensure that the data points are associated with those parts of the human body which generated them, we exploit the fact that stereo data is view-dependent. We render the front-facing parts of the model back to front onto the image plane to determine the visible faces in a particular view. Pixels within each visible face are assigned a unique numerical label in a sequential order. Figure 6.16(a) shows an example of the projected model with labelled faces, where each numerical label has been converted to a grayscale value between 0 and 255 for visualisation.

The stereo data is then projected on top of the labelled faces as shown in Figure 6.16(b). Individual data points are given the label of the face to which they project. The final correspondences are established between the data points and model mesh vertices by connecting each data point with the closest vertex of the corresponding face.

Data points which project on the background are not associated with any face and discarded from the fit. However, because the correspondences are re-estimated at every level of the fit due to the change in model mesh density, the majority of such points will be

assigned to a corresponding face at a later stage, when the model has deformed to fit the shape of the data better. It is also important to note that the points which project on the background belong to the extreme regions, where the imaged surface sharply bends away from the camera and are thus likely to be noisy. Discarding the points which do not project onto any model face at a later stage of the fit therefore also contributes to reducing the number of outliers in the data.

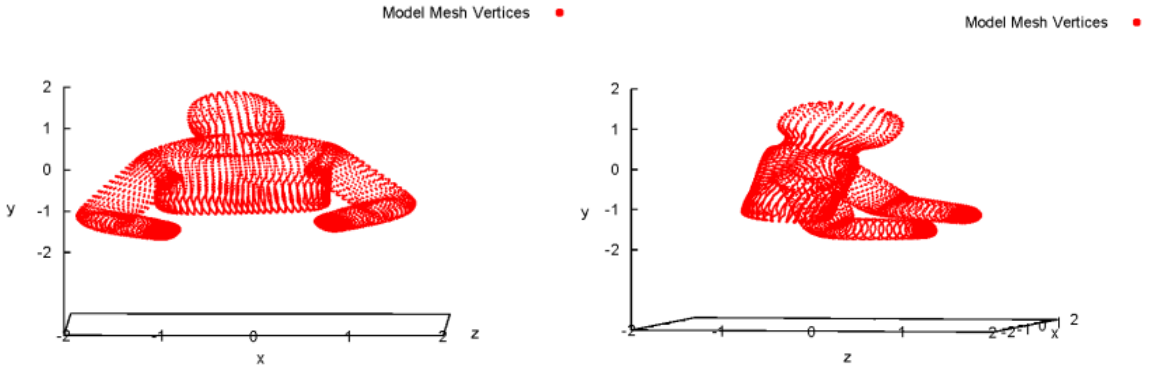
Figure 6.17 illustrates the process of finding correspondences. The 3-D model vertices are shown in red and the 3-D data points in blue. Figure 6.17(b) illustrates the view-dependent nature of the stereo data. The resulting correspondence pairs are shown as green lines connecting appropriate data points and mesh vertices (Figure 6.17(c)).

Because we do not model hands with our generic subdivision surface model, erroneous correspondences are found between the stereo data originating from the hands and the mesh faces of the torso, on which those data points project. The stereo data also contains outliers (erroneous matches which survived the consistency check). As a result, the established correspondences must be examined for potential erroneous matches before they are used for the fit.

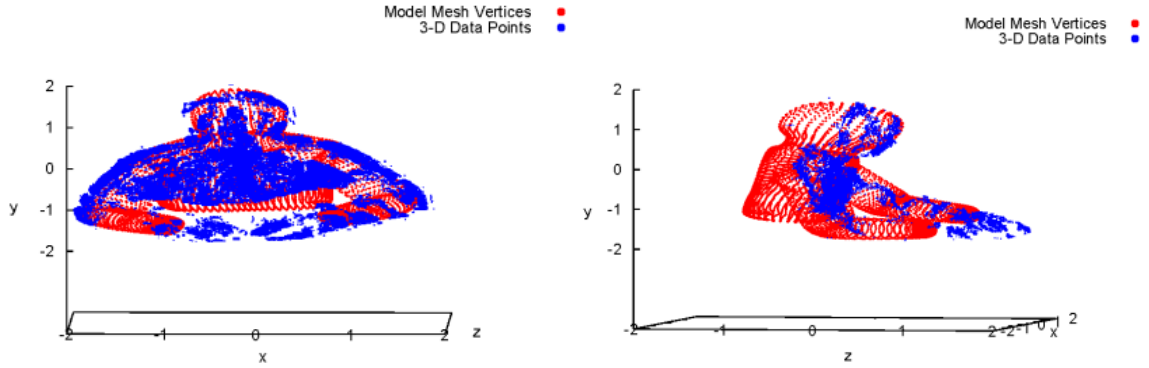
We remove the most obvious outliers by placing a threshold value on the Euclidean distance between the data point and its corresponding mesh vertex. The value of the threshold is assigned experimentally, taking into account the quality of the data. Only those correspondences for which the Euclidean distance is smaller than the experimentally defined threshold are kept. This resolves the hand data problem as well as removes the obvious "salt-and-pepper" outliers.

6.4.2 Fitting Results

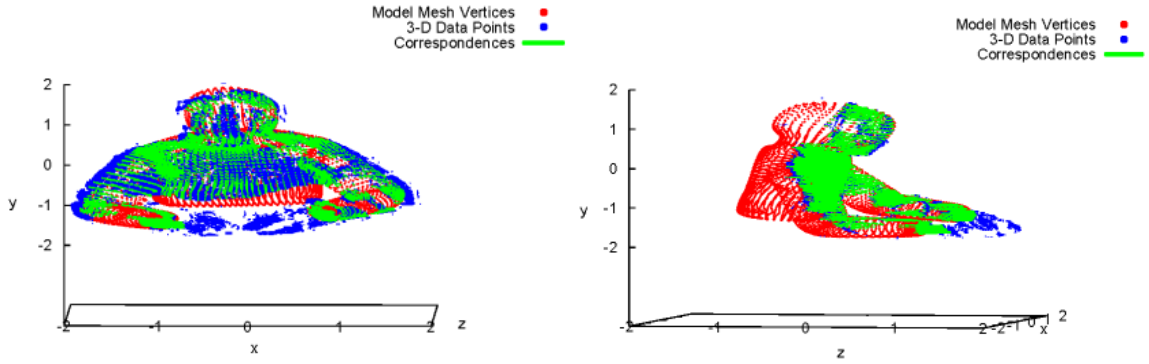
We fit the generic model of the upper body to stereo data originating from different upper body poses. First, the poses are estimated using the PSO algorithm described in Chapter 5 and then the models are fit to the stereo data as described above.



(a) Model mesh vertices.



(b) Model with stereo data.



(c) Correspondences between stereo data and the model.

Figure 6.17: Finding correspondences between the data points and model mesh vertices. (a) Model mesh vertices shown from the front and the side. (b) Stereo data points shown together with the model to illustrate the view-dependent nature of the data. (c) The corresponding pairs denoted by the green lines connecting mesh vertices with corresponding data points.

In comparison with the generic models used for range data fitting experiments, presented in Section 6.3, the generic model of the human body consists of a more complex base mesh (see Figure 6.15(a)). Such a base mesh is required to realistically model the shape of the human body. As a result, through the iteration levels of the fitting algorithm, the base mesh quickly becomes very dense. Because the fitting method is local and the data patchy, this means that only a limited number of iteration levels can be used if one is to prevent the model from explicitly replicating the patchy nature of the data.

Figures 6.18, 6.19 and 6.20 illustrate the model deformation for three different test subjects as it progresses through successive levels of the fitting algorithm. The data was obtained by running a correspondence search on the stereo pair of images which were used to estimate the model pose. While the model on level 1 of the fit is still only approximating the data, the base mesh on level 2 becomes so dense that it begins to deform consistently with the patchy data instead of providing a smooth approximation to it.

The level 2 fit in Figures 6.18(c), 6.19(c) and 6.20(c) demonstrates a drawback of the fitting method. In order to smoothly approximate the noisy and patchy data, one must begin with a base mesh that is as simple as possible, containing a small number of large faces, in order for the vertex deformation to affect a larger area of the surface and so approximate the data more smoothly.

The denser the mesh, the more constrained the influence of the single vertex deformation, which eventually leads to modelling the individual patches of data points. While a simpler starting mesh will allow for a smoother approximation, it will not, however, always enable a fully realistic modelling of the human body shape which we require to complete the missing data regions.

In our experiments we deform the model to level 1. This includes the initialisation step with quasi-interpolation and one step of the fitting loop (see Algorithm 6.1). Level 1 deformation sufficiently modifies the model to better reflect the test subject's body proportions, while not explicitly modelling the detail in the data. Given the incomplete nature of

the data, this is all we require. Our generic model shown for different subjects in Figures 6.18(a), 6.19(a) and 6.20(a) provides a compromise between the mesh simplicity and the realistic representation of human shape and lends itself well to the fitting method. Model fitting results for different poses and subjects are shown in Figures 6.21, 6.22 and 6.23.

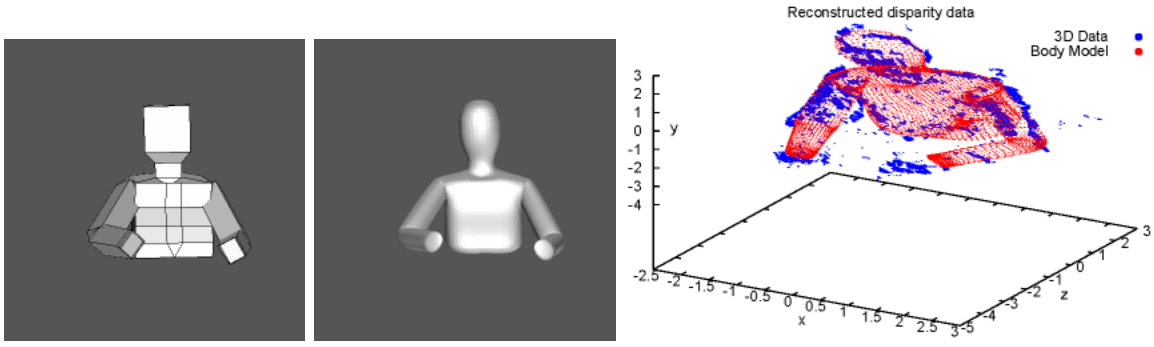
6.5 Conclusion

In this chapter we presented our method for fitting a generic subdivision surface model to unstructured, incomplete and noisy cloud of data points.

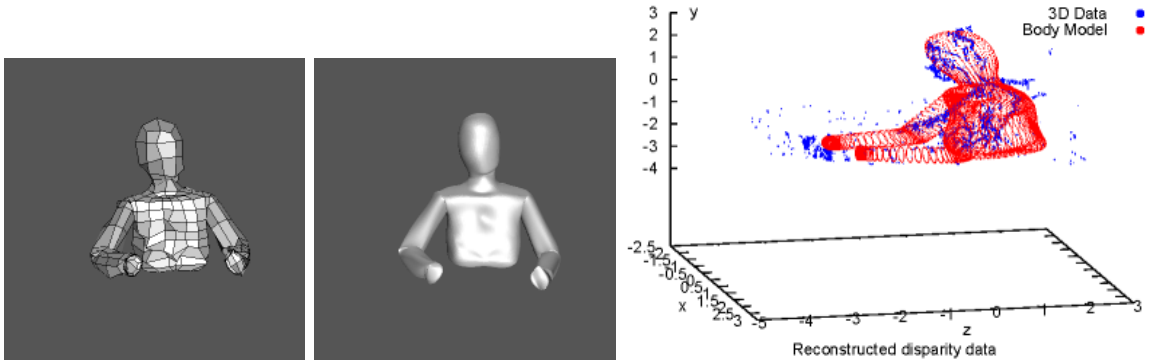
We first demonstrated the method on a set of range data which has been post-processed to remove any outliers and inconsistencies owing to the scan process. The high quality of the range data allowed us to demonstrate that the presented fitting method is capable of replicating the input data in 3-5 iterations, depending on the desired level of detail. We also showed that, when using accurate and dense data such as the range data example, the final result of the fit is not sensitive to minor variations in the starting conditions used for the model, such as its size and relative position with respect to the data (inside, outside, at the data) and its mesh density.

We used a simple generic head model to demonstrate that the method can be used to complete missing parts, in this case the back of the head. As the simple model proved to be insufficient, we designed a more elaborate model which better imitated the required head shape. This enabled us to more realistically complete the missing back of the head and also led to the idea of strategic control points which allowed us to force the correspondences with more prominent data features (*e.g.*, nose and ears) to be established first. This improved the quality of the final result and the speed of fit convergence.

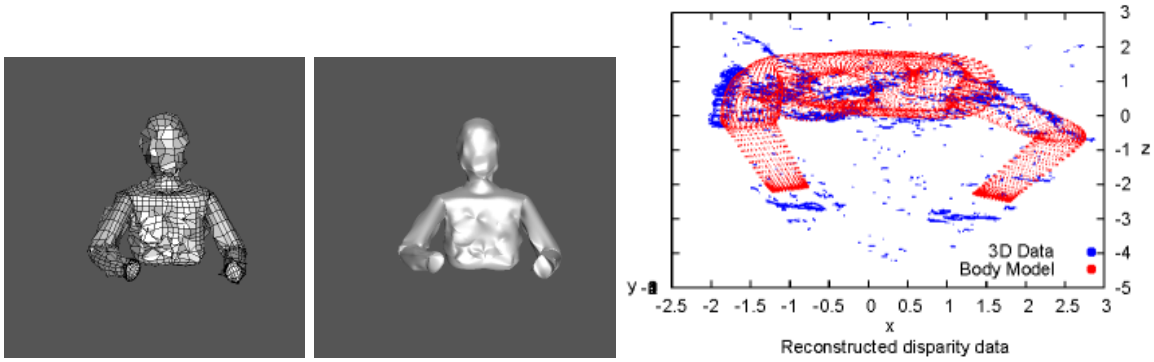
While range data was used to demonstrate the potential of the fitting method, our primary aim was to fit generic models to noisy and patchy stereo data. To this end, we described the process of establishing correspondence pairs between the stereo data points and



(a) Level -1

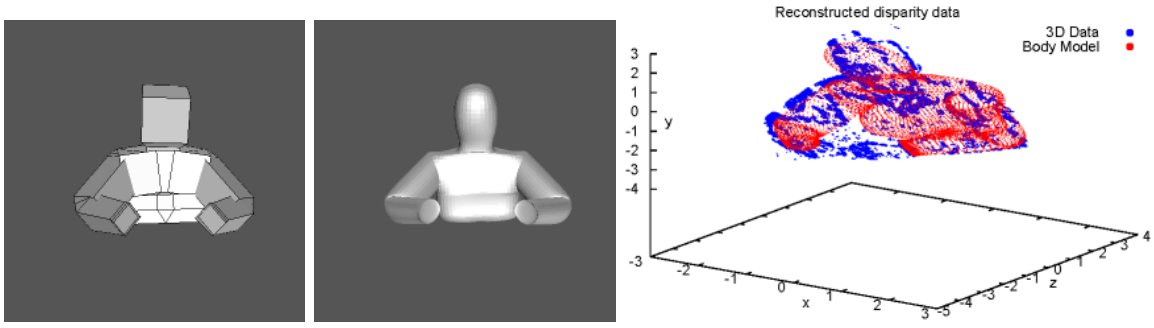


(b) Level 1

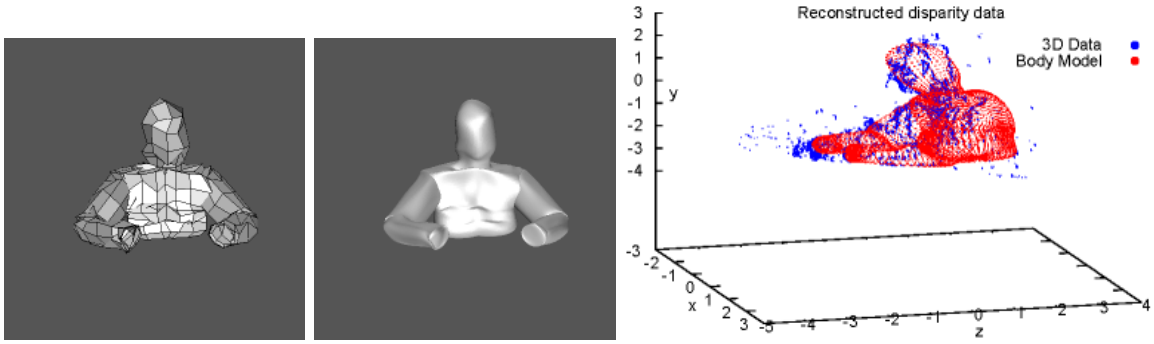


(c) Level 2

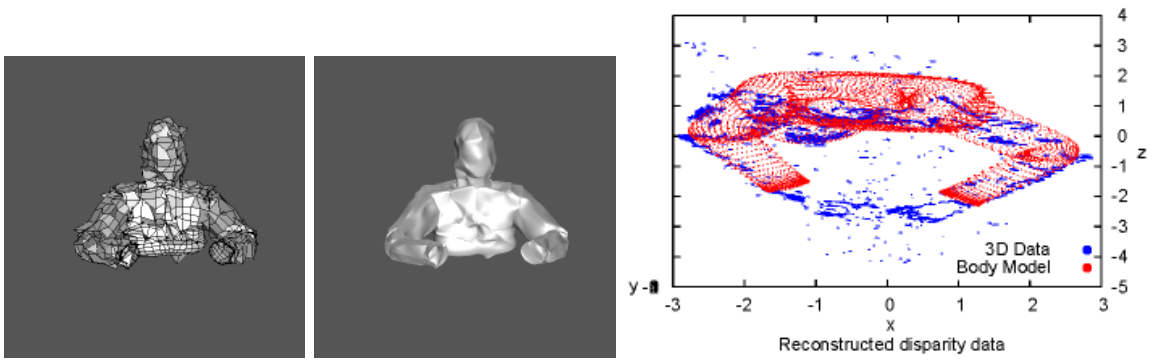
Figure 6.18: On the left, the 3-D upper body model for subject *S* and its fit stages shown in succession from top to bottom. On the right, the data is displayed from three different viewpoints together with the generic model to facilitate the understanding of its origin and quality.



(a) Level -1

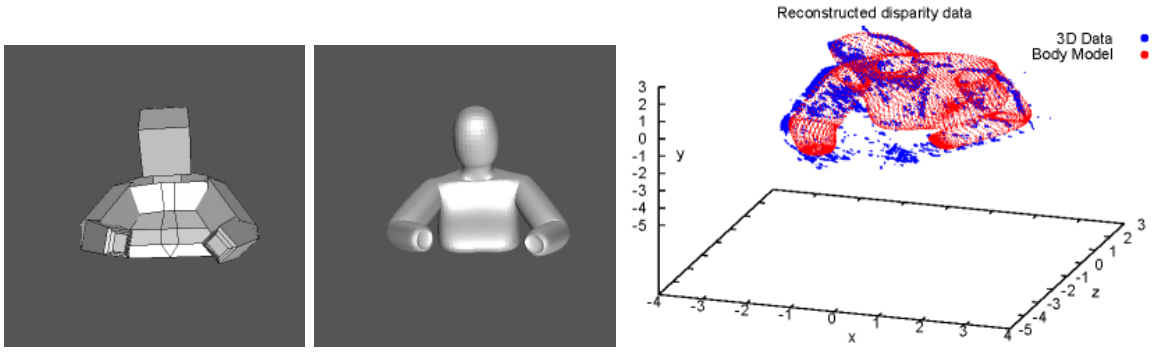


(b) Level 1

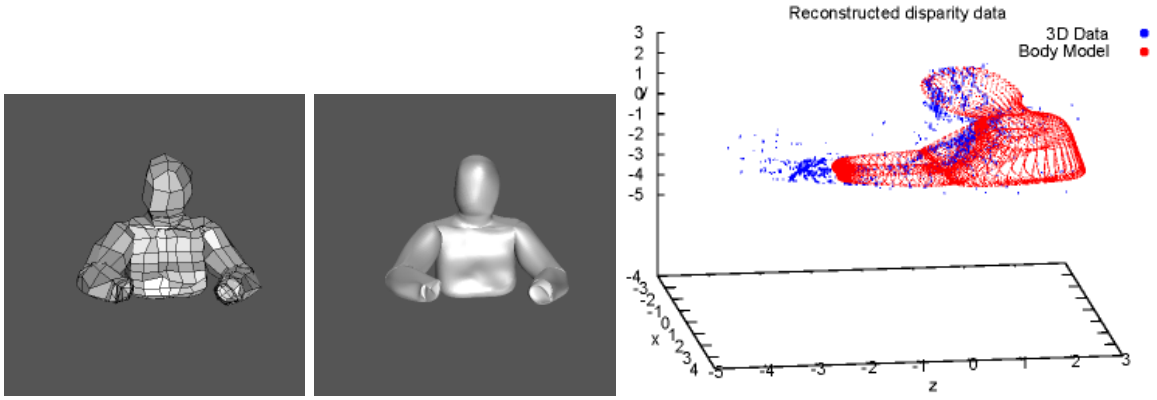


(c) Level 2

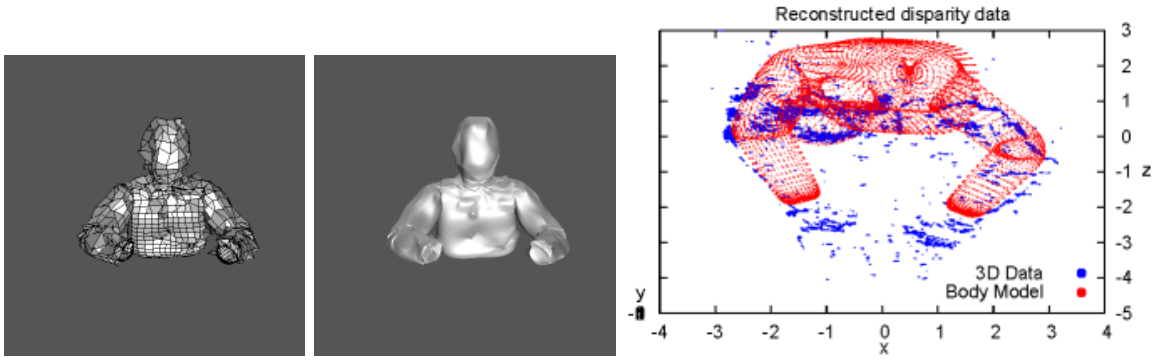
Figure 6.19: On the left, the 3-D upper body model for subject *D* and its fit stages shown in succession from top to bottom. Again, the data is shown in the column on the right.



(a) Level -1



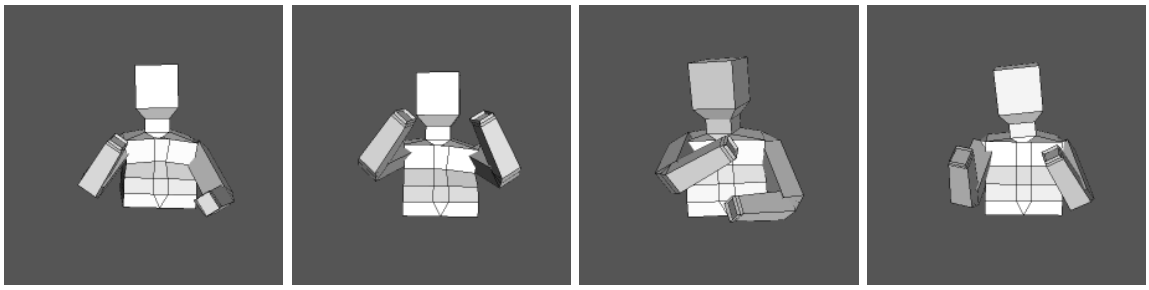
(b) Level 1



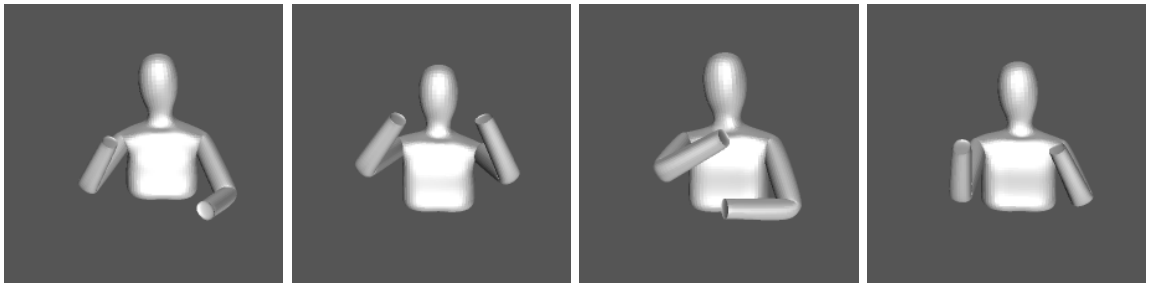
(c) Level 2

Figure 6.20: On the left, the 3-D upper body model for subject *G* and its fit stages shown in succession from top to bottom. Like in the previous two figures, the data is shown with the corresponding generic model in the column on the right.

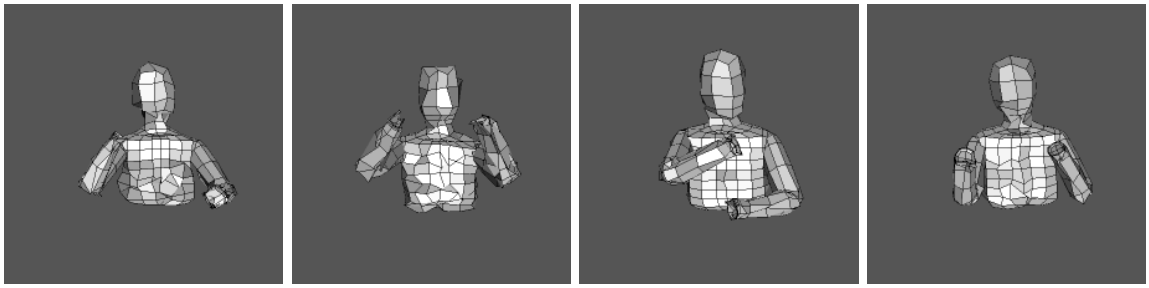
(a) Input image as seen by camera 2.



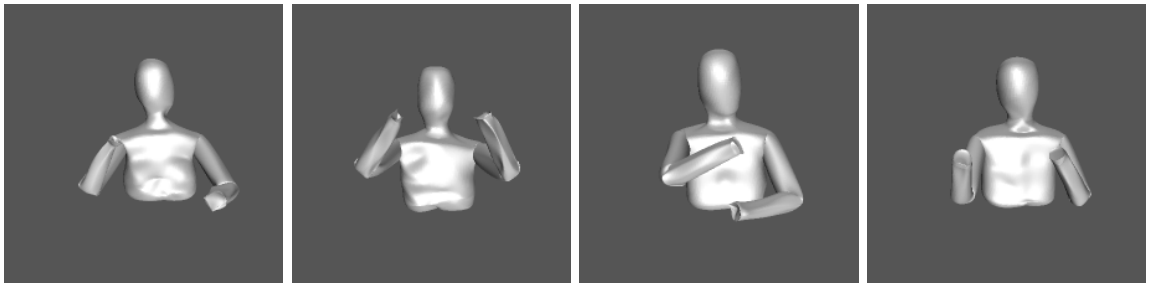
(b) The base mesh of the generic model at the initialisation stage.



(c) The subdivision surface (the skin) of the generic model at the initialisation stage.

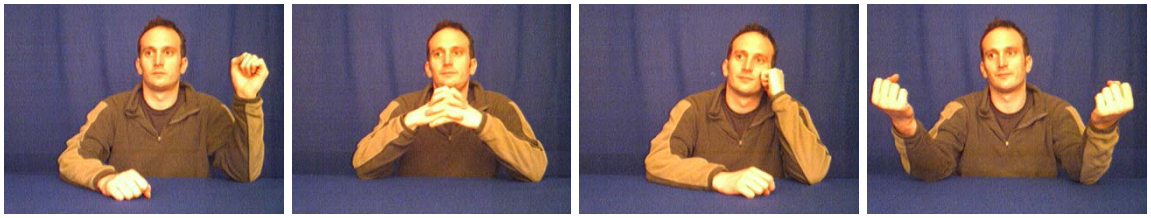


(d) The base mesh at level 1 of the fitting algorithm.

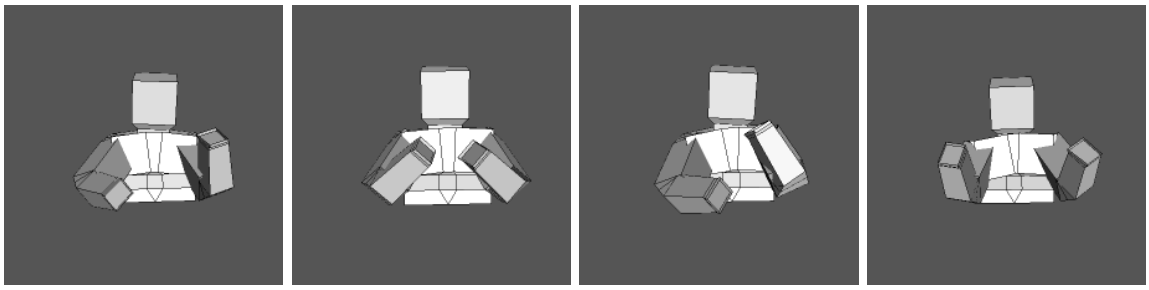


(e) The subdivision surface at level 1 of the fitting algorithm.

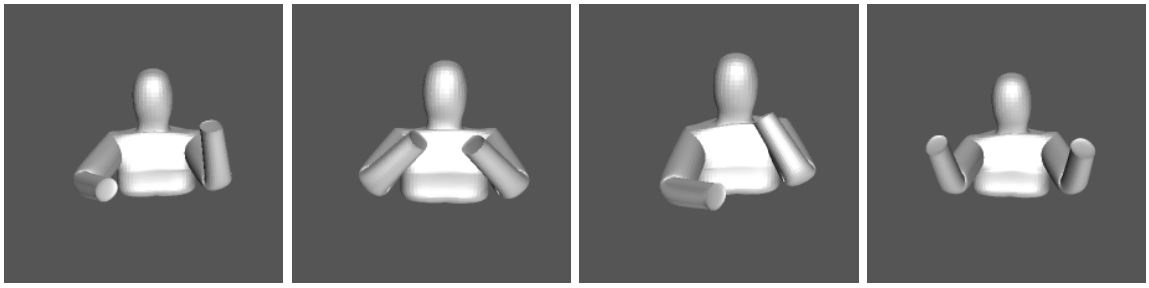
Figure 6.21: 3-D upper body models for subject *S* after the fit to stereo data. Each column shows one body pose.



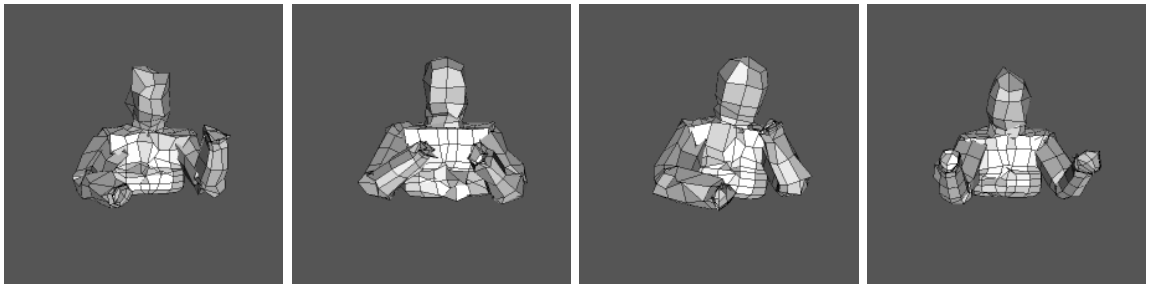
(a) Input image as seen by camera 2.



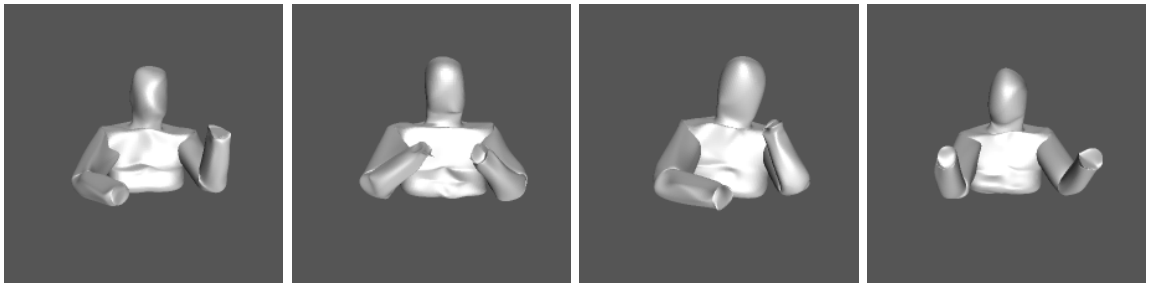
(b) The base mesh of the generic model at the initialisation stage.



(c) The subdivision surface (the skin) of the generic model at the initialisation stage.

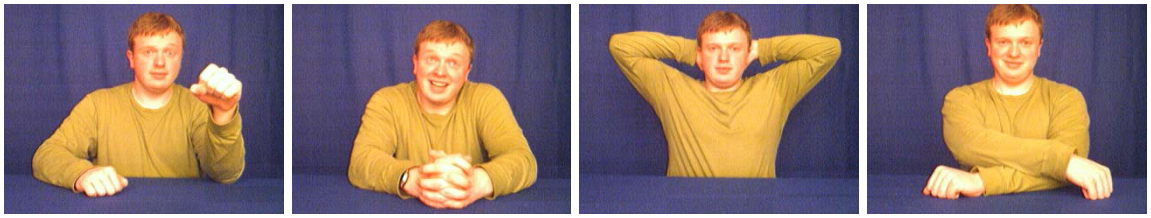


(d) The base mesh at level 1 of the fitting algorithm.

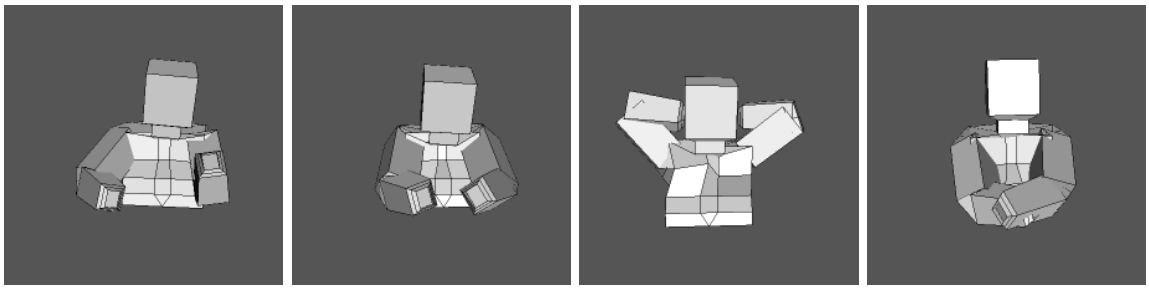


(e) The subdivision surface at level 1 of the fitting algorithm.

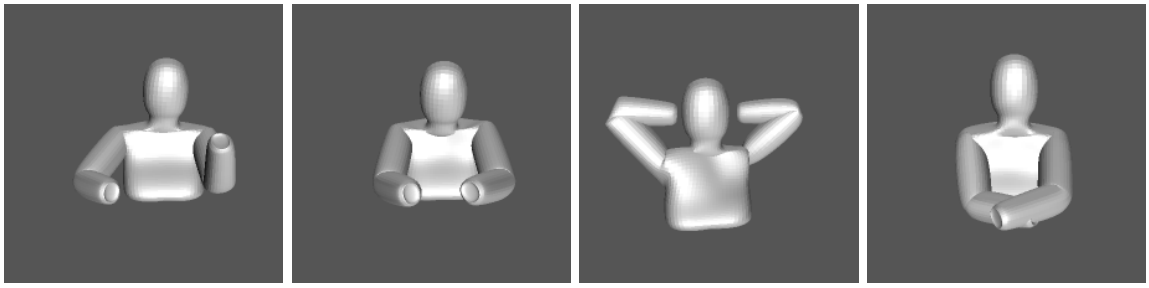
Figure 6.22: 3-D upper body models for subject *D* after the fit to stereo data. Each column shows one body pose.



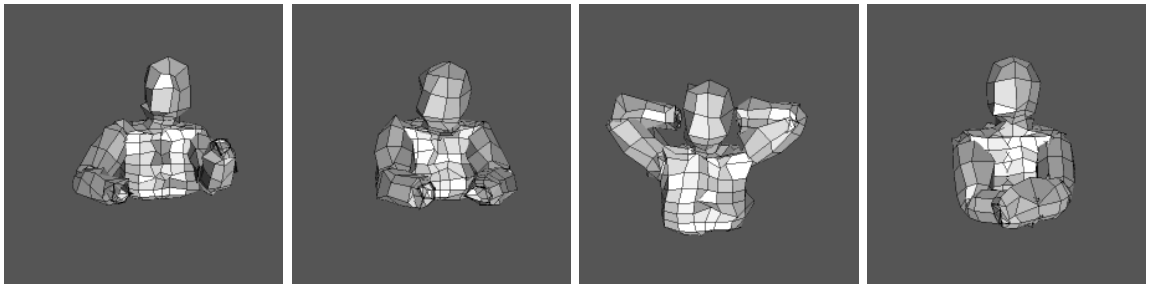
(a) Input image as seen by camera 2.



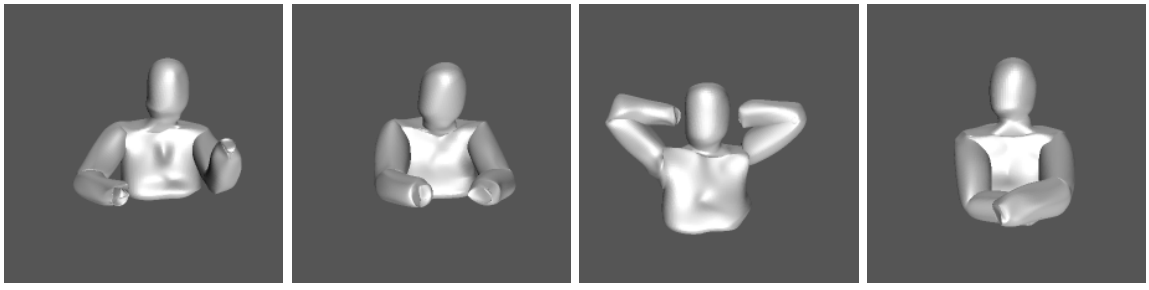
(b) The base mesh of the generic model at the initialisation stage.



(c) The subdivision surface (the skin) of the generic model at the initialisation stage.



(d) The base mesh at level 1 of the fitting algorithm.



(e) The subdivision surface at level 1 of the fitting algorithm.

Figure 6.23: 3-D upper body models for subject *G* after the fit to stereo data. Each column shows one body pose.

model mesh vertices, where we took advantage of the view-dependent nature of the data. We then demonstrated the fitting results on the stereo data of the upper human body.

In Chapter 8, we use the deformed human body model to generate a complete disparity map and use it for high-quality novel view synthesis. First, however, we look at the pose estimation and model fitting in disparity space in Chapter 7.

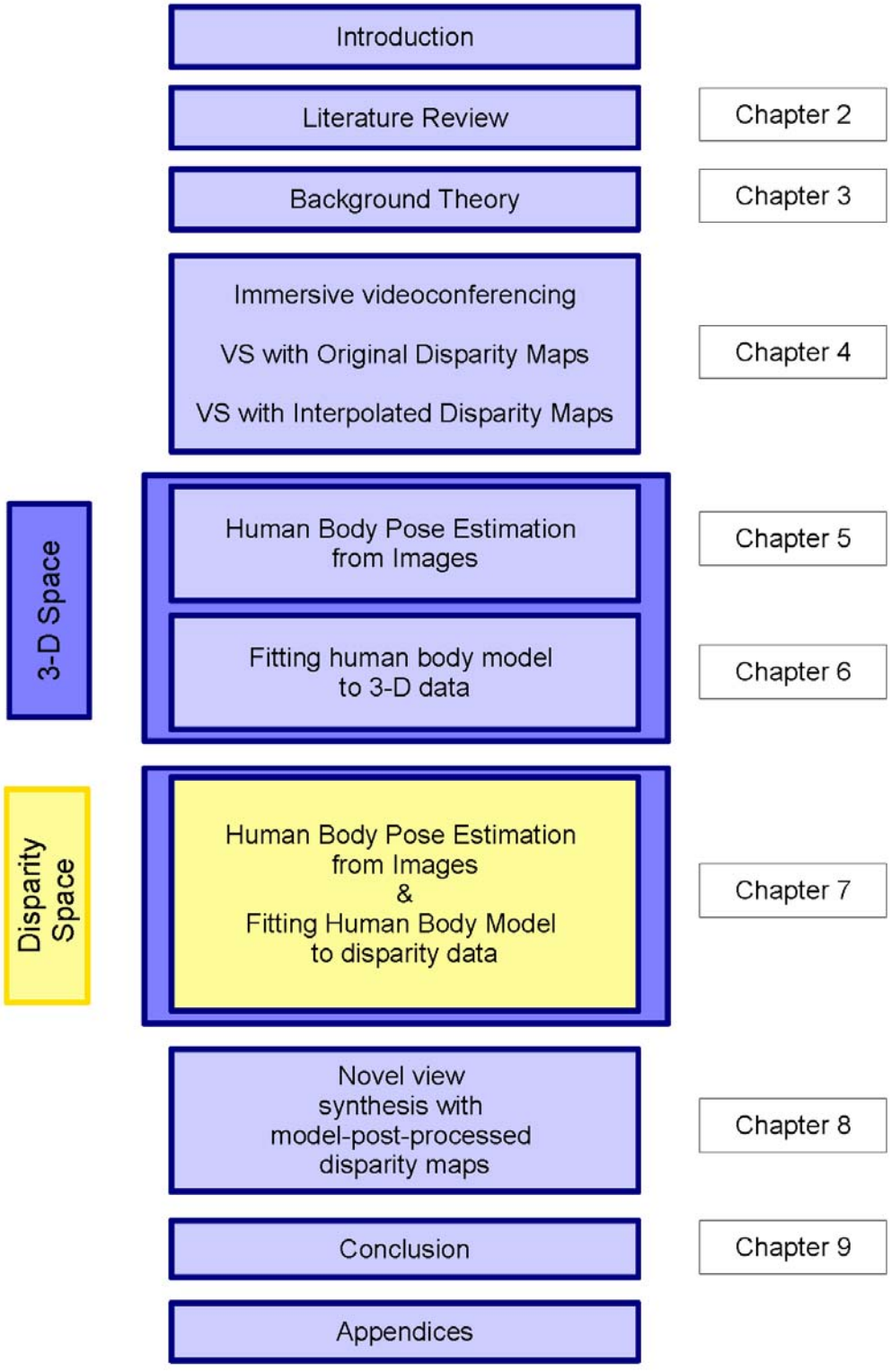


Figure 6.24: The thesis structure diagram.

Chapter 7

Disparity Space

7.1 Introduction

In Chapters 5 and 6, we presented in detail how a generic model can be used to complete patchy disparity data for the purpose of better novel-view synthesis. In this chapter, we present the idea of completing the patchy data in its space of origin, the disparity space.

We show that, in disparity space, it is possible to model the human body, perform articulated pose estimation, as well as fit the model to disparity data. Completing the disparity data in disparity space allows us to skip the step of 3-D reconstruction which not only reduces the computational complexity of the entire approach but also adds to its accuracy as the inevitable 3-D reconstruction error is avoided.

We begin by presenting the concept of disparity space and its relationship with the 3-D space in Section 7.2. We then present the disparity space version of the articulated human body model in Section 7.3 and discuss its pose estimation in Section 7.4. The disparity model fit to disparity data directly is presented in Section 7.5 and a comparison between the 3-D and disparity space approach in Section 7.6. We conclude with Section 7.7.

7.2 Disparity Space

Let us assume that a 3-D point $\boldsymbol{X} = (X, Y, Z, 1)^\top$ is viewed by two distinct cameras, left camera with the projection matrix P_l and right camera with the projection matrix P_r. The image of the point $\boldsymbol{X}$ is defined as $\boldsymbol{x}_l = (x_l, y_l, 1)^\top \simeq P_l \boldsymbol{X}$ and $\boldsymbol{x}_r = (x_r, y_r, 1)^\top \simeq P_r \boldsymbol{X}$, in the left and right camera's image plane, respectively, where $\simeq$ denotes equality up to a scale factor.

The corresponding points $\boldsymbol{x}_l$ and $\boldsymbol{x}_r$ are related by a *disparity* which, in a general case, is defined as:

$$d(\boldsymbol{x}_r, \boldsymbol{x}_l) = \boldsymbol{x}_r - \boldsymbol{x}_l = (x_r - x_l, y_r - y_l). \tag{7.1}$$

When the images are rectified (see Section 3.3), the two corresponding points lie on the same scanline, $y_r - y_l = 0$, and the disparity simplifies to a displacement along that scanline:

$$d(\boldsymbol{x}_r, \boldsymbol{x}_l) = x_r - x_l. \tag{7.2}$$

We define the disparity space as a three-dimensional space $\mathbb{D}^3 = \{x, y, d\}$, which exists for a given rectified stereo pair. The first two dimensions, x and y, are defined by the column and row dimensions of the rectified image planes, respectively. The disparity dimension d is represented by the displacement of the corresponding points, as defined in Equation (7.2).

7.2.1 Mapping Between the 3-D Space and Disparity Space

Let us denote the 3-D space as $\mathbb{R}^3$ and the disparity space as $\mathbb{D}^3$. The mapping P_D between 3-D space and disparity space

$$P_D : \mathbb{R}^3 \rightarrow \mathbb{D}^3 \tag{7.3}$$

can be represented with a 4×4 homogeneous matrix which we derive as follows.

We assume, without loss of generality, a specific form of the rectified projection matrices [61]. Let the left and right rectified camera projection matrices, $\tilde{P}_l$ and $\tilde{P}_r$, be written as:

$$\tilde{P}_l = \begin{pmatrix} p^l_{11} & p^l_{12} & p^l_{13} & p^l_{14} \\ p^l_{21} & p^l_{22} & p^l_{23} & p^l_{24} \\ p^l_{31} & p^l_{32} & p^l_{33} & p^l_{34} \end{pmatrix} \quad \tilde{P}_r = \begin{pmatrix} p^r_{11} & p^r_{12} & p^r_{13} & p^r_{14} \\ p^l_{21} & p^l_{22} & p^l_{23} & p^l_{24} \\ p^l_{31} & p^l_{32} & p^l_{33} & p^l_{34} \end{pmatrix}. \tag{7.4}$$

Projecting a 3-D point $\boldsymbol{X} = (X, Y, Z, 1)^\top \in \mathbb{R}^3$ into left and right view gives the left and right image points, $\boldsymbol{x}_l$ and $\boldsymbol{x}_r$:

$$\boldsymbol{x}_l = \begin{pmatrix} x_l \\ y_l \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{p^l_{11}X + p^l_{12}Y + p^l_{13}Z + p^l_{14}}{p^l_{31}X + p^l_{32}Y + p^l_{33}Z + p^l_{34}} \\ \frac{p^l_{21}X + p^l_{22}Y + p^l_{23}Z + p^l_{24}}{p^l_{31}X + p^l_{32}Y + p^l_{33}Z + p^l_{34}} \\ 1 \end{pmatrix} \simeq \tilde{P}_l \boldsymbol{X} \tag{7.5}$$

$$\boldsymbol{x}_r = \begin{pmatrix} x_r \\ y_r \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{p^r_{11}X + p^r_{12}Y + p^r_{13}Z + p^r_{14}}{p^l_{31}X + p^l_{32}Y + p^l_{33}Z + p^l_{34}} \\ \frac{p^l_{21}X + p^l_{22}Y + p^l_{23}Z + p^l_{24}}{p^l_{31}X + p^l_{32}Y + p^l_{33}Z + p^l_{34}} \\ 1 \end{pmatrix} \simeq \tilde{P}_r \boldsymbol{X} \tag{7.6}$$

A point $\boldsymbol{D} \in \mathbb{D}^3$, written in homogeneous coordinates, is defined as:

$$\boldsymbol{D} = \begin{pmatrix} x_l \\ y_l \\ x_r - x_l \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{p^l_{11}X + p^l_{12}Y + p^l_{13}Z + p^l_{14}}{p^l_{31}X + p^l_{32}Y + p^l_{33}Z + p^l_{34}} \\ \frac{p^l_{21}X + p^l_{22}Y + p^l_{23}Z + p^l_{24}}{p^l_{31}X + p^l_{32}Y + p^l_{33}Z + p^l_{34}} \\ \frac{p^r_{11}X + p^r_{12}Y + p^r_{13}Z + p^r_{14}}{p^l_{31}X + p^l_{32}Y + p^l_{33}Z + p^l_{34}} - \frac{p^l_{11}X + p^l_{12}Y + p^l_{13}Z + p^l_{14}}{p^l_{31}X + p^l_{32}Y + p^l_{33}Z + p^l_{34}} \\ 1 \end{pmatrix}, \tag{7.7}$$

and the transformation P_D, for which

$$\boldsymbol{D} \simeq P_D \boldsymbol{X}, \quad \boldsymbol{X} \in \mathbb{R}^3, \boldsymbol{D} \in \mathbb{D}^3 \tag{7.8}$$

as

$$P_D = \begin{pmatrix} p^l_{11} & p^l_{12} & p^l_{13} & p^l_{14} \\ p^l_{21} & p^l_{22} & p^l_{23} & p^l_{24} \\ p^r_{11} - p^l_{11} & p^r_{12} - p^l_{12} & p^r_{13} - p^l_{13} & p^r_{14} - p^l_{14} \\ p^l_{31} & p^l_{32} & p^l_{33} & p^l_{34} \end{pmatrix}. \tag{7.9}$$

When working with a *canonical* parallel stereo setup, where the intrinsic parameters matrix K is the same for both cameras:

$$K = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{7.10}$$

and the left and right camera matrices are defined as $\tilde{P}_l = K[I|\mathbf{0}]$ and $\tilde{P}_r = K[I|\mathbf{t}]$, respectively, where $\mathbf{t} = (t_x, 0, 0)$ is the baseline translation along the x-axis of the world coordinate system:

$$\tilde{P}_l = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \tilde{P}_r = \begin{pmatrix} f & 0 & 0 & ft_x \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{7.11}$$

our projective transformation P_D in Equation (7.9) simplifies to:

$$P_D = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & ft_x \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{7.12}$$

which is the equation derived by Demirdjian and Darrell for rigid motion in [47].

Although the treatment of disparity space in [47] is very similar to our own, we would like to emphasize that the presented derivation of the transformation P_D, as shown in Equation (7.9), has been achieved independently without the knowledge of [47], with the aim of modelling an articulated motion and structure (the shape and motion of a human body) in disparity space. The work of Demirdjian and Darrell has subsequently been drawn to our attention by an anonymous reviewer and we duly acknowledge it here as its existence has further motivated the research presented in this chapter.

The transformation P_D shown in Equation (7.9) is a 4×4 projective transformation which represents the link between the 3-D space and disparity space and allows us to transform the structure and the motion of objects from 3-D space, where we have an intuitive understanding of their nature and properties, to the disparity space, where they are computationally cheaper to estimate and manipulate as well as more accurate to measure, as shown by related research work [8, 47, 49, 71].

7.2.2 Homogeneous Transformations in Disparity Space

Let $\mathbf{X} \in \mathbb{R}^3$, P_D be as in Equation (7.9), and $A = [R|\mathbf{t}]$ be a known homogeneous transformation in $\mathbb{R}^3$, transforming $\mathbf{X}$ into $A\mathbf{X}$. Homogeneous transformation B, which transforms $\mathbf{D} \in \mathbb{D}^3$ to $B\mathbf{D} \in \mathbb{D}^3$, can then be derived as follows (see the transformation diagram in Figure 7.1):

$$\begin{aligned} BP_D\mathbf{X} &= P_DA\mathbf{X} \\ BP_D &= P_DA \\ B &= P_DAP_D{}^{-1} \end{aligned} \tag{7.13}$$

This result shows that the points can be rotated and translated in disparity space directly if, for every known homogeneous transformation A in 3-D space, a corresponding transformation B is computed as in Equation (7.13).

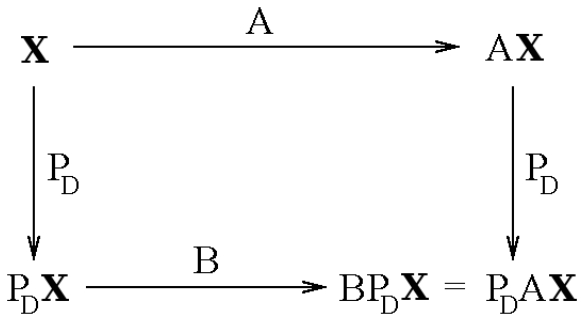


Figure 7.1: Transformation diagram

Equation (7.13) was derived without the knowledge of the work in [47], which was drawn to our attention later. The consequence of Equation (7.13) is that *a known homogeneous transformation A, influencing a stereo-reconstructed point in* $\mathbb{R}^3$*, can be applied to the same point in disparity space* $\mathbb{D}^3$ *directly, by computing its similarity transformation B.* This consequence has several important implications for the research presented in this chapter and we discuss these implications next.

7.2.3 Why Use Disparity Space?

The aim of this work is to find a way of post-processing the incomplete disparity data resulting from a stereo correspondence search. Post-processing using interpolation techniques without any assumptions about the scene can only recover the missing data to a limited extent, as we have shown in Chapter 4. Ideally, some form of *a priori* knowledge of the scene should be used to help with the recovery of missing data.

We chose to model the *a priori* knowledge using a generic human body model, which we fit to the available disparity data. The missing disparity data is then recovered by sampling the deformed model, thereby making use of the *a priori* information provided.

The *input data* to the model fitting algorithm is the *disparity data*. The expected *output* is a complete *disparity map*, which can be used for novel view synthesis. If the generic human body is modelled in 3-D, the input data has to be converted to 3-D points via 3-D reconstruction first. Once the model is deformed to fit the data, it then has to be projected back onto the stereo pair of image planes to allow for a complete disparity map to be generated, which is then used for novel-view synthesis. This process is illustrated on the right side of the diagram in Figure 7.2.

Computational Savings

The use of the 3-D model as described contains several redundant steps. Ideally, the human body should be modelled in disparity space directly, so that the fitting can be done to the disparity data, not the reconstructed 3-D data. This means that the 3-D reconstruction step can be completely avoided. Additionally, producing a complete disparity map from the deformed model in disparity space means sampling the disparity model directly, instead of the additional two perspective projections required when using the 3-D model. This approach is shown on the left side of the diagram in Figure 7.2.

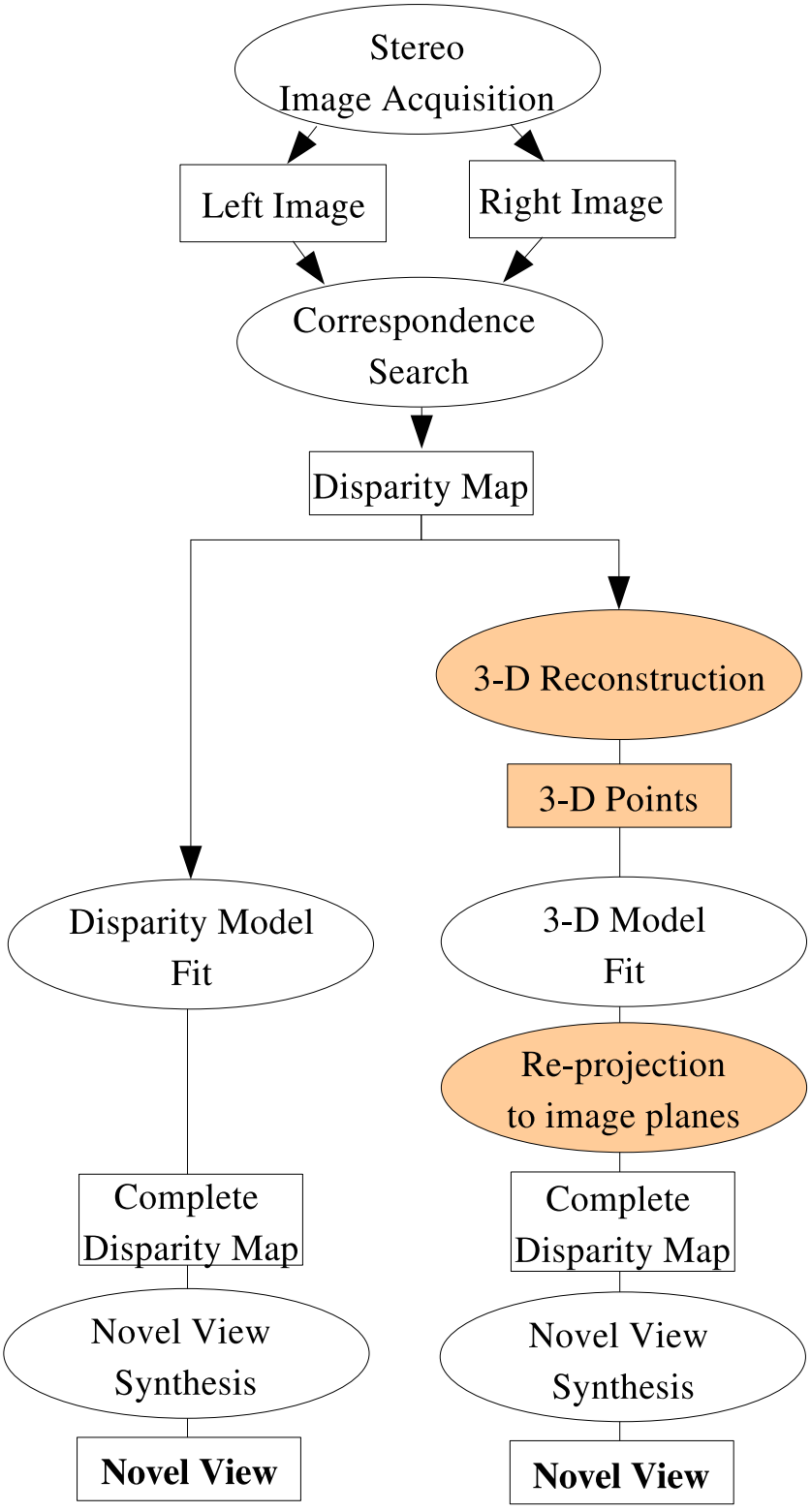


Figure 7.2: The diagram showing the comparison between fitting a model to disparity data and to reconstructed 3-D data. The equivalent steps in both approaches are shown on the same level. The colour-shaded parts of the diagram are the extra steps necessary to perform the fit to 3-D data.

Avoiding Error Propagation

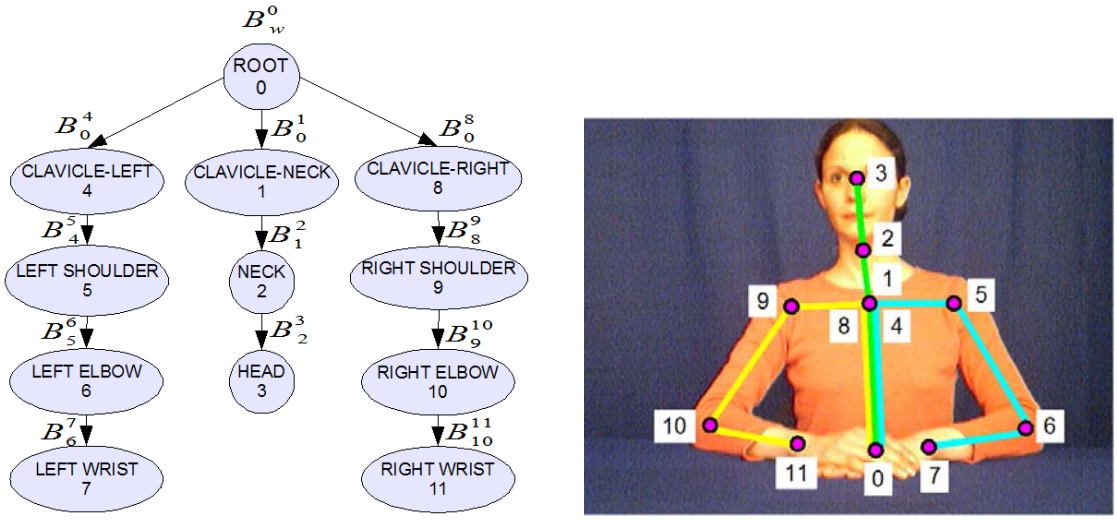
In order to reconstruct a 3-D position of a point from its stereo correspondences, an intersection of the corresponding optical rays must be found. Due to the image discretisation, the rays in space do not normally intersect, and the 3-D point is found as the midpoint of the shortest segment connecting the two rays [167]. Projecting the so-found reconstructed point back onto the image plane introduces yet another approximation into the computation. Both approximations are avoided by working in disparity space directly instead.

Higher Accuracy

The reconstructed 3-D points are affected by a point-dependent noise which is anisotropic and inhomogeneous [105]. In the statistical literature, this kind of noise is referred to as *heteroscedastic noise*. As a consequence, accurately estimating parametric models from reconstructed 3-D data does not allow for a closed-form solution, but instead requires an iterative treatment, *e.g.*, with an errors-in-variables model approach [105].

If the disparity values are restricted to the image points which have been stereo-matched with enough accuracy, *i.e.*, if the standard deviation of the noise in the estimated disparity values can be approximated with $\sigma_d = 1$ pixel, and the noise in x and y can be attributed to image discretisation with the standard deviation of the pixel detection error $\sigma_x = \sigma_y = 1$ pixel, the disparity space noise can then be fairly approximated by a covariance matrix $\Sigma = \sigma^2 I$, where $\sigma = \sigma_x = \sigma_y = \sigma_d$, and can be assumed to be isotropic and homogeneous, or, in statistical terms, *homoscedastic* [47].

Convergent stereo pairs need to be rectified first before our presented disparity space approach applies and the rectification process may introduce additional approximations as the warped image pixel positions must, in general, be computed via bilinear interpolation [61]. However, the noise in disparity space is still far more isotropic than the noise in 3-D space [47] and thus allows for more accurate parameter estimation. Several researchers have taken advantage of the homoscedastic nature of disparity data when estimating accurate rigid motion from stereo, *e.g.*, [8, 47, 49, 71].



(a) (b)

Figure 7.3: (a) Kinematic tree structure of the upper body model in disparity space and the corresponding transformations B_i^j. (b)Kinematic chains shown overlaid on the upper body.

7.3 Disparity Space Articulated Human Body Model

Like in the case of the 3-D upper-body model, the generic disparity space upper-body model also consists of two layers: the *skeleton* and the *skin*. The disparity space body model differs from the 3-D model in the fact that the mesh vertices composing the skin are now disparity points $D \in \mathbb{D}^3$ rather than 3-D points $X \in \mathbb{R}^3$, and the skeleton transformations are the similarity transformations B_i^j of the 3-D transformations A_i^j (see Section 5.2 for 3-D skeleton definition). We describe both layers in more detail next.

Skeleton

The skeleton of the disparity space model is defined as a set of disparity space transformation matrices which encode the position and orientation of every joint with respect to its parent joint in the hierarchy. The transformation matrices are arranged in a kinematic tree specifying the articulated motion.

Let pairs of indices (i, j) denote the parent-child pairs of nodes in the kinematic tree, $Q = \{(i, j)\}$ denote a set of all such pairs for a given kinematic tree and $N = |Q|$ denote the

cardinality of Q. We define the disparity space skeleton layer as a set of transformations:

$$Skeleton = \{B_w^0, B_{i_1}^{j_1}, B_{i_2}^{j_2}, \ \ldots \ , B_{i_N}^{j_N}\}, \ \ \forall (i,j) \in Q, \tag{7.14}$$

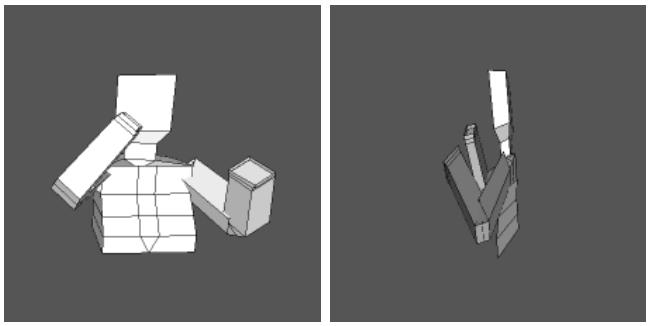
where B_i^j is a homogeneous transformation matrix encoding the position and orientation of the child joint coordinate system j with respect to the parent joint coordinate system i, defined as in Equation (7.13). The transformation B_w^0 denotes the transformation between the disparity space "world" coordinate system and position and orientation of the root joint coordinate system.

Figure 7.3(a) shows the structure of the kinematic tree for our disparity space upper body model. In Figure 7.3(b), the tree is shown as three simple open kinematic chains overlaid on top of the upper body. Note that the only difference between the kinematic tree specification in disparity space and its equivalent in 3-D space (Figure 5.3) are the disparity space transformation matrices B_i^j (Equation 7.13).

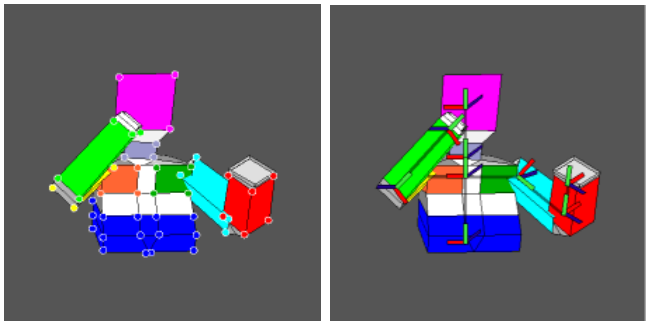
Skin

The skin layer represents the second layer of the generic disparity space upper-body model, connected to the skeleton through the joints' local coordinate systems. It is modelled as a subdivision surface in disparity space $\mathbb{D}^3$ with the base mesh shown in Figure 7.4(a). Note how the base mesh in disparity space looks a lot slimmer than its counterpart in 3-D space. This is due to the fact that the $x \in [0, 640)$ and $y \in [0, 480)$ dimensions refer to the image plane, while $d \in (-29, 116)$ is the disparity dimension with a much smaller range (the stated disparity interval is specific to the pose shown). Each of the joints controls the movement of a specific part of the base mesh by influencing the position and orientation of the base mesh vertices through its local coordinate system. Different colours in Figure 7.4(b) illustrate parts of the base mesh influenced by different joints. Figure 7.4(c) shows the approximate position of the underlying skeleton with local coordinate systems belonging to individual joints. The coordinate system x, y and d axes are colour coded as red, green and blue, respectively (*cf.* Figure 5.3).

Formally, the skin can be defined as a set of transformation matrices B_i^j from the skele-

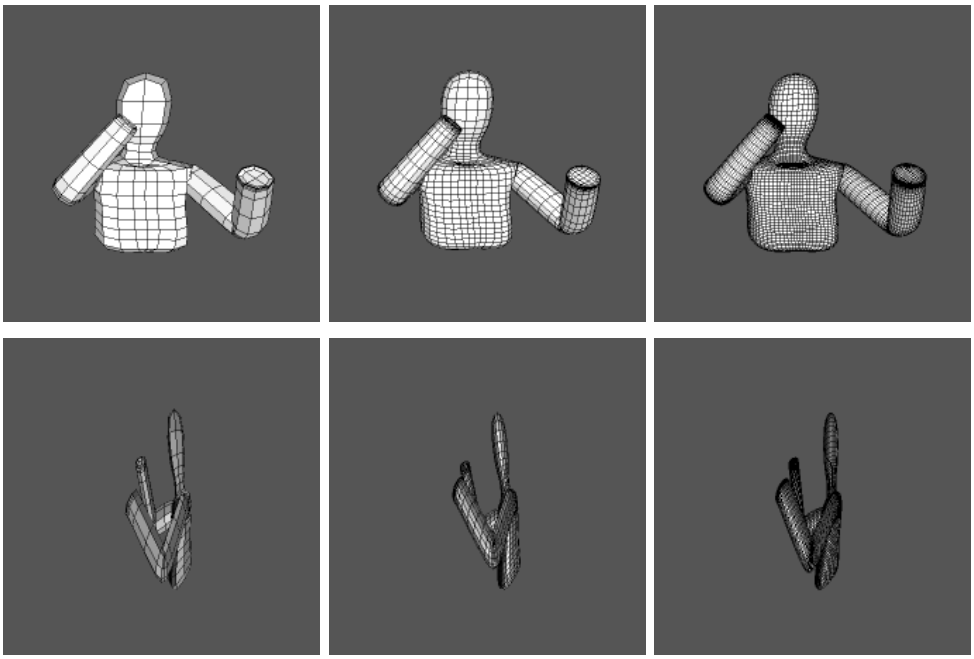


(a)



(b) (c)

Figure 7.4: The skin is modelled as a subdivision surface in disparity space. (a) The base mesh, front and side view. (b) Different colours denote parts of the base mesh influenced by different joints. (c) Approximate skeleton position for this particular pose and the corresponding local coordinate systems.



(a) (b) (c)

Figure 7.5: The skin subdivision levels with front and side view. (a) 1 level. (b) 2 levels. (c) 3 levels.

ton layer, as in Equation (7.14), combined with the corresponding sets of base mesh vertices $D_{B_i^j}$ directly influenced by each of these transformations:

$$Skin = \{(B_w^0, \{D_{B_w^0}\}), (B_{i_1}^{j_1}, \{D_{B_{i_1}^{j_1}}\}), (B_{i_2}^{j_2}, \{D_{B_{i_2}^{j_2}}\}), \ \ldots \ , (B_{i_N}^{j_N}, \{D_{B_{i_N}^{j_N}}\})\}, \tag{7.15}$$

where

$$D_{B_w^0} \cap D_{B_{i_1}^{j_1}} \cap D_{B_{i_2}^{j_2}} \cap \cdots \cap D_{B_{i_N}^{j_N}} = \emptyset. \tag{7.16}$$

The vertices $X \in \mathbb{R}^3$ of the subdivision surface base mesh in 3-D are transferred to $D \in \mathbb{D}^3$ in the disparity space using Equation (7.8). This transfer is only performed once to define the model in disparity space. All subsequent operations are performed on the vertices in disparity space directly.

In order to generate a smooth skin surface, all vertices must first be transformed from their corresponding local coordinate system, where they are defined, into a common global coordinate system, such as the world coordinate system. This is achieved by constructing the appropriate kinematic tree transformation for each group of vertices $D_{B_i^j}$. For example, the set of vertices $D_{B_{i_k}^{j_k}}$ is transformed into the world coordinate system by concatenating

all parent-child transformations which, according to the kinematic tree, lead to the joint j_k:

$$D^w_{B^{j_k}_{i_k}} = B^0_w \cdot B^{j_1}_0 \cdot \ \ldots \ \cdot B^{j_k}_{i_k} \cdot D_{B^{j_k}_{i_k}}. \tag{7.17}$$

The base mesh M_0, *e.g.*, the one shown in Figure 7.4 (a), then consists of all vertices V, expressed in a common coordinate system, connected with edges into quadrilateral faces F:

$$M_0 = \{V, F\}, \tag{7.18}$$

where

$$V = \{D^w_{B^j_i}\}, F = \{(m^w_a, m^w_b, m^w_c, m^w_d)\}, \ \text{ and } \ m^w_{\{a,b,c,d\}} \in V. \tag{7.19}$$

Finally, to obtain the smooth limit surface M_∞ of the base mesh, *i.e.*, the skin in Figure 7.5, the base mesh is repeatedly subdivided:

$$M_\infty = \lim_{k \to \infty} S^k M_0, \tag{7.20}$$

where S is the Catmull-Clark subdivision operator, described in Section 3.6, and S^k is its k-th application.

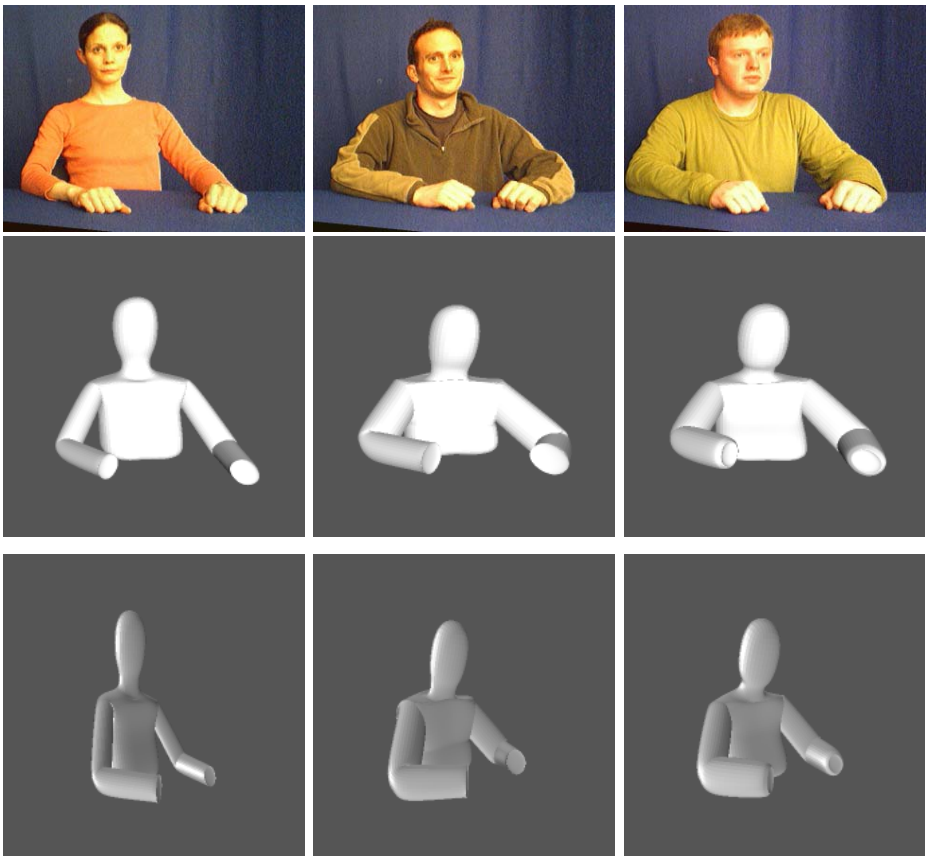
Figure 7.5 shows the front and side view of the first three levels of subdivision starting from the base mesh in Figure 7.4 (a).

Why is the 3-D model still necessary?

The reason for constructing the disparity space human body model via the 3-D model lies in the fact that we have an intuitive understanding of the model's shape and motion in 3-D. If we had the same intuitive understanding of the shape and motion in disparity space, the vertices and transformations could be expressed in disparity space directly, without relying on the 3-D representation to begin with. Transforming the model from 3-D into the disparity space is a one-off requirement: once the model has been specified, it is transformed into the disparity space and then used as if the 3-D model had never existed.

7.4 Pose Estimation in Disparity Space

The PSO pose estimation algorithm remains exactly the same as the one described in Chapter 5, Section 5.3. The difference between the disparity space approach and the 3-D space



(a) Subject S. (b) Subject D. (c) Subject G.

Figure 7.6: Front and side view of the generic upper body disparity space models for different test subjects.

approach lies in the way particle fitness evaluation is carried out (Algorithm 5.3 in Section 5.3.5) and we describe this procedure next.

For each particle in the swarm, a disparity space subdivision model representing the pose is generated. This is done by calculating the appropriate transformations B_i^j and applying them to the generic model base mesh in disparity space. The base mesh is then subdivided to produce the smooth limit surface representing the model in a particular pose.

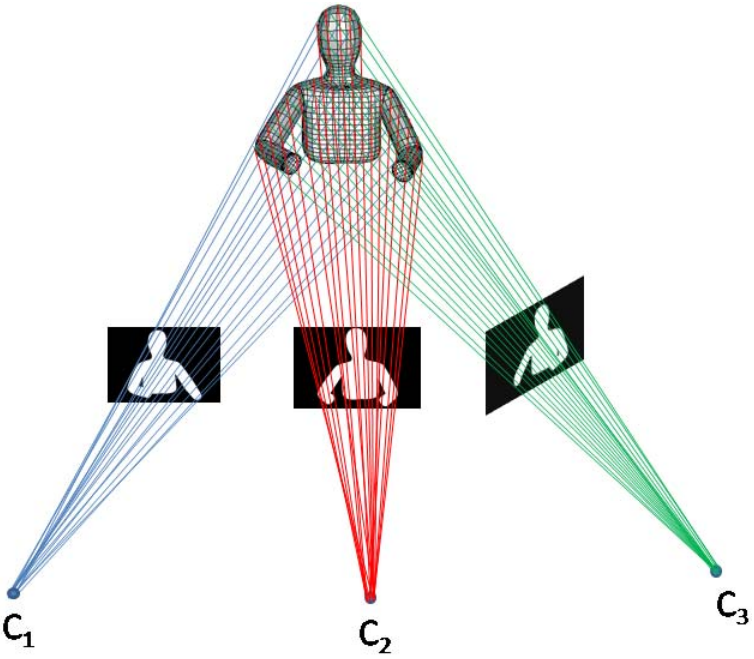
In order to evaluate the fitness of individual particles, model silhouettes must first be generated for every particle in the swarm. When generating silhouettes for a 3-D model, the model is projected onto a corresponding image plane using perspective projection (see Figure 7.7(a)).

In disparity space, the silhouette generation process is slightly different. As the disparity space is defined with respect to a rectified stereo pair of cameras, whereas the pose can be estimated from more than only two views, the procedure for generating model silhouettes differs depending on the view.

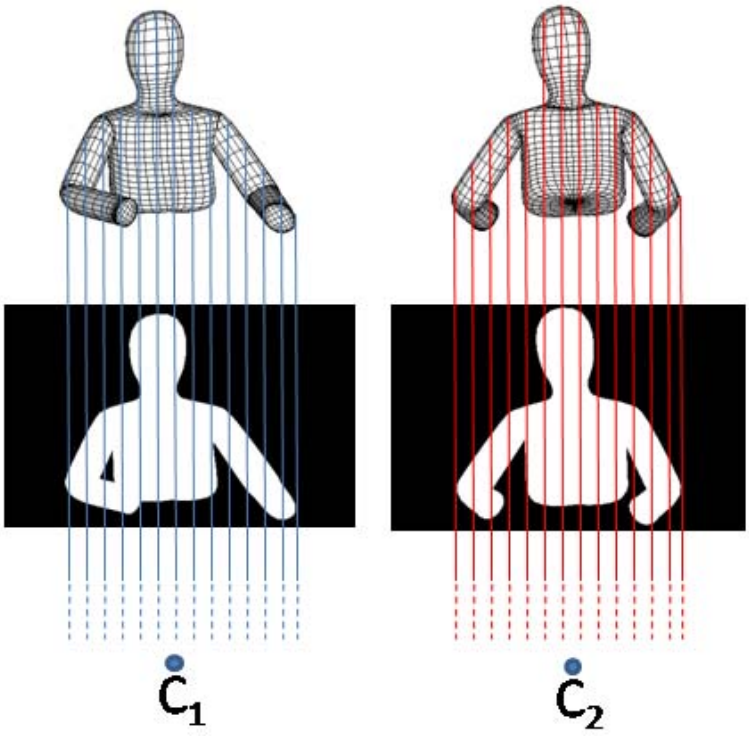
Let us refer to the stereo pair of cameras which define the disparity space as the *reference stereo pair*. Let one of the reference cameras be known as *camera* 1 and the other as *camera* 2, so that the disparity values are calculated with respect to camera 1.

The disparity model silhouette for every view in the setup is then generated as follows:

- Camera 1 view: the disparity space model vertices are orthographically projected onto the reference image plane (see Figure 7.7(b)).
- Camera 2 view: the vertices of the disparity space model are changed from (x,y,d) to $(x+d,y,-d)$ which swaps the roles of cameras 1 and 2. The model is then orthographically projected onto the camera 2 image plane (see Figure 7.7(b)).
- Non-reference views: the model vertices are transferred to any image plane outside the reference stereo pair by using trilinear tensor transfer (see Section 3.9.2) on pairs



(a) 3-D model silhouettes are generated using perspective projection.



(b) Disparity space model silhouettes for the reference stereo pair are generated using orthographic projection (the centre of projection is infinitely far away).

Figure 7.7: Silhouette generation in 3-D space and disparity space.

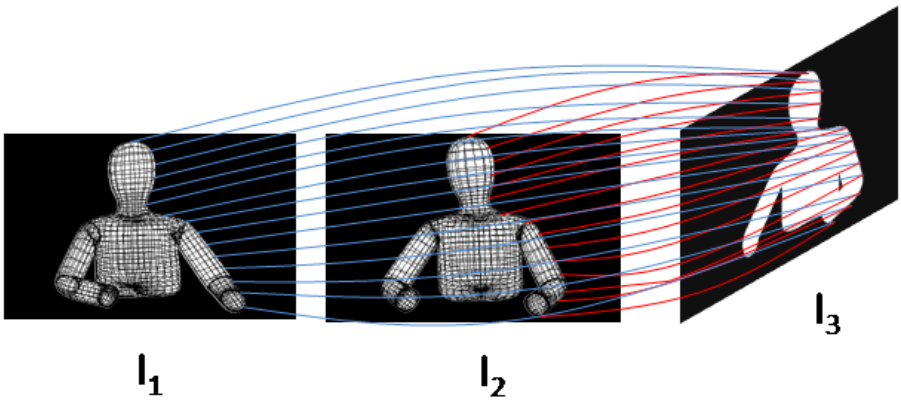


Figure 7.8: Non-reference view silhouettes for the disparity space model are generated with trilinear tensor transfer from the reference stereo pair.

$(x,y),(x+d,y)$, as shown in Figure 7.8.

Examples of pose estimates using the disparity space model are shown in Figures 7.9, 7.10 and 7.11 for three different test subjects. In all three figures, the left view of the reference stereo pair used to define the disparity space is shown in the leftmost column. As opposed to the 3-D model which is not view-dependent, in disparity space everything is defined with respect to the reference stereo pair. Notice how the disparity space model for different poses imitates the pose shown in the left stereo view.

7.5 Model Fitting in Disparity Space

Once the pose of the model has been estimated, the model can then be fit to disparity data directly, without 3-D reconstruction. The model fitting algorithm described in Chapter 6, Section 6.2.3, is used in disparity space as well, with the subdivision surface model now being the disparity space model and the 3-D data points the disparity data points.

The process of establishing correspondences changes similarly to the process of generating silhouettes. Instead of perspective projection, orthographic projection is used to project the disparity data from the reference stereo pair onto the image plane. Additional disparity data originating from non-reference stereo pairs can be transfered into the disparity space defined with respect to the reference stereo pair via trilinear tensor transfer.

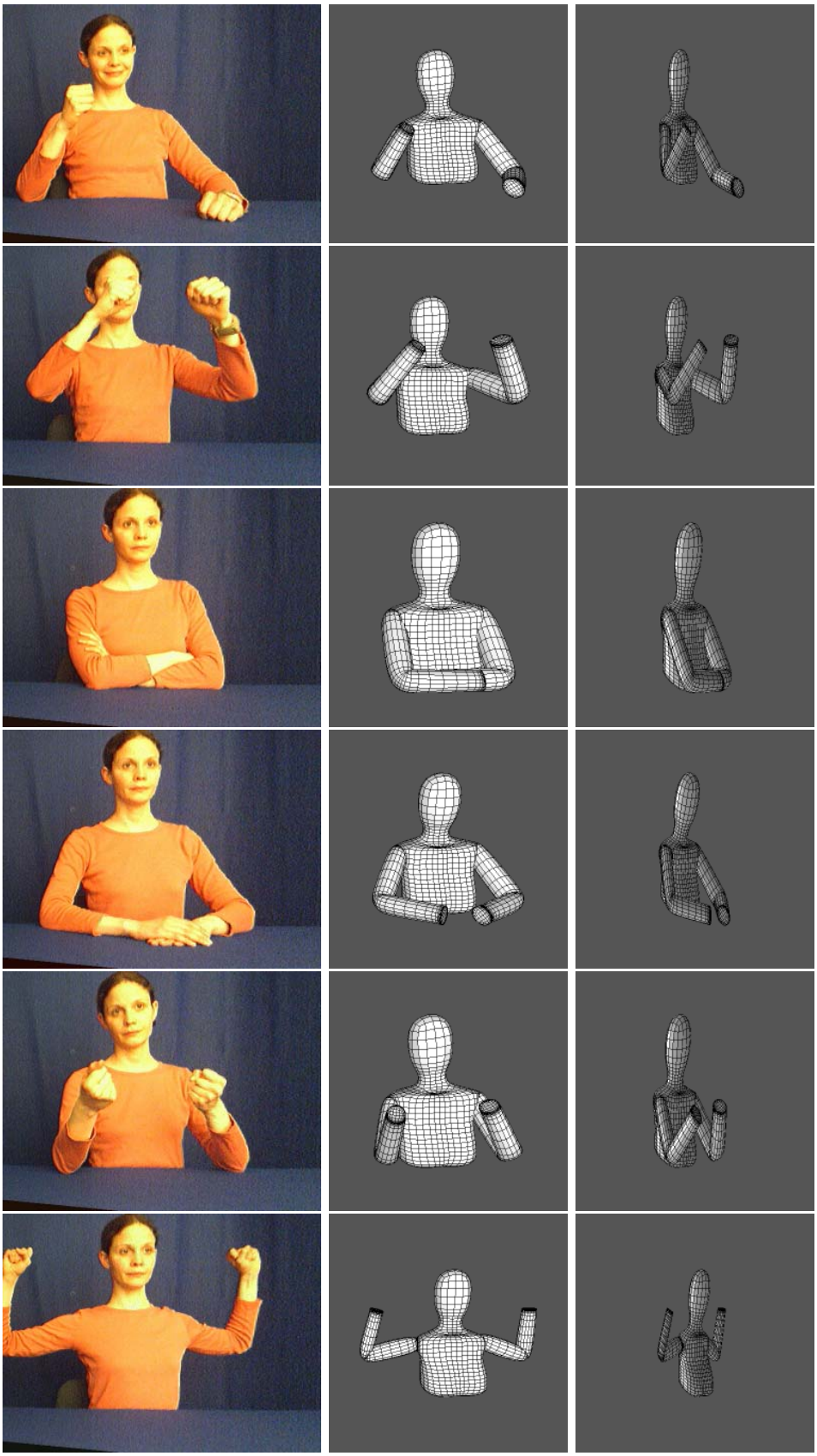


Figure 7.9: Different pose estimates in disparity space for subject S.

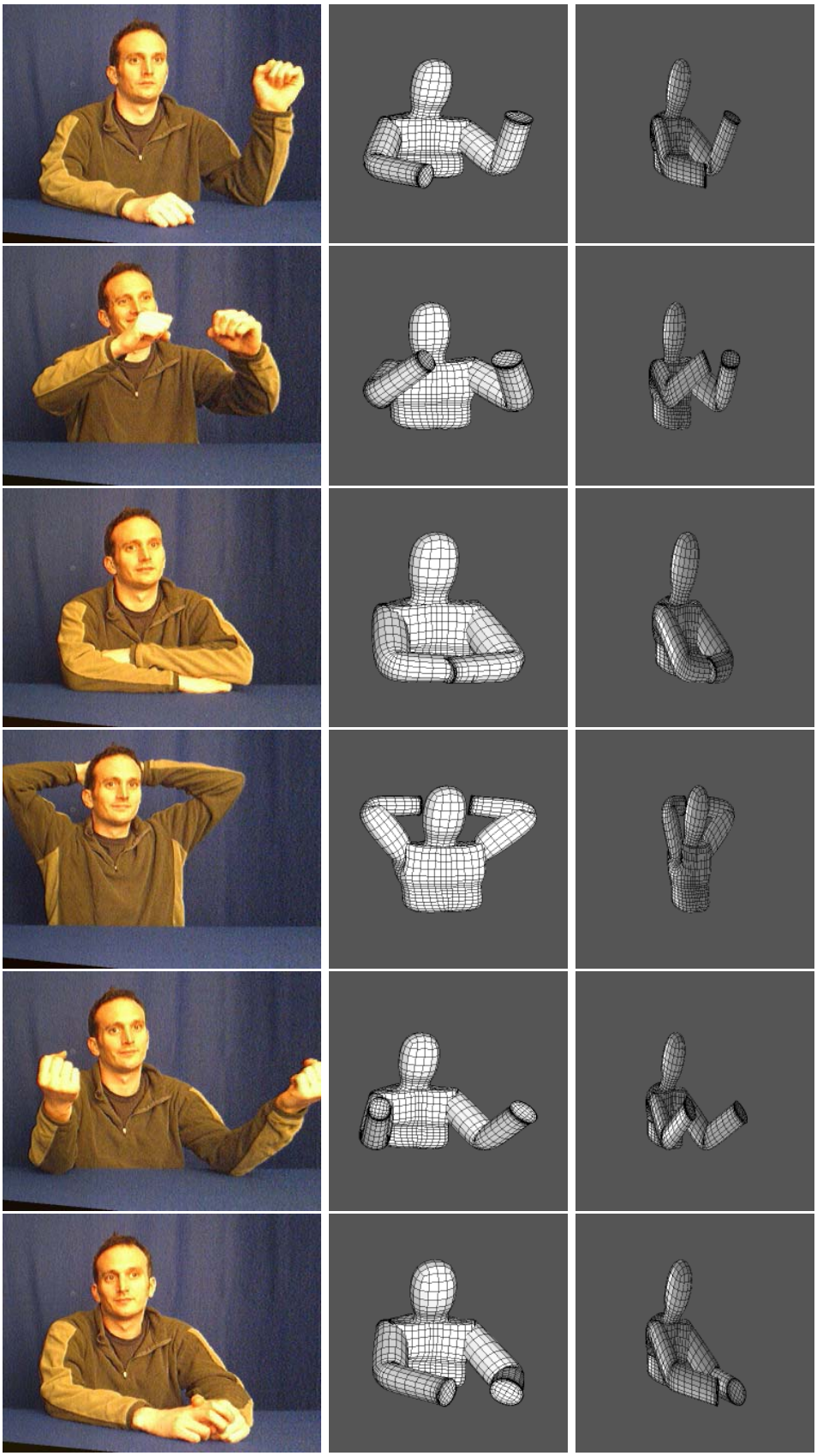


Figure 7.10: Different pose estimates in disparity space for subject D.

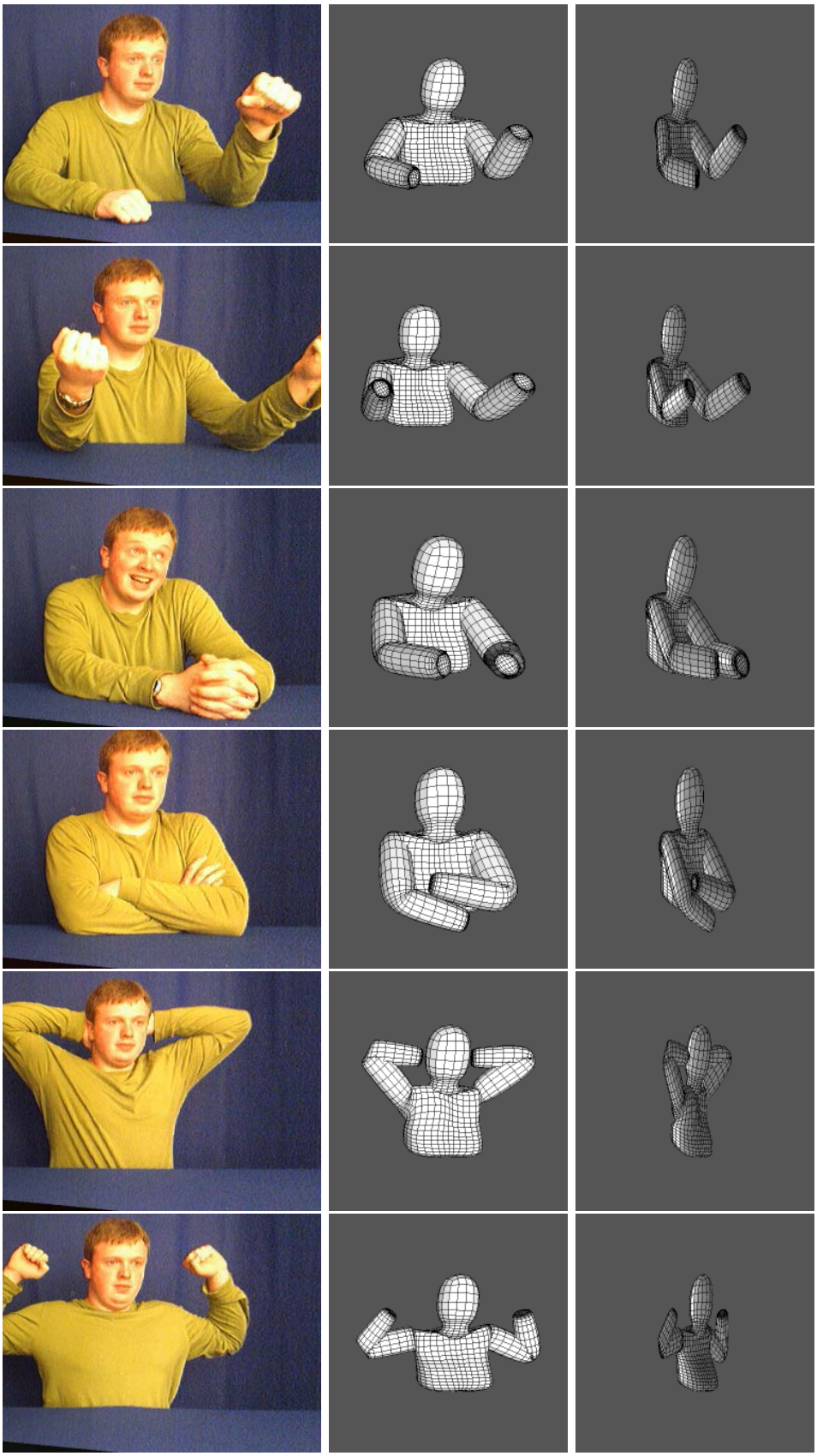
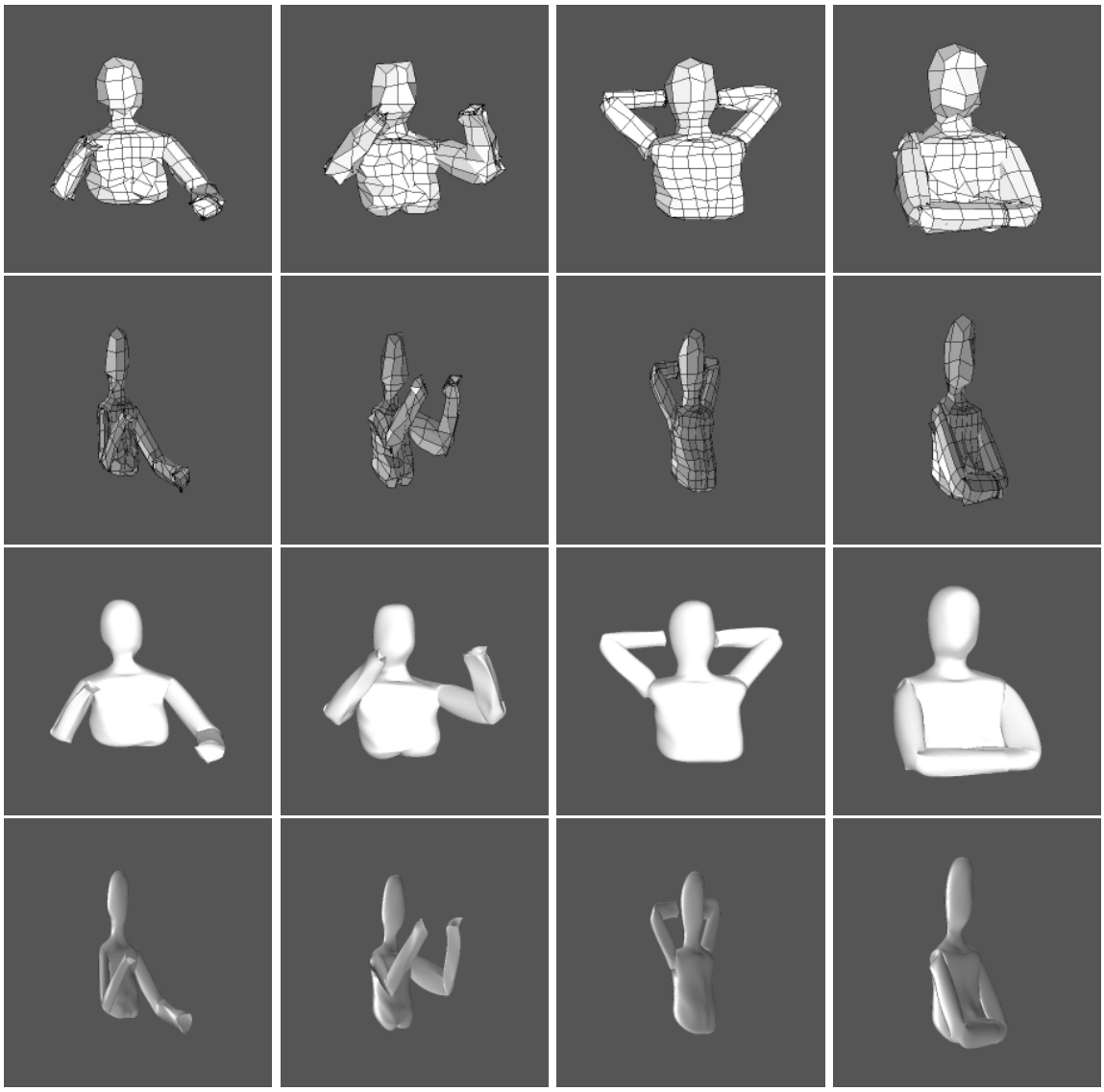


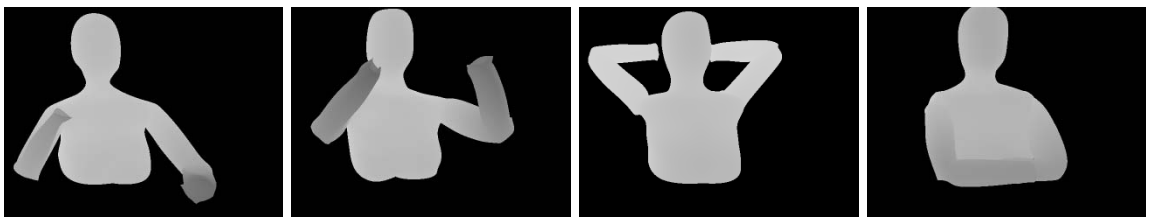
Figure 7.11: Different pose estimates in disparity space for subject G.

Like in the 3-D case, we show examples of disparity space models after the fit to disparity data. Figures 7.12, 7.13 and 7.14 illustrate the results of the model fit in disparity space for the three different test subjects and also show the disparity maps generated from the deformed models, later used for novel-view synthesis.

(a) Left view of the reference stereo pair, shown for different poses.

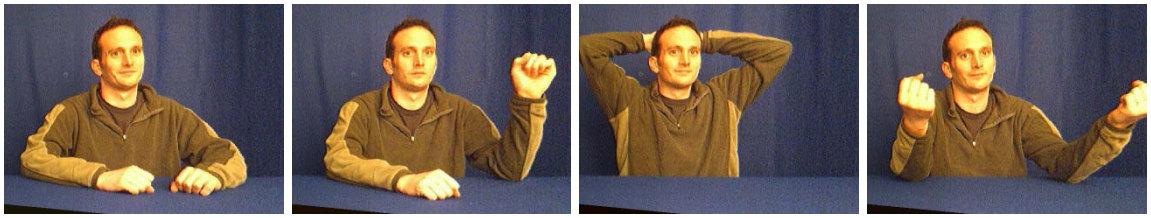


(b) Disparity space model base mesh and subdivided surface after fit level 1, front and side view.

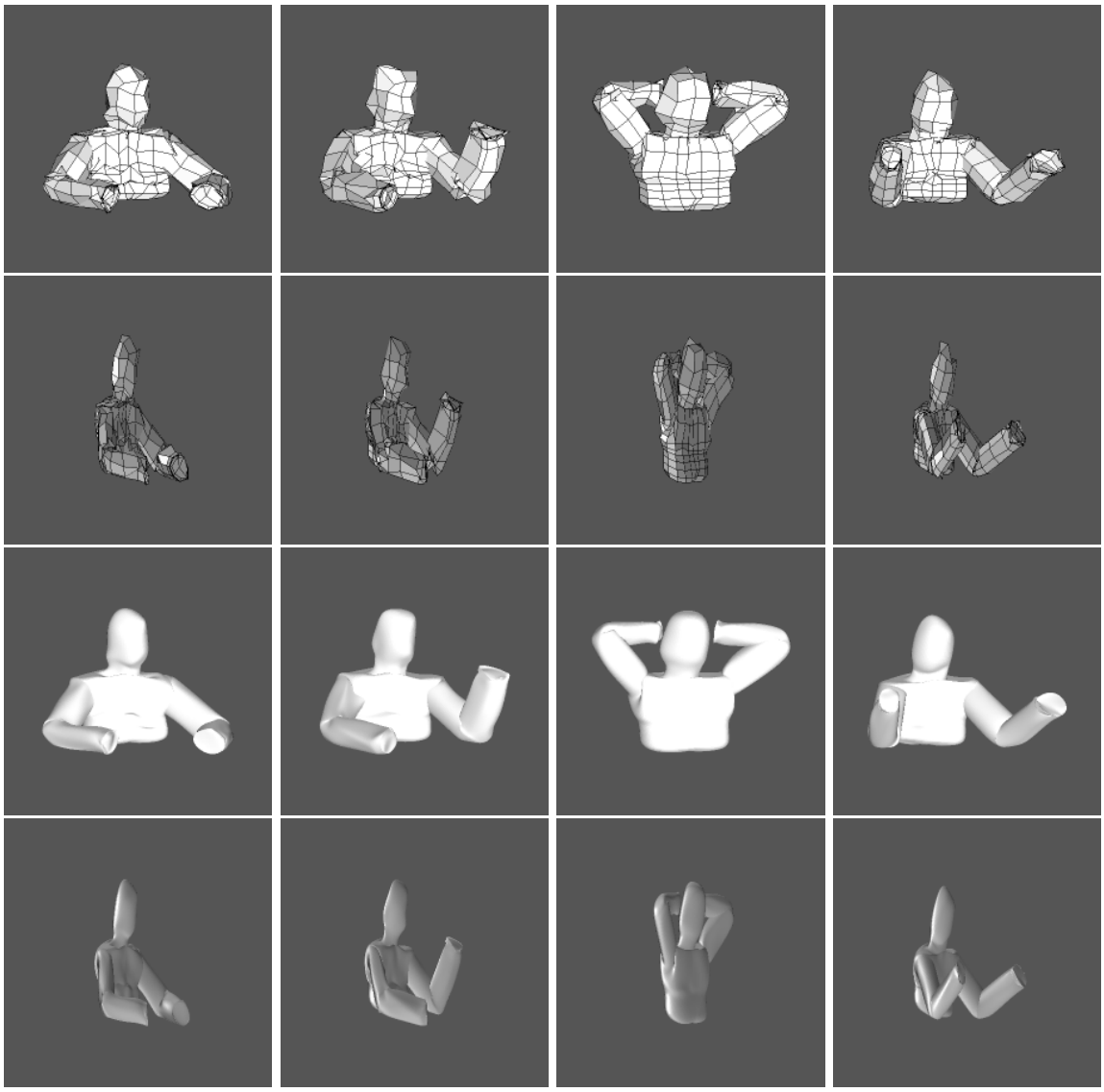


(c) Disparity maps generated from deformed disparity models.

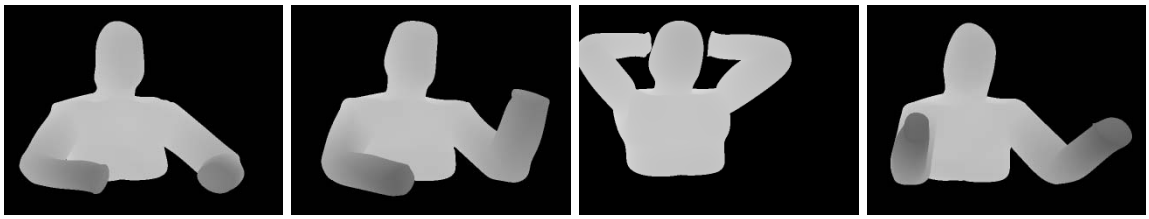
Figure 7.12: Disparity models for different poses of subject S after fit to disparity data.



(a) Left view of the reference stereo pair, shown for different poses.



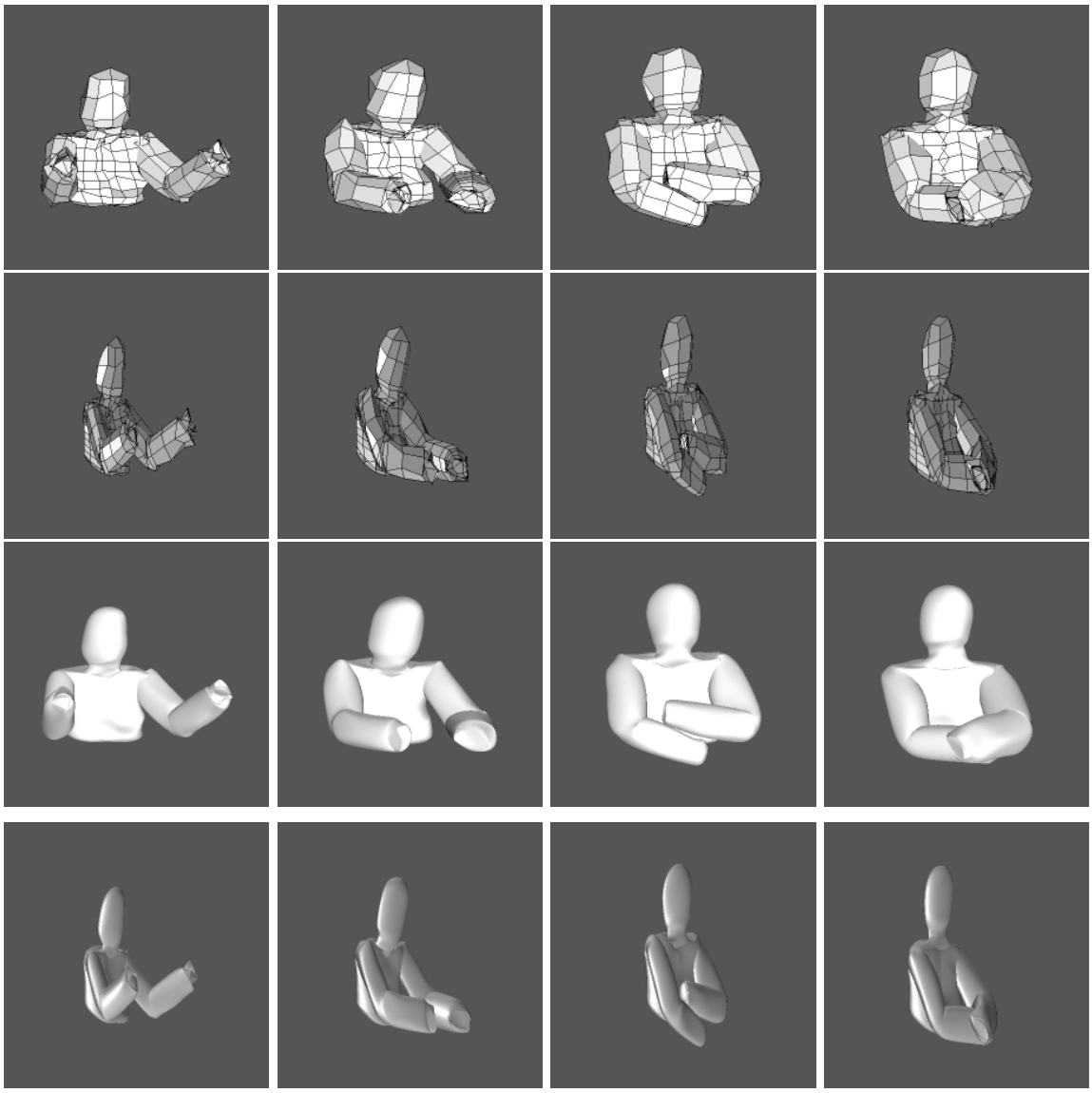
(b) Disparity space model base mesh and subdivided surface after fit level 1, front and side view.



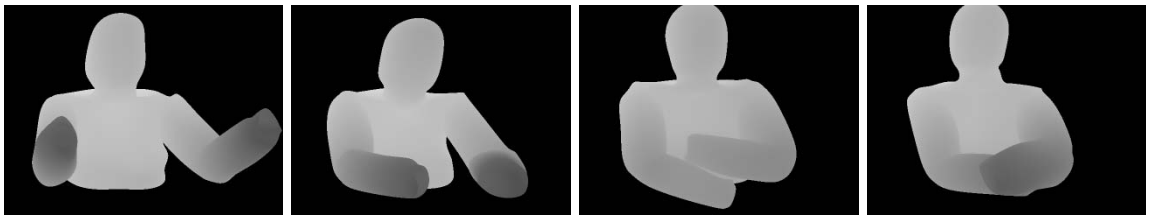
(c) Disparity maps generated from deformed disparity models.

Figure 7.13: Disparity models for different poses of subject D after fit to disparity data.

(a) Left view of the reference stereo pair, shown for different poses.



(b) Disparity space model base mesh and subdivided surface after fit level 1, front and side view.



(c) Disparity maps generated from deformed disparity models.

Figure 7.14: Disparity models for different poses of subject G after fit to disparity data.

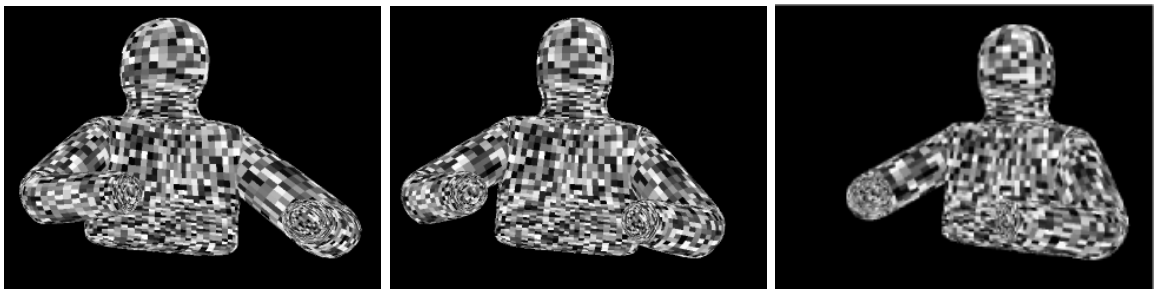


Figure 7.15: The three synthetic views generated for the comparison experiment.

7.6 Comparison of $\mathbb{D}^3$ and $\mathbb{R}^3$

We performed an experiment on a synthetic stereo pair of images to compare the performance of the post-processing algorithm in 3-D space with the one in disparity space. Every care was taken to ensure that the conditions were kept the same for both approaches and a fair comparison was possible. The model-data correspondence information was determined for the 3-D model and then used for both 3-D and disparity space deformation to guarantee consistency. Figure 7.16 shows the diagram of the experiment.

As shown in Figure 7.16, three views were acquired with an OpenGL synthetic camera setup (see Figure 7.15). The left and centre view were used to obtain disparity data and the right view was used as ground truth for the view synthesis.

Figure 7.17 illustrates the comparison process. The upper row refers to the 3-D space algorithm and the lower row to the disparity space algorithm. Column (a) shows the *a priori* 3-D model and disparity model. Column (b) shows the reconstructed 3-D points using triangulation and disparity data as three-dimensional data in disparity space. The 3-D and disparity model after fitting to corresponding data are shown in column (c). Column (d) shows the resulting disparity maps and column (e) the synthesised novel view. Column (f) contains the difference images between the synthesised views and the ground truth right view.

Table 7.1 lists the sum of absolute differences (SAD) measure for each of the synthesised views compared to the ground truth right view. The ground truth disparity map is

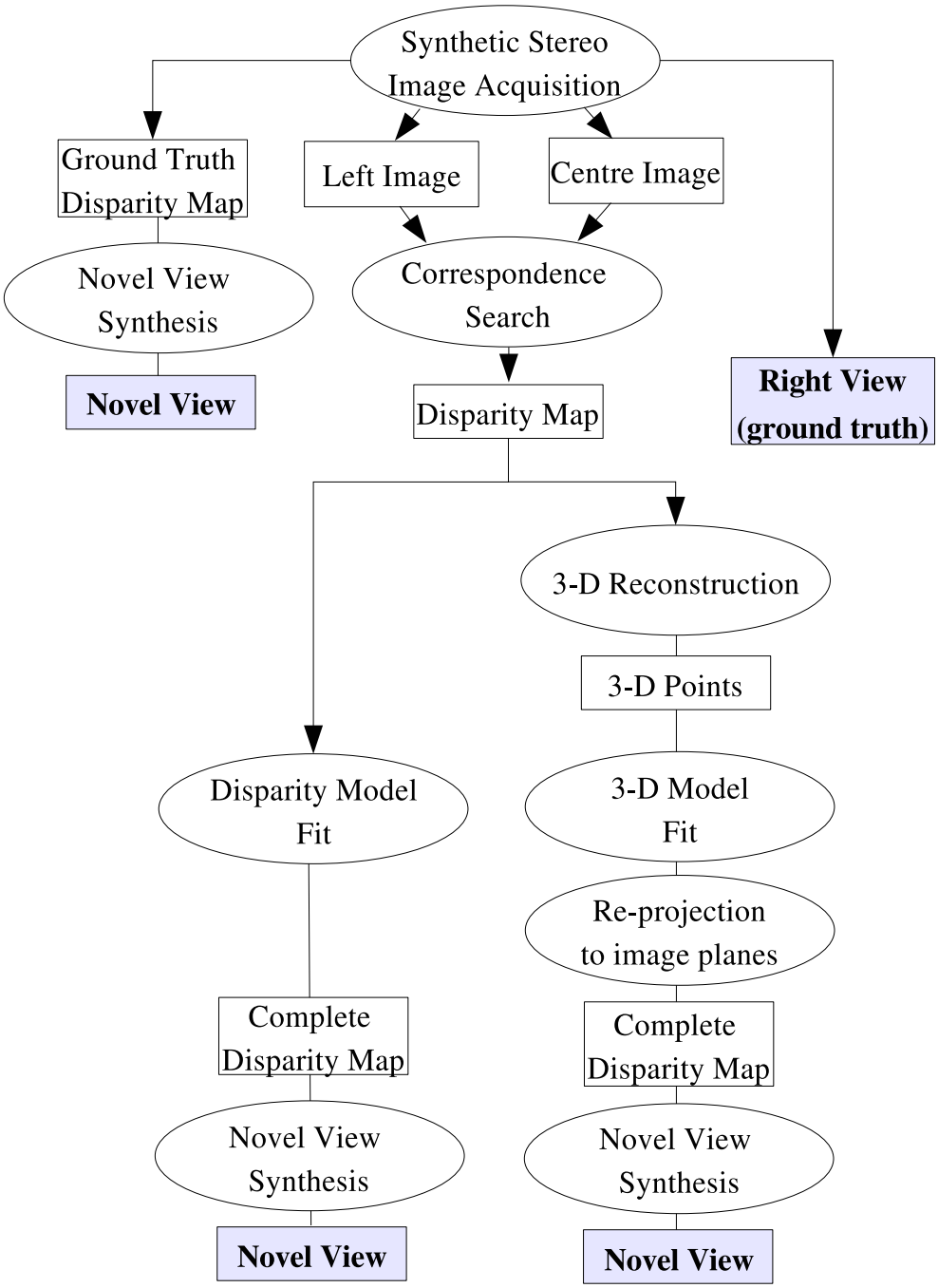
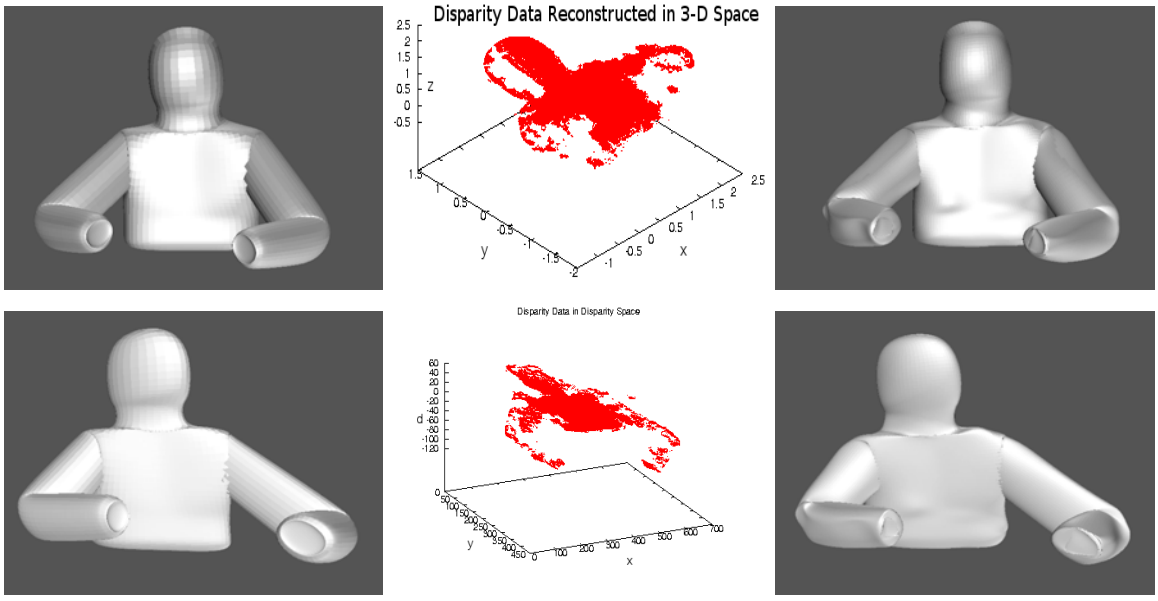
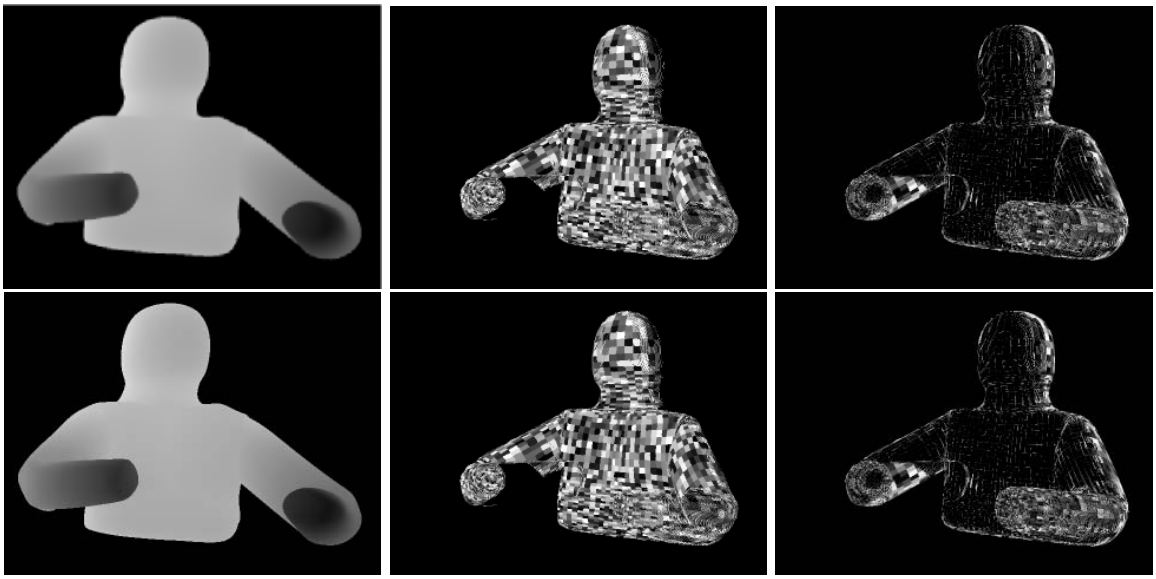


Figure 7.16: The flowchart of the experiment where we compared the post-processing in 3-D space with post-processing in disparity space. The shaded boxes are the views which were used for comparison.



(a) 3-D model and disparity model. (b) 3-D data and disparity data. (c) Deformed model.



(d) Disparity map. (e) Synthesised novel view. (f) Difference image.

Figure 7.17: Illustration of the comparison steps on synthetic data. In every column, the upper image refers to the 3-D algorithm and the lower image to the disparity space algorithm.

	Ground Truth View
Ground Truth DM	2659991
Disparity Space	3915826
3-D Space	3924308

Table 7.1: Pixel-by-pixel sum of absolute differences (SAD) comparison between the ground truth view (the right view) and the synthesised novel views. The synthesised views were computed using the ground truth disparity map, the disparity space post-processed disparity map and the 3-D space post-processed disparity map.

the closest in terms of view synthesis quality, which is to be expected because the disparity map generation via the correspondence search, which is used as an input to the other two methods, necessarily introduces some noise in the data. Of the remaining two, the disparity space algorithm slightly outperforms the 3-D space approach. This difference is most likely due to the fact that an unavoidable approximation error was introduced when reconstructing the 3-D data in order to perform the fit in 3-D space, and then projecting the resulting model back onto the image plane to compute the disparity map.

As demonstrated, disparity space algorithm not only saves on computational complexity, as illustrated in Diagram 7.2, but also has a performance which is at least equivalent to the 3-D approach. By the latter we mean that, on top of avoiding the 3-D reconstruction error when fitting to disparity data directly, there is also a potential for the disparity space pose estimation approach to be more accurate than the 3-D space approach, if care is taken to preserve the data's homoscedastic nature [105]. We do not take advantage of this in our algorithm, because we do not estimate the pose parameters directly from disparity data, as it proves too sparse to serve as a reliable constraint (we use the silhouettes instead). However, should the data be available and used in this way, the pose parameter estimation in disparity space would be preferable.

7.7 Conclusion

In this chapter we presented the concept of disparity space, an alternative to 3-D space. We showed that the articulated body pose estimation as well as model deformation can be carried out in disparity space directly, without resorting to 3-D reconstruction methods.

One claim that we make in this chapter is that the disparity space approach can be more accurate as well as less computationally demanding compared to the equivalent approach in 3-D space. This claim requires further clarification which we attempt to give here.

One advantage of disparity space is the homoscedastic nature of the disparity data which allows for parameter estimation which is provably more accurate than the one in 3-D space. This relates to our research on deforming the body model to better fit the data, where fitting to disparity data directly is advantageous over fitting to 3-D data.

The other advantage of working in disparity space is the fact that the step of 3-D reconstruction can be avoided, therefore saving some computational effort. One could argue that this is in fact not the case as the savings achieved by not reconstructing the data in 3-D are offset by the fact that the transformation matrices in disparity space must be pre- and post-multiplied with the additional similarity transformation. While this is indeed the case, as shown in Equation (7.13), the similarity transformation is fixed and can be incorporated into individual kinematic chain transformation matrices in advance, as required.

Finally, as already pointed out earlier, in our pose estimation algorithm we do not use the data points themselves to constrain the search and as a result the 3-D space pose estimation is really not all that different from the pose estimation in disparity space. The main difference is in the nature of projection, as the 3-D space requires a perspective projection, while the disparity space uses the orthographic projection which is somewhat simpler. Should one decide to include the data as a constraint in the pose estimation cost function, the homoscedastic property of the disparity data would make the disparity space pose estimation a better choice than the equivalent algorithm in 3-D.

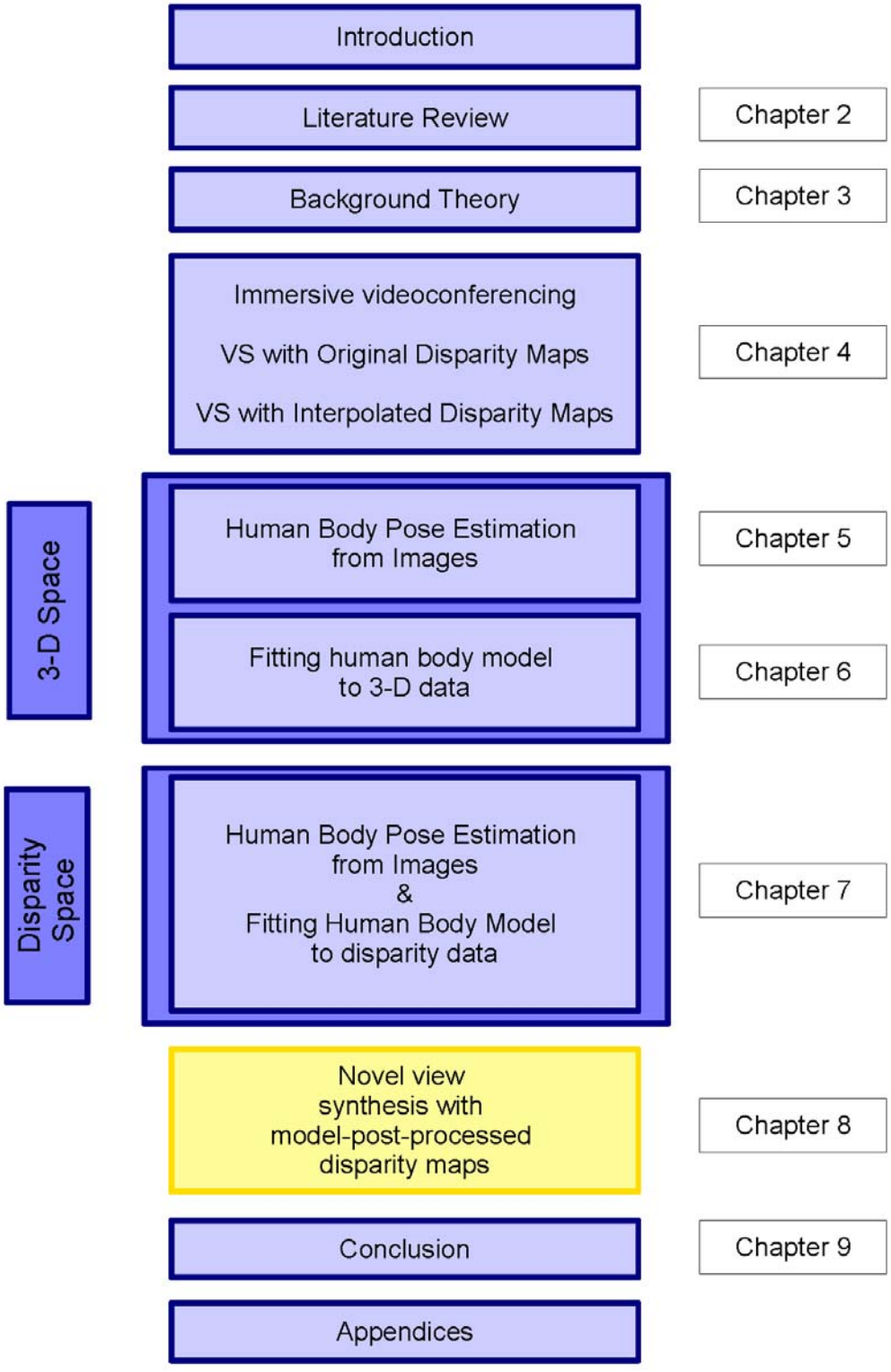


Figure 7.18: The thesis structure diagram.

Chapter 8

View Synthesis with Model-Completed Disparity Maps

8.1 Introduction

In this thesis, we addressed the problem of incomplete wide-baseline disparity data used in image-based novel-view synthesis. We looked at different ways of completing the disparity data in order to achieve better novel view synthesis.

We first completed the disparity map using three different interpolation algorithms, as described in Chapter 4. Although the quality of view synthesis improved with respect to using the original, incomplete disparity maps, the improvement was limited and highly dependent on the initial quality of the original disparity maps (the more data, the better the interpolation).

We then decided to exploit the fact that the scene was fairly constrained and could thus be modelled with a generic model, thereby introducing additional knowledge into the disparity data completion process. We developed model pose estimation and model fitting algorithms which allowed us to use the generic 3-D model of a human body for data completion. We also addressed the question of modelling the scene in disparity space, where the data originates, to save on computational complexity and improve the accuracy of the completion process.

In this chapter, we demonstrate the improvement that a model-completed disparity map brings to the quality of the novel view synthesis.

8.2 Which Model?

In this thesis, the term model-completed disparity map refers to a disparity map that has been completed using a generic model of the scene. Within the scope of the thesis, this can be either the 3-D model or the disparity space model.

In the results of the comparison experiment performed on synthetic data in Chapter 7, Section 7.6, the disparity space model approach outperformed the 3-D model approach. We therefore demonstrate the improvement in the view synthesis results using disparity maps completed with the disparity space algorithm.

8.3 Is Model Deformation Necessary?

The shape of the generic model does not necessarily capture the actual body shape of the imaged person very accurately. We use the available disparity data to deform the model so that it better resembles the real person. As the data is patchy and noisy, we deform the model to such extent that it only approximates the data and not interpolates it. The shape resemblance with the real person is therefore limited.

We investigated the quality of the view synthesis in connection with model deformation. We compared the average RGB distance of the synthesised view, using the deformed and undeformed model, with the ground truth view across different body poses and for three different synthetic cameras, camera 1, camera 3 and camera 4 (see diagram in Figure 4.4). Equation (8.1) shows the distance measure used, where R and C is the number of image rows and columns, respectively, s denotes the synthesised view and GT denotes the ground truth view, to which the synthesised view was compared. r, g and b are the RGB colour components of the individual image pixels.

$$\frac{1}{RC}\sum_{i=1}^{RC}\sqrt{(r_i^s - r_i^{GT})^2 + (g_i^s - g_i^{GT})^2 + (b_i^s - b_i^{GT})^2} \tag{8.1}$$

Figures 8.1, 8.2 and 8.3 show the graphs for all three subjects and all three synthetic cameras. As expected, there is an obvious improvement from using only the original disparity maps to using the disparity maps which were completed with a generic model. The deformed body models improve the result further, however only slightly. This is in line with our expectations, given that the model deformation was deliberately very limited to avoid modelling the noise.

The numerical values which were used to plot the graphs in Figures 8.1, 8.2 and 8.3, are given in the Appendix D, where Tables D.1, D.2 and D.3 contain RGB difference values for subject S in cameras 1, 3 and 4, respectively, and in similar fashion Tables D.4, D.5 and D.6 and Tables D.7, D.8 and D.9 detail the values for subjects D and G.

In the light of the comparison results shown in Figures 8.1, 8.2 and 8.3, we argue the following.

Firstly, given the quality of the data that we used, the difference in view synthesis quality achieved when deforming the model is not big enough to justify its computational requirements and can in this case be avoided. More time can instead be spent on making the generic model dimensions before deformation match the test subject dimensions more accurately.

Secondly, although the view synthesis quality improvement with deformed models is very small, it does exist. This shows that there is potential for further improvement if more data becomes available. This is not an unrealistic expectation given state-of-the-art stereo correspondence algorithms [134] which should be able to outperform our simple pyramidal correspondence search algorithm. The knowledge of the generic model in correct body pose also works to our advantage as it can be used as a disparity search interval constraint in the correspondence search, increasing the chances of finding the correct disparity values.

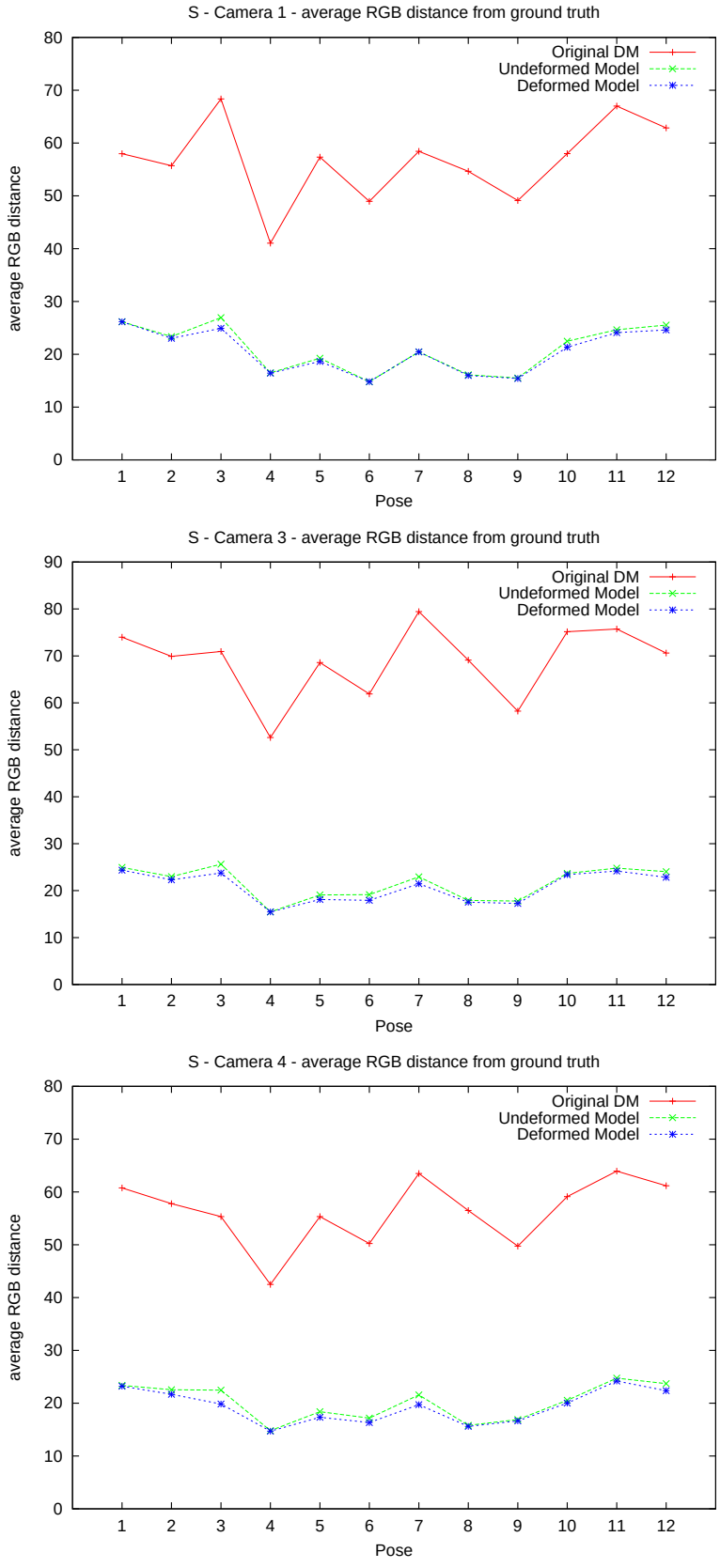


Figure 8.1: The average RGB distance per pose for Subject S in synthetic cameras 1, 3 and 4, measured with respect to the ground truth image.

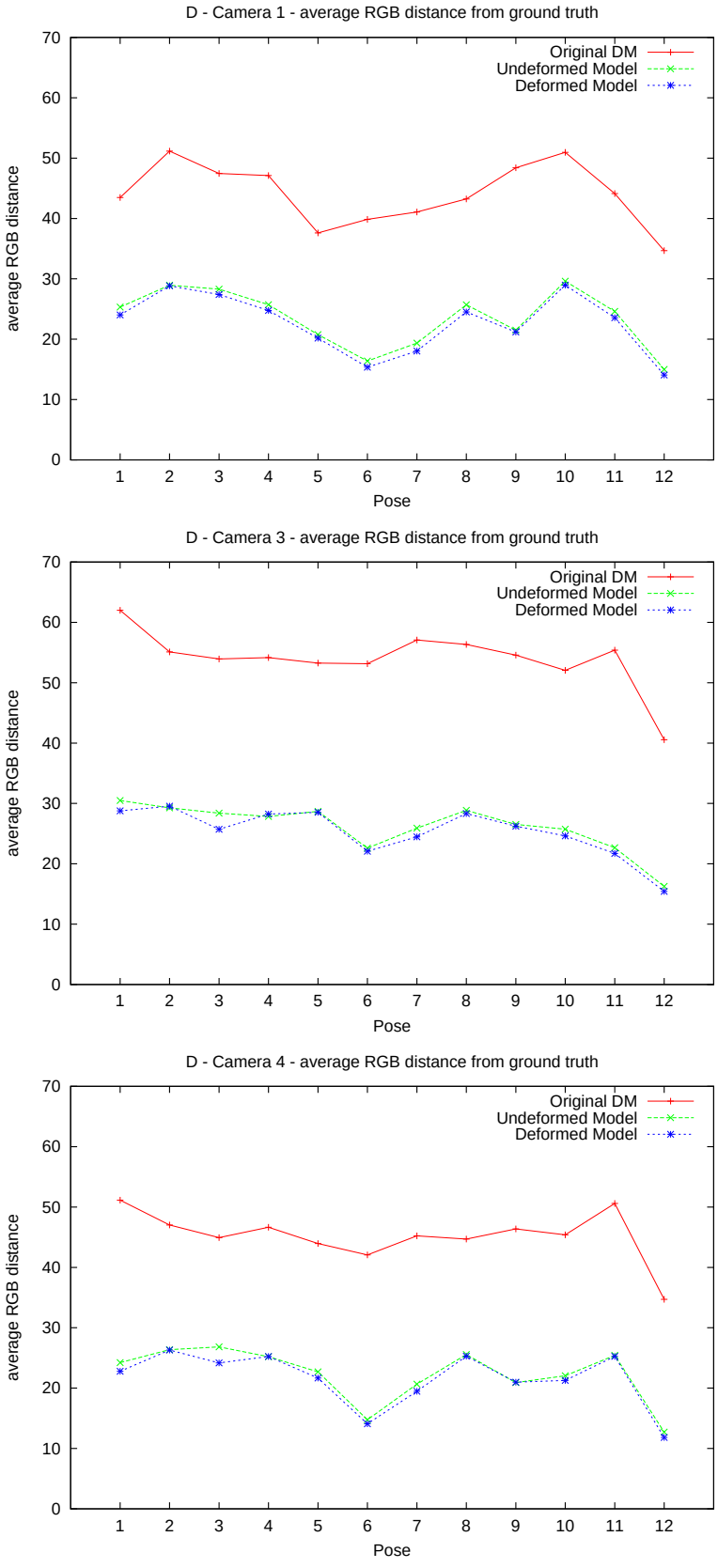


Figure 8.2: The average RGB distance per pose for Subject D in synthetic cameras 1, 3 and 4, measured with respect to the ground truth image.

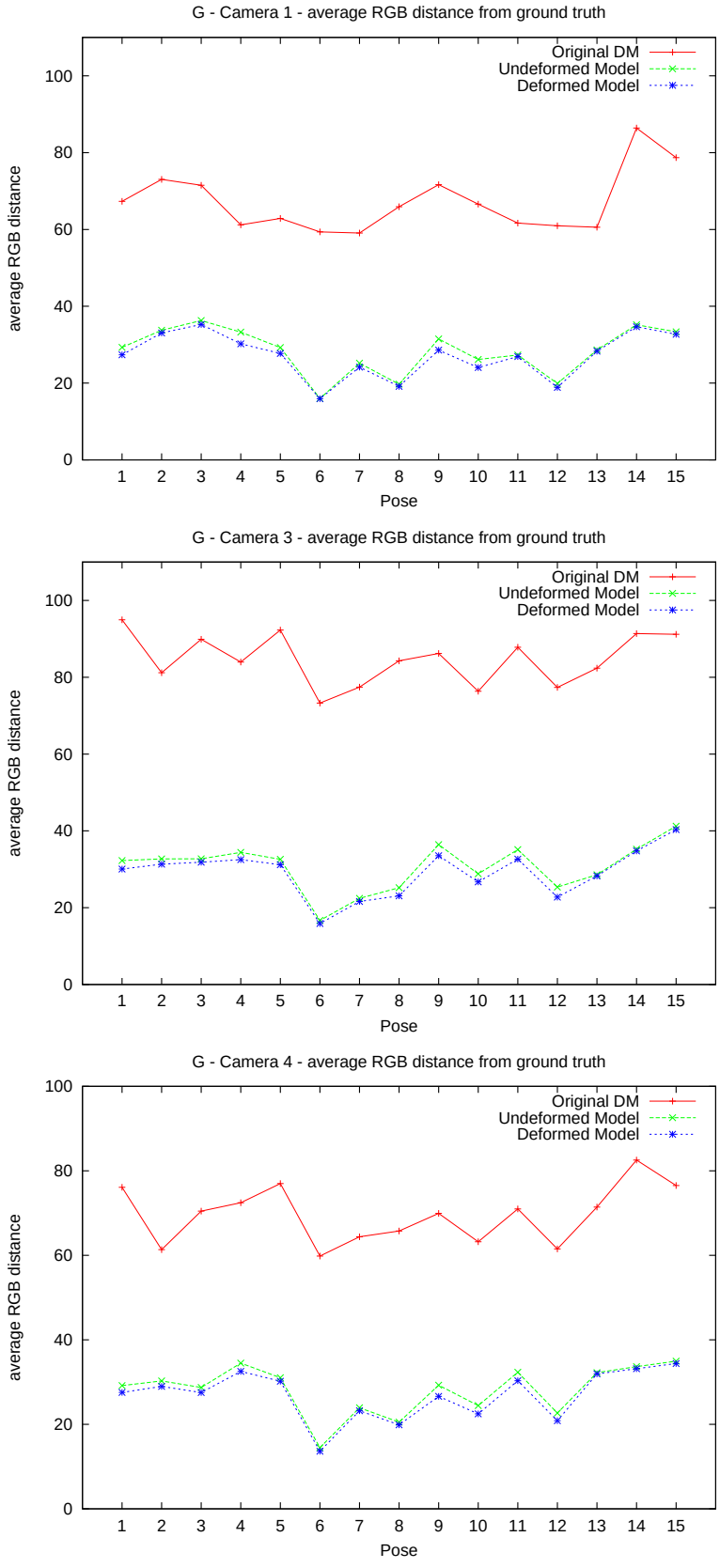


Figure 8.3: The average RGB distance per pose for Subject G in synthetic cameras 1, 3 and 4, measured with respect to the ground truth image.

8.4 View Synthesis Results

In this section, we show the novel-view synthesis results for the three different test subjects used throughout the thesis. Figure 8.6 shows novel views for subject S. For clarity, the first novel view is shown magnified in Figure 8.7. Figure 8.8 shows novel views for subject D, with the first view shown magnified in Figure 8.9. Lastly, Figure 8.10 shows novel views for subject G and the magnified view in Figure 8.11. All novel-view synthesis results are shown with the ground truth image shown in the first row, the novel view synthesised using the original disparity map in the second row and the novel view synthesised using the deformed disparity model in the third row.

Compared to the view synthesis results using the original disparity map, the improvement obtained by using the model is clearly visible from the synthesised images. However, several artefacts also appear in the views synthesised using our model-based disparity completion method and we discuss these next.

The simple generic model does not match the shape of the real person entirely. We intended to compensate for this limitation by deforming the model to fit the available disparity data. Due to the patchy and noisy nature of the data, this could only be achieved to a limited extent and the resulting body shape is still only an approximation. This is most obvious in the face area, where a lack of generic model features, such as nose and ears, often causes the synthesised face to be unrealistically deformed.

Another important limitation is the lack of hand models. In the pose estimation step, described in Chapter 5, we found that modelling the hands as simple blobs rather than articulated objects misled the optimisation and better results were achieved by not modelling the hands at all. While this was advantageous in the case of pose estimation, it clearly becomes a limitation when synthesising novel views. Articulated hand modelling and pose estimation is a research area on its own and is as such outside the scope of this work, however, it represents an important problem which should be addressed in the future in order to improve the view synthesis results.

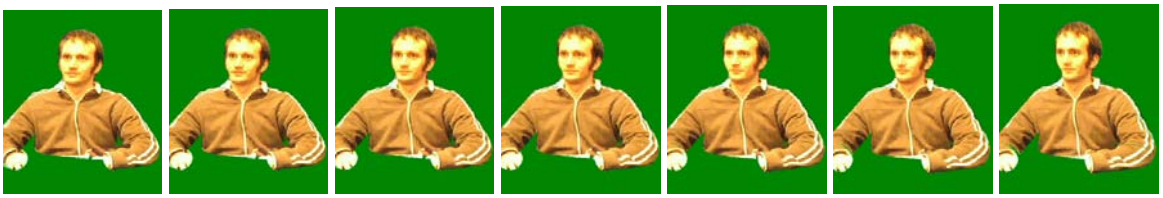


Figure 8.4: View synthesis with an extrapolated virtual view. The synthesised results look convincing, however, the effects of the model-based approach, such as the unrealistic shape of the head, also become obvious when extrapolating.

Last but not least, the view synthesis suffers from artefacts due to the differing colour nuances in the stereo pair of views, which become obvious when the two views are combined together to form the synthetic view (see Figure 8.5). This is further exaggerated by the conservative foreground-background segmentation which was necessary because of poor lighting conditions in which the images were acquired. These artefacts can be removed by using a higher quality camera and lighting setup as well as by normalising the colour content of the images to lie in the same part of the colour spectrum.

In spite of the limitations mentioned above, the resulting synthetic views supported by the prior knowledge of the scene geometry allow for novel views to be synthesised relatively far away from the original stereo pair. Figure 8.4 shows an example view synthesis where the synthetic camera has been extrapolated with the view changing from camera 2 to camera 4. The synthetic views look fairly convincing while also demonstrating that a more realistically shaped model is required for them to be entirely believable when the viewpoint is very far away from the original stereo pair.

Finally, Figure 8.5 shows the comparison between the novel views synthesised using the disparity map completed with linear, bilinear, cubic interpolation and the disparity model. It demonstrates the advantage of using *a priori* knowledge to complete the disparity map rather than interpolating the noisy and patchy disparity values from the original disparity map. It also illustrates the disadvantage of the model-based approach. The virtual view obtained by using the model does not contain the spiky artefacts and outliers present in the novel views synthesised using the interpolated disparity maps. While the model-based

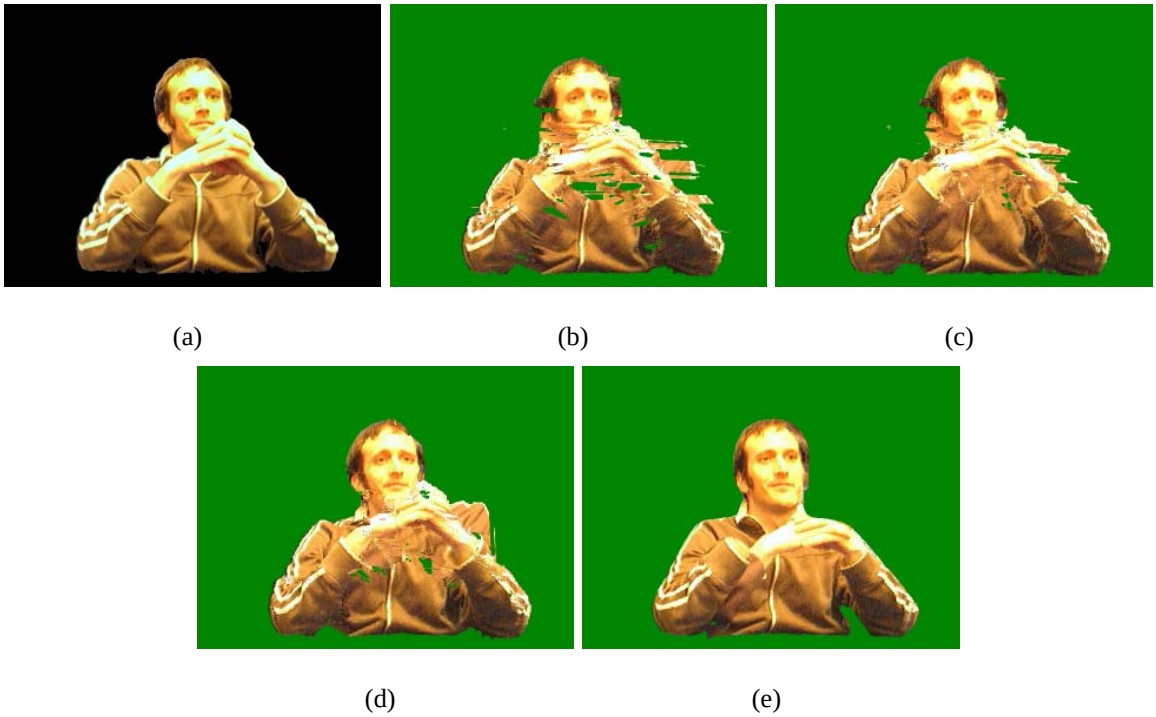
virtual view is much cleaner, it does not handle the hands correctly as they are not a part of the generic model. As a consequence, when using the model-completed disparity map to synthesise novel views, the image regions belonging to the hands move as a part of the torso, hence the strange detached hand artefact in Figure 8.5(e). A better model which included the hands would solve this problem.

The difference in texture colour below the subject's right upper arm in Figure 8.5(e) is a result of the way the view synthesis algorithm fills in the missing texture information. The texture under the right upper arm was not available in the first view that was used for trilinear tensor transfer view synthesis and had to be sourced from the second view, where a different camera colour response and loosely controlled lighting conditions made the texture look slightly different. This artefact is common to all virtual views shown in Figure 8.5, interpolated and model-based, but it is more obvious in the model-based virtual view because that view contains fewer other artefacts. More carefully controlled acquisition conditions as well as normalising the camera colour response would help alleviate this artefact.

8.5 Conclusion

In this chapter we presented the novel-view synthesis results obtained with our disparity space human body model completion method. The results were obtained by first manually setting the dimensions of the generic body model to match those of the imaged person. Then, the pose of the imaged person was estimated, as described in Chapters 5 and 7. The resulting generic body model in the correct body pose was deformed to better describe the disparity data, as shown in Chapters 6 and 7, and finally, the complete disparity maps were generated and used to synthesise novel views.

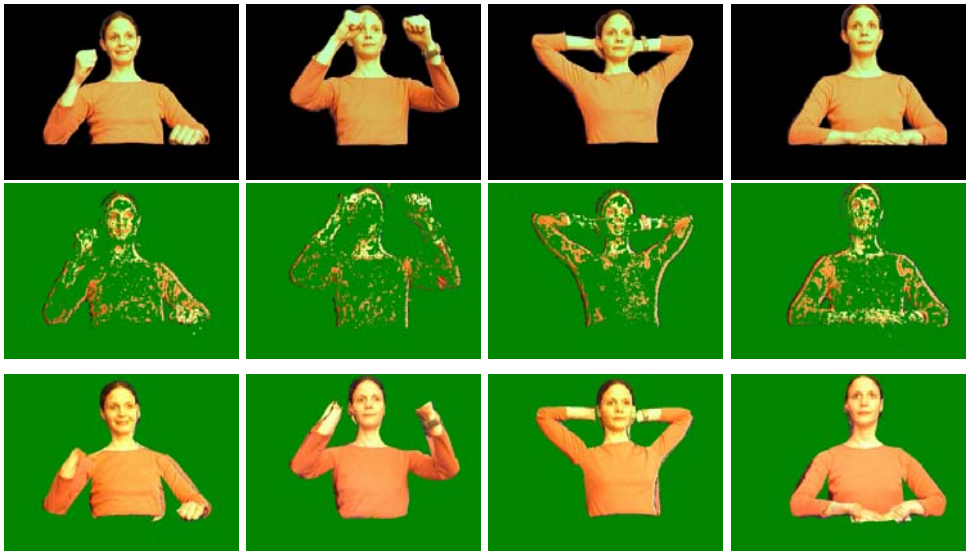
The results shown in this chapter demonstrate the improvement achieved with respect to the original disparity maps, visually and quantitatively. In spite of the clearly improved view synthesis results, there is still scope for further research which we discuss in the final chapter.



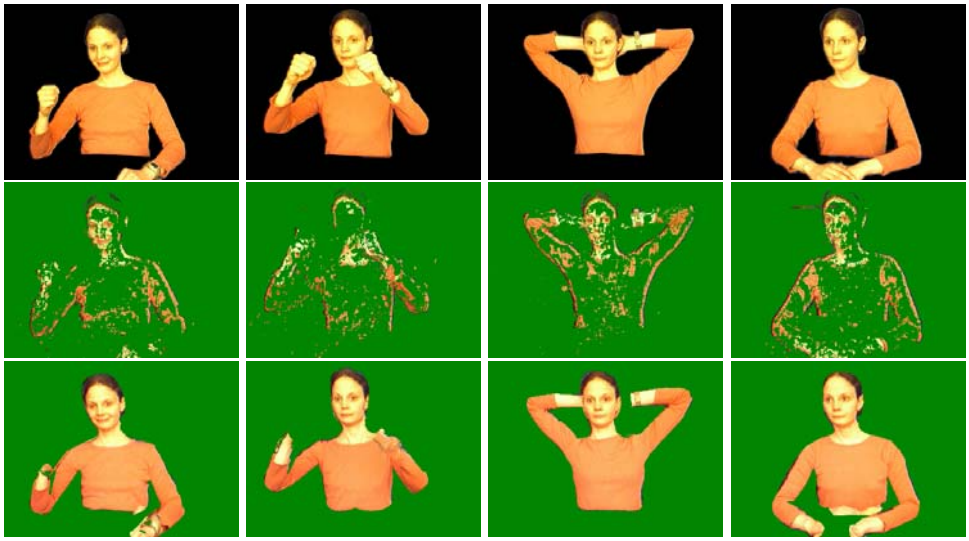
(a) (b) (c)

(d) (e)

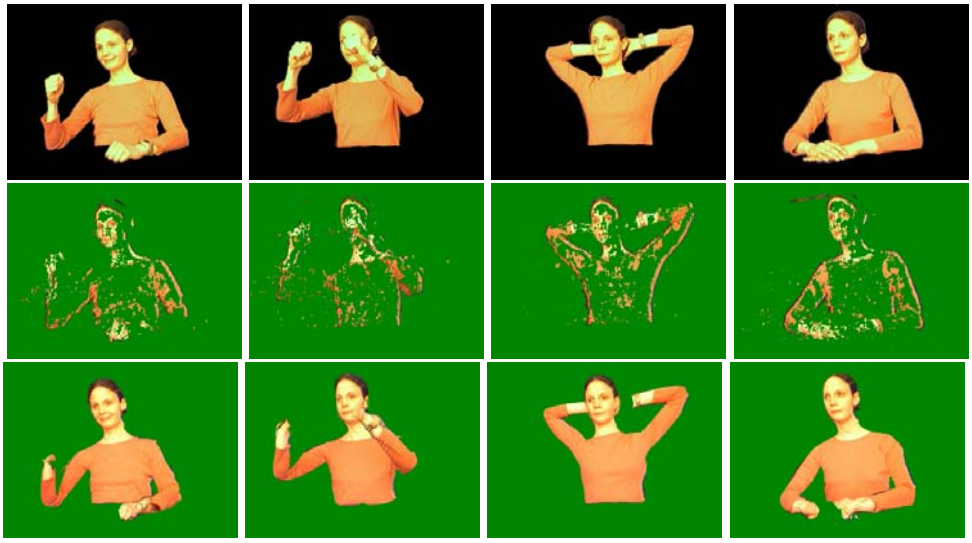
Figure 8.5: Comparison of view synthesis results for the ground truth test view (a) and disparity maps completed using (b) linear interpolation, (c) bilinear interpolation, (d) spline interpolation and (e) disparity space model. The novel views synthesised with interpolated disparity maps exhibit unrealistic artefacts due to the self occlusion created by placing the arms in front of the torso. The model-based approach can cope with this kind of self-occlusions much better, however the lack of hand models becomes a limitation in this case as the hands are assigned the disparity values of the torso and move accordingly, creating the detached hand artefact in view (e).



(a) Camera 1



(b) Camera 3



(c) Camera 4

Figure 8.6: Synthesised views for subject S.

(a) Virtual view for camera 1.

(b) Virtual view for camera 3.

(c) Virtual view for camera 4.

Figure 8.7: Synthesised views for subject S, magnified.

(a) camera 1

(b) camera 3

(c) camera 4

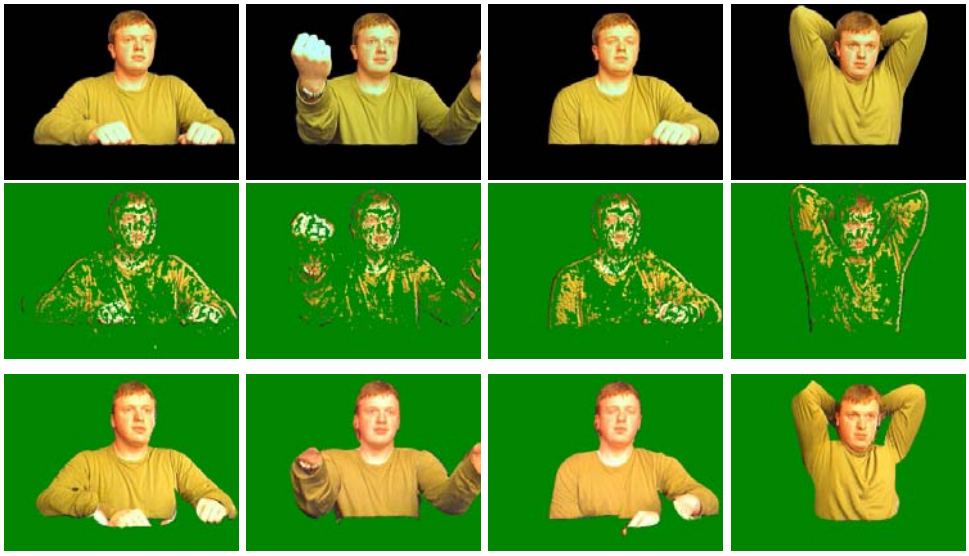
Figure 8.8: Synthesised views for subject D.

(a) Virtual view for camera 1.

(b) Virtual view for camera 3.

(c) Virtual view for camera 4.

Figure 8.9: Synthesised views for subject D, magnified.



(a) camera 1

(b) camera 3

(c) camera 4

Figure 8.10: Synthesised views for subject G.

(a) Virtual view for camera 1.

(b) Virtual view for camera 3.

(c) Virtual view for camera 4.

Figure 8.11: Synthesised views for subject G, magnified.

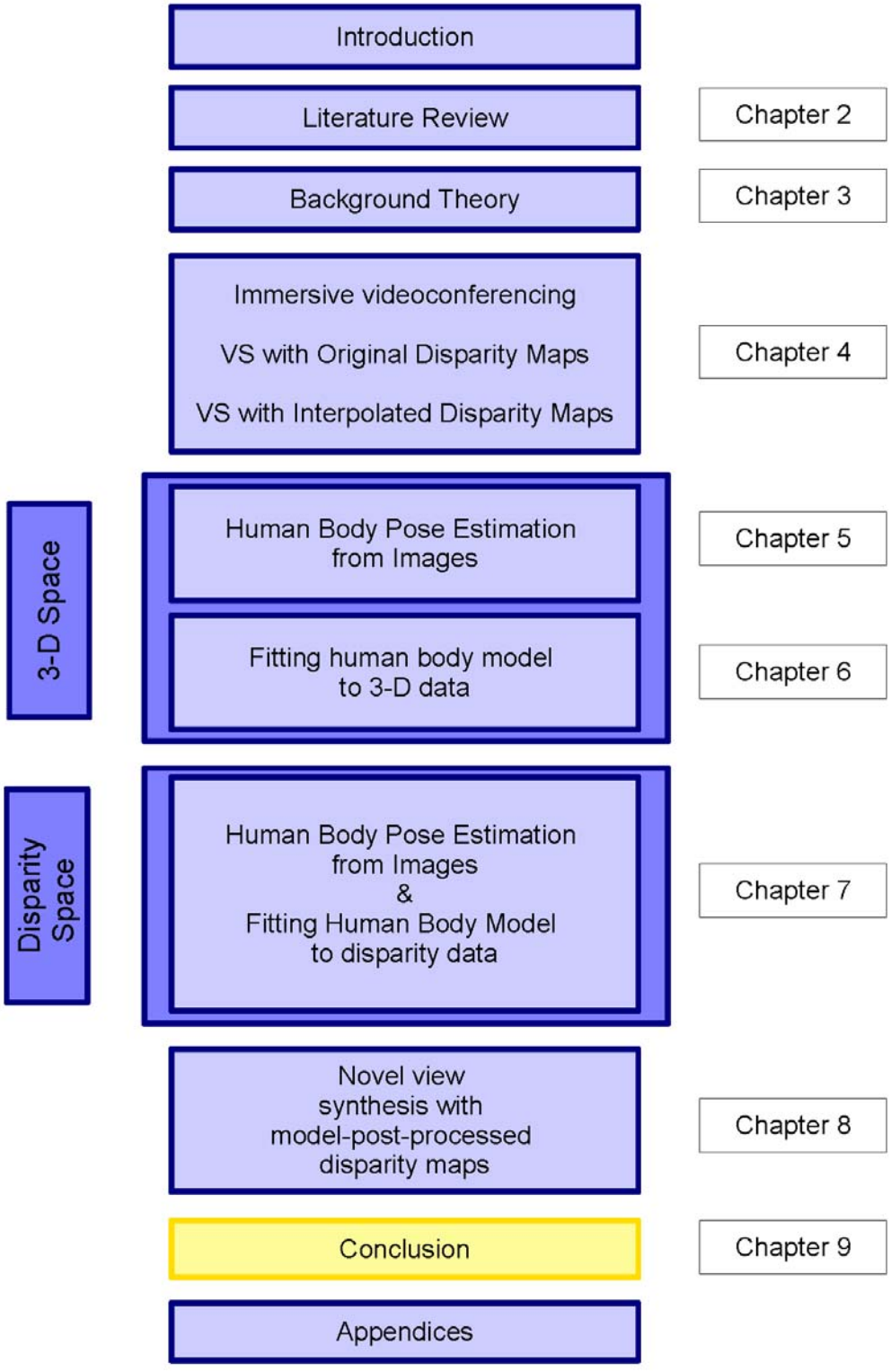


Figure 8.12: The thesis structure diagram.

Chapter 9

Conclusion

9.1 Thesis Summary

In this thesis, we presented an approach to completing patchy and noisy wide-baseline disparity data using an *a priori* generic model of the human body. Two different alternatives for model-based disparity map completion were discussed: modelling the human body in 3-D space and disparity space. We described the model, its pose estimation and the data fitting method for both alternatives. It was experimentally shown on synthetic data with known ground truth that the disparity space algorithm can be computationally less expensive and potentially more accurate. The resulting complete disparity maps were used for novel-view synthesis from wide-baseline stereo pairs using the trilinear tensor transfer approach. This work is specific to the class of view synthesis methods which require dense disparity maps to synthesise novel views.

We reviewed the related work in Chapter 2 and explained the necessary background theory in Chapter 3.

In Chapter 4, we presented the view synthesis in the context of immersive videoconferencing and demonstrated the quality of novel views synthesised using the wide-baseline stereo correspondence data without any post-processing. We also investigated different interpolation methods, the linear, bilinear and cubic interpolation, for disparity map completion.

In Chapter 5, we introduced the subdivision surface human body model and presented the particle swarm optimisation method for articulated pose estimation from multi-view image snapshots. We compared the performance of the pose estimation method on real data with equivalent algorithms using simulated annealing and gradient descent and we established that the particle swarm approach outperformed both other techniques.

In Chapter 6, we described a fitting method based on quasi-interpolation, which was used to deform the subdivision surface body model to better fit the disparity data. The method was demonstrated on accurate range data to show that it was capable of faithfully reproducing the data in 3-5 iterations, as well as on noisy and patchy stereo data for the purpose of disparity map completion.

In Chapter 7, we presented the disparity space alternative to the 3-D model-based disparity map completion. We described the disparity space articulated body model, its articulated pose estimation and model deformation by fitting to disparity data. An experimental comparison of the 3-D approach and the disparity space approach on synthetic data with known ground truth was also presented.

In Chapter 8, we presented the results of the model-based disparity map completion framework. We synthesised novel views and compared them to the ground truth views available in our camera setup. We also discussed the importance of model deformation within our framework and presented an experimental evaluation of the view synthesis results with undeformed and deformed body models.

9.2 Discussion

9.2.1 A note on our choices outlined in the thesis introduction

Stereo correspondence search method. By definition, the disparities for the occluded regions (those regions which are not simultaneously visible in both cameras) cannot be recovered through stereo correspondence alone as there are no pairs of corresponding points

available to compute the disparity from. In this thesis, we propose a completion method which aims to recover the disparities for such occluded regions, i.e., regions which could not be recovered with any stereo correspondence search method unless it resorted to post-processing steps which made assumptions about the scene content. We chose to use a simple and quick pyramidal stereo correspondence search algorithm to generate the input disparity map data and complete the resulting missing disparities with our completion approach. While small-baseline stereo correspondence algorithms, such as those presented on the Middlebury stereo page [134], were not developed to cope with wide-baseline imaging conditions, a more sophisticated dense wide-baseline stereo correspondence search method, *e.g.*, the dense wide-baseline correspondence methods based on interest points described in Chapter 2, Section 2.4, will contribute a denser initial disparity map in non-occluded regions and can be used with our proposed approach as well, however, they will significantly increase the computational complexity of the approach, while not necessarily adding much more information in the occluded regions, unless a large number of views is used as an input to the correspondence search, in which case the method becomes impractical for the videoconferencing scenario.

View synthesis method. Trilinear-tensor point transfer is a projective geometry technique which allows for a theoretically sound way of synthesising a novel view when dense correspondences are available for the input stereo pair of images. Its strong dependence on the quality of disparity maps allows us to clearly demonstrate the improvement in view synthesis achieved by our completion approach, however, the same completion approach would also be useful for any method relying on the quality of disparity map data when synthesising novel views, e.g., [41, 43, 95]. Novel-view synthesis techniques exist, which combine the implicit search for disparities in the form of a photoconsistency likelihood with the prior knowledge of the texture content of the input images [58, 178, 177], thereby successfully disambiguating the multi-modal likelihood with texture information. While their performance is very impressive on images containing complex natural structures with rich and delicate texture, they have not been demonstrated on scenes containing depth variations and occlusions as extreme as ours. These techniques also work with a relatively large number of images, impractical if used in the videoconferencing setting. While we are

hopeful that using such techniques on our images would further improve the fidelity of the novel-view synthesis, we also believe that it would even further highlight the need for the high-quality disparity information for an efficient novel-view synthesis.

9.2.2 Caveats of model-based completion and ways to address them

The presented disparity map completion approach suffers from the same drawbacks as all model-based approaches, primarily concerning the realism of the model. The generic model does not match the real person's shape perfectly, even after the deformation, and this shows in the view synthesis results.

On one hand, the insufficient model deformation is due to the lack of available disparity data in the original disparity map. Although techniques exist which can generate dense disparity maps from wide-baseline stereo pairs, as described in Section 2.4 in Chapter 2, they have so far been formulated as multi-view stereo matching from high-resolution images which relied on slow interest point methods to bootstrap the dense correspondence search and were computationally too demanding to represent a practical solution for our problem (the computation of the dense disparity map per stereo pair of frames with these methods takes longer than our entire disparity map completion framework, despite the fact that the implementation of our approach was not optimised to work in real time). The recent work of Tola *et al.* [165] describes a new local image descriptor which gives rise to an algorithm capable of computing dense wide-baseline disparity maps from only two views with the image resolution comparable to ours and with a considerably lower computational complexity and could therefore realistically replace our simple pyramidal correlation algorithm in order to generate better input disparity data for our completion approach. The aforementioned dense wide-baseline stereo approaches have not been demonstrated on images of articulated bodies and further experimental work is necessary to establish how dense the resulting disparity maps will be in practice. The fact that we estimate the articulated body pose purely from silhouette constraints and therefore have the disparity space model available before the stereo correspondence search also means that the model in the correct pose can be used as a very efficient constraint when deciding on the disparity search interval size,

which should be an important contribution to any stereo correspondence search method that we decide to use in the future.

On the other hand, the insufficient model deformation is also due to the fact that our generic model represents a very weak prior. The deformation method cannot take full advantage of the available disparity data, because the quality of the deformation depends on the mesh density. After one iteration of the fit the model mesh becomes so dense that any further deformations violate the assumptions about the generic model shape and start modelling individual disparity points. Although one way of addressing this problem would be to re-design the model, ensuring that the mesh density in the initial stage of the fit is as low as possible, a more promising approach would be to combine our generic body model with a more informative prior about the generic human body shape and its anthropometric variations, along the lines of recent work by Balan *et al.* [19]. This would allow the model to take full advantage of the available disparity data without worrying about explicitly modelling the noisy disparity values in the process.

When synthesising novel views of people in our setup, even though the entire upper body is shown, the head usually attracts the most attention, because we tend to spend most of the conversation time looking at the conversation partner's face. Our generic model contains a very basic head shape which in combination with the sparse disparity data does not look sufficiently realistic even after the deformation step. It is therefore important to use a more detailed generic model with explicitly modelled generic face features, such as nose and ears, to ensure the head novel view synthesis is as realistic as possible and avoid the flat face effects which can be seen in our results.

The view synthesis results obtained with the body model completion technique also contain artefacts due to the lack of hand models. Although work exists which attempts to recover the articulated hand pose, it usually estimates the pose from images showing only hands, like, *e.g.*, in the human-computer interaction scenarios. In our case, the hands occupy a very small portion of the image while still exhibiting complicated high-DOF articulated motion. Estimating this motion accurately and fast enough not to slow down the

entire completion algorithm represents a difficult problem which remains to be addressed.

9.2.3 Comments on the articulated pose estimation

Due to the low-cost nature of our acquisition setup, we had to limit the experimental work presented in this thesis to multi-view still images of the human body, rather than multi-view synchronised video sequences. This meant that a comparison of our Particle Swarm Optimisation pose estimation approach with state-of-the-art tracking approaches was not possible. In the future work, we plan to extend the pose estimation method to multi-view sequences, which will allow for a meaningful comparison with other generative pose estimation approaches such as particle-filter-based algorithms [101, 149, 51] and Markov model approaches [30].

In contrast, a large body of work has been reported on 3-D pose estimation from monocular still images or monocular sequences. The methods addressing monocular 3-D pose estimation usually fall in the category of discriminative approaches which learn the mapping from various image features directly to 3-D poses and remove the need for an explicit body model, such as the one we use in our work. In order to perform well in domains with large number of different activities, these methods require very large training sets and are therefore often impractical to train. Recently, work has been reported which addresses these limitations and proposes solutions for more efficient training and inference with such discriminative methods [170, 23].

Although they have been shown to perform well on monocular still images and sequences, the discriminative approaches have not been extensively used in multi-view scenarios, such as ours, as learning the direct mapping between multiple views and a 3-D pose represents an even more difficult training problem than learning the same mapping from monocular images. The usability of these methods is also limited to datasets with available accurate marker-based motion capture training data, as the current state-of-the-art techniques do not generalise very well to pose estimation on sequences for which they were not trained.

In a multi-view scenario, the inherent depth ambiguity present in monocular images and sequences, which calls for the use of prior knowledge in the form of training data, becomes less of an issue and analysis-by-synthesis approaches such as ours can be used instead to provide a reliable pose estimate. As we had no motion capture training data available for our upper body images, we could not compare the performance of our approach with that of the state-of-the-art discriminative methods, however, we believe that a much more meaningful performance comparison will be possible once our method is extended to multi-view image sequences, where it can be compared with other, much more related, generative approaches.

9.2.4 Other observations

The view synthesis evaluation shown in Chapter 8 highlighted the fact that believable novel views can be synthesised by using the generic human body model alone, without the extra step of skin deformation. While the results will only be as realistic as the model itself, this is still an interesting prospect when dealing with very patchy disparity data or even when the disparity map computation is for some reason not possible.

Our novel-view synthesis experimental results contain artefacts due to wide-baseline imaging conditions which can be alleviated through pre-processing of input image textures before they are used for novel-view synthesis. For example, the camera colour response can be normalised to correct for the difference in the texture colour content of the stereo pair and the background segmentation can be made less conservative to exclude all background pixels. An alpha-matting technique could also be used to make the blend between textures sourced from different views more seamless.

9.3 Future Work

The work presented in this thesis offers several possible directions for future research.

The disparity space concept offers an interesting alternative to 3-D space. Although the disparity space modelling is based on the concept of “modelling in 3 dimensions”, it is

somewhat closer to image-based rendering than its 3-D counterpart, in the sense that the step of 3-D reconstruction from corresponding pairs of points is avoided. The associated approximation is therefore also avoided, which creates potential for more accurate results. While it might seem from this work that the concept of disparity space is somewhat limited to the problem that we addressed in this thesis, this is in fact not the case. Several researchers have recognised its importance in relation to data accuracy and homoscedasticity and several research articles have been published on this topic, primarily in the area of rigid motion estimation from stereo [71, 8, 49]. A systematic comparison of the algorithm performance on real data in disparity space and 3-D space would provide further insight into the proposed disparity space method.

Another interesting possibility is a development of a more general disparity space theory for the algorithms presented in this thesis, which would not require rectified stereo pairs and could deal with disparities as vectors rather than scalars. Although a rectified stereo setup is not in itself a limitation, as any stereo pair of cameras can be rectified, it would be interesting to see whether, for a general stereo pair, this process could be avoided while still benefiting from the nice properties of disparity space, although in this case in four dimensions.

The algorithms presented in this thesis were developed as a proof-of-concept, rather than a real-time application. An implementation focusing on faster execution times would make the proposed approach more readily applicable to the problem of novel-view synthesis for immersive videoconferencing. At the moment, the total processing time needed to estimate the articulated pose, fit the model to data and generate a novel view takes from 1.5 to 2 minutes per multi-view set of images. The bottleneck of our completion approach is the pose estimation step, which in our case takes up the vast majority of the processing time. At least two solutions to this problem are immediately available. The first solution is a parallel implementation of PSO, which has already been reported in the literature [138] and would allow us to speed up the approach by distributing the computation across several CPUs. The second solution is related to tracking in video sequences. At the moment, the articulated upper body pose is estimated from scratch for every test pose, as there is no tem-

poral continuity in the still-image test data. Once the approach is extended to sequences, it will not be necessary to estimate the entire pose from scratch for every time step. Instead, the previous frame estimate will be used to guide the pose estimation in the next frame, which should also reduce the complexity of the method.

Automatic initialisation of body model dimensions is another area where this work can be extended. Using our approach on images of different people requires the model to be initialised to their body proportions. We currently initialise the model dimensions manually, however, attempts have been reported by researchers to customise the *a priori* model's shape to that of a specific person automatically [31, 150].

The results of the pose estimation algorithm on multi-view snapshots are very encouraging and, on their own, provide an excellent tool for an automatic initialisation of an articulated body tracking in a video sequence. A lot of work has been published on such articulated tracking, which could use our algorithm as an initialisation. Additionally, an extension of our algorithm to video sequences, using a temporal consistency strategy in the form of a motion model, would allow us to compare the performance of the PSO pose estimation with other established tracking approaches, especially in the light of the recent HumanEva initiative [146].

Recent work by Balan et al. [19] on using more realistic human body models trained from a database of body scans and optical motion capture data also opens new possibilities for our pose estimation approach as the use of more realistic models is very likely to improve the results of the pose and shape estimation with a 3-D or disparity space human body model.

The extension of the subdivision surface fitting method to work on video sequences is also a possible research direction. With new stereo data available at every time step, the generic model could be incrementally improved through time and, with a sufficiently high-quality stereo algorithm, the generic model could eventually be entirely customised to a particular person when an appropriately long video sequence of that person was available.

Finally, let us touch on the question of image-based rendering and the role of our technique in this area. One could argue that the presented approach is just another way of explicitly modelling the scene for the purpose of novel-view synthesis and as such represents a step away from the image-based rendering paradigm that has been advocated a lot recently. While we accept such criticism as justified, we would also emphasise that IBR research shows a clear need for prior knowledge to be included in the synthesis process, be it in the form of depth smoothness priors [154] or texture priors [58]. Our work has demonstrated one possible way of doing so, that is, modelling the geometry of the scene in disparity space, and we hope that this work will help further advance the state of the art in the image-based rendering and novel-view synthesis.

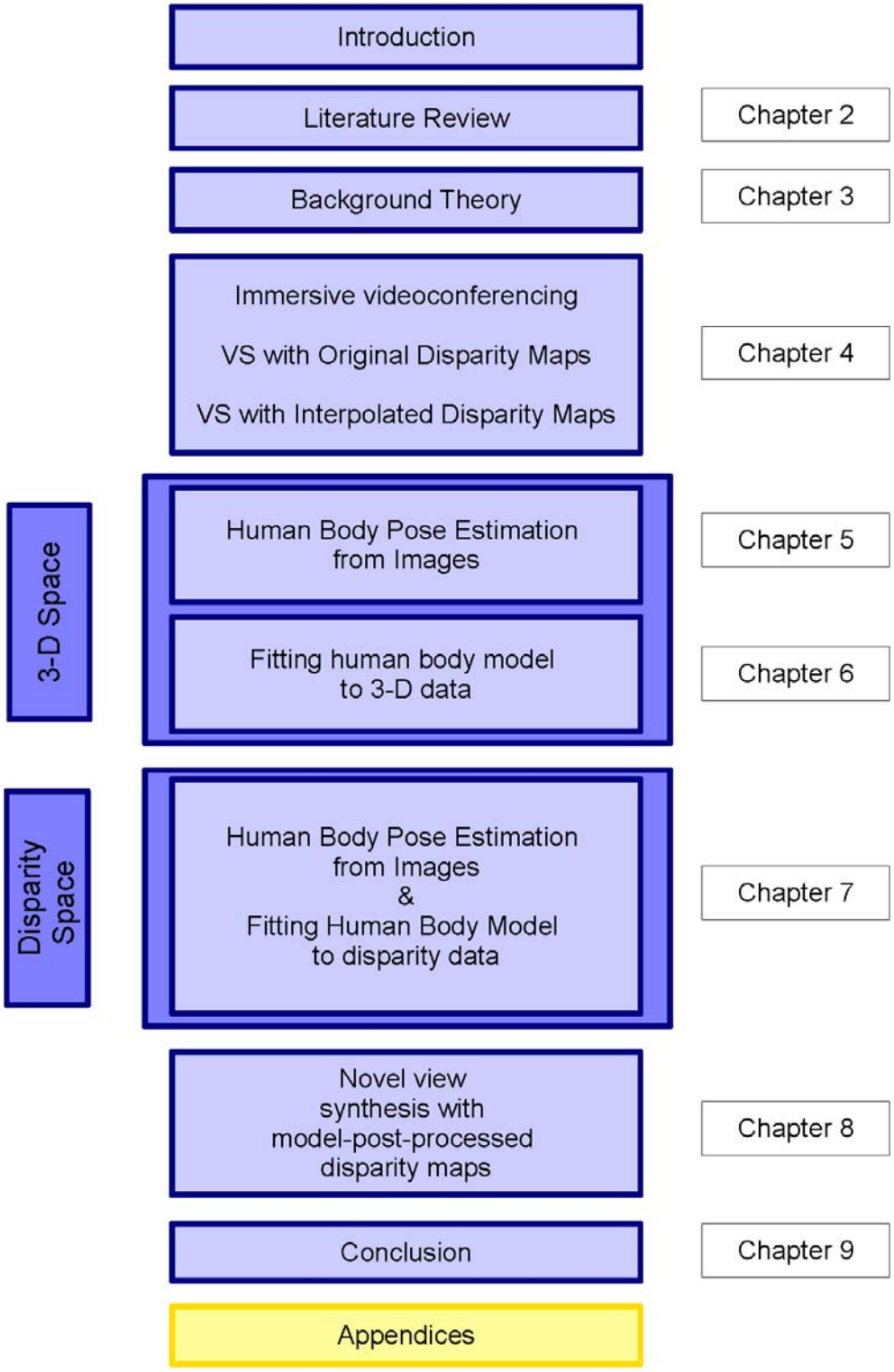


Figure 9.1: The thesis structure diagram.

Appendix A

Cross Product with Skew-Symmetric Matrices

Vector cross products can be expressed using 3×3 skew-symmetric matrices. The following description is taken from Hartley and Zisserman [70].

If $\mathbf{a} = (a_1, a_2, a_3)^\top$ is a 3-vector, where $^\top$ denotes vector transpose operator, then one defines a corresponding skew-symmetric matrix as follows:

$$[\mathbf{a}]_\times = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}. \tag{A.1}$$

Note that any skew-symmetric 3×3 matrix may be written in the form $[\mathbf{a}]_\times$ for a suitable vector $\mathbf{a}$. Matrix $[\mathbf{a}]_\times$ is singular and $\mathbf{a}$ is its null-vector. Hence, a 3×3 skew-symmetric matrix is defined up to a scale by its null vector.

The cross product, also called vector product, of two 3-vectors $\mathbf{a} \times \mathbf{b}$ is the vector $(a_2b_3 - a_3b_2, a_3b_1 - a_1b_3, a_1b_2 - a_2b_1)^\top$. The cross product is related to skew-symmetric matrices according to

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = (\mathbf{a}^\top [\mathbf{b}]_\times)^\top. \tag{A.2}$$

Appendix B

Rectifying Homography for Metric Reconstruction

We present a detailed step-by-step derivation of the rectifying homography for the metric reconstruction of cameras and structure, used in the multi-camera self-calibration method by Svoboda *et al.* [159].

Definitions: Single Camera. Let a projection matrix $\hat{P}$ of a calibrated finite projective camera be defined as:

$$\hat{P} \sim K[R|\mathbf{t}] = \mu K[R|\mathbf{t}], \tag{B.1}$$

with

$$K = \begin{bmatrix} f & 0 & x_0 \\ 0 & \alpha f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} \mathbf{i}^\top \\ \mathbf{j}^\top \\ \mathbf{k}^\top \end{bmatrix} \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{B.2}$$

where K is the calibration matrix with intrinsic parameters including focal length f, aspect ratio α, and principal point (x_0, y_0). The skew is assumed to be zero. R is the rotation matrix with **i**, **j** and **k** denoting the direction unit vectors of the individual rotation axes, **t** is the translation vector and μ is a non-zero scale factor.

For the purpose of the derivation we will also use the following notation for the camera

matrix $\hat{P}$:

$$\hat{P} = \mu K[R|\mathbf{t}] = [\mu KR|\mu K\mathbf{t}] = [M|\mathbf{T}] = \left[\begin{array}{c|c} \mathbf{m}_x^\top & T_x \\ \mathbf{m}_y^\top & T_y \\ \mathbf{m}_z^\top & T_z \end{array}\right] \tag{B.3}$$

which gives:

$$M = \begin{bmatrix} \mathbf{m}_x^\top \\ \mathbf{m}_y^\top \\ \mathbf{m}_z^\top \end{bmatrix} = \mu KR = \begin{bmatrix} \mu f\mathbf{i}^\top & + & \mu x_0\mathbf{k}^\top \\ \mu\alpha f\mathbf{j}^\top & + & \mu y_0\mathbf{k}^\top \\ & \mu\mathbf{k}^\top & \end{bmatrix} \tag{B.4}$$

One can expand $\mathbf{T} = \mu K\mathbf{t}$ in a similar way.

Let $\hat{\mathbf{X}}$ denote a $3D$ point in Euclidean space, written in homogeneous coordinates. Projecting $\hat{\mathbf{X}}$ onto the image plane of the camera $\hat{P}$ gives:

$$\lambda\mathbf{x} = \hat{P}\hat{\mathbf{X}}, \tag{B.5}$$

where

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \textit{and} \quad \hat{\mathbf{X}} = \begin{bmatrix} \nu x \\ \nu y \\ \nu z \\ \nu \end{bmatrix} = \begin{bmatrix} \nu\mathbf{s} \\ \nu \end{bmatrix}, \tag{B.6}$$

λ is the projective depth of the image point $\mathbf{x}$ and $\mathbf{s} = [x\ y\ z]^\top$.

Definitions: Multiple Cameras. Let us denote the i-th calibrated finite projective camera matrix as $\hat{P}^i$, where

$$\hat{P}^i \sim K^i[R^i|t^i] = \mu^i K^i[R^i|t^i] \tag{B.7}$$

and

$$K^i = \begin{bmatrix} f^i & 0 & x_0^i \\ 0 & \alpha^i f^i & y_0^i \\ 0 & 0 & 1 \end{bmatrix} \quad R^i = \begin{bmatrix} \mathbf{i}^{i\top} \\ \mathbf{j}^{i\top} \\ \mathbf{k}^{i\top} \end{bmatrix} \quad \mathbf{t}^i = \begin{bmatrix} t_x^i \\ t_y^i \\ t_z^i \end{bmatrix} \tag{B.8}$$

From (B.3) we also have

$$\hat{P}^i = [M^i|\mathbf{T}^i] = \left[\begin{array}{c|c} \mathbf{m}_x^{i\top} & T_x^i \\ \mathbf{m}_y^{i\top} & T_y^i \\ \mathbf{m}_z^{i\top} & T_z^i \end{array}\right], \tag{B.9}$$

which gives (see B.4):

$$M^i = \begin{bmatrix} \mathbf{m}_x^{i\top} \\ \mathbf{m}_y^{i\top} \\ \mathbf{m}_z^{i\top} \end{bmatrix} = \mu^i K^i R^i = \begin{bmatrix} \mu^i f^i \mathbf{i}^{i\top} + \mu x_0^i \mathbf{k}^{i\top} \\ \mu^i \alpha f^i \mathbf{j}^{i\top} + \mu y_0^i \mathbf{k}^{i\top} \\ \mu^i \mathbf{k}^{i\top} \end{bmatrix} \tag{B.10}$$

Definitions: Structure and Motion. Combining the projection matrices $\hat{P}^i$ for all $i = 1 \ldots m$ cameras into one matrix gives:

$$\hat{P}_{3m\times 4} = \begin{bmatrix} \hat{P}^1 \\ \hat{P}^2 \\ \vdots \\ \hat{P}^m \end{bmatrix} = \begin{bmatrix} [M^1|\mathbf{T}^1] \\ [M^2|\mathbf{T}^2] \\ \vdots \\ [M^m|\mathbf{T}^m] \end{bmatrix} = [M_{3m\times 3}|\mathbf{T}_{3m\times 1}], \tag{B.11}$$

where

$$\begin{aligned} M_{3m\times 3} &= [M^{1\top}, M^{2\top}, \ldots, M^{m\top}]^\top \\ &= [\mathbf{m}_x^1, \mathbf{m}_y^1, \mathbf{m}_z^1, \ldots, \mathbf{m}_x^m, \mathbf{m}_y^m, \mathbf{m}_z^m]^\top \end{aligned} \tag{B.12}$$

and

$$\mathbf{T}_{3m\times 1} = [T_x^1, T_y^1, T_z^1, \ldots, T_x^m, T_y^m, T_z^m]^\top$$

The *shape matrix* is a collection of 3D points written in homogeneous coordinates:

$$\hat{X}_{4\times n} = [\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_2, \ldots, \hat{\mathbf{X}}_n] = \begin{bmatrix} \nu_1 \mathbf{s}_1 & \nu_2 \mathbf{s}_2 & \ldots & \nu_n \mathbf{s}_n \\ \nu_1 & \nu_2 & \ldots & \nu_n \end{bmatrix} \tag{B.13}$$

where

$$\mathbf{s}_j = \begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} \text{ and } \hat{\mathbf{X}}_j = \begin{bmatrix} \nu_j \mathbf{s}_j \\ \nu_j \end{bmatrix} \tag{B.14}$$

Definitions: Rectifying Homography. We are looking for a 4×4 homography H which will transform the projective motion P and shape X, recovered by factorisation, into Euclidean motion $\hat{P}$ and shape $\hat{X}$:

$$\hat{P}\hat{X} = PHH^{-1}X \tag{B.15}$$

where

$$\begin{aligned} \hat{P} &= PH \\ \hat{X} &= H^{-1}X \end{aligned} \tag{B.16}$$

Let the unknown homography H be defined as:

$$H = [A \mid \mathbf{b}], \tag{B.17}$$

where A is a 4×3 matrix and $\mathbf{b}$ is a 4×1 vector.

Computation of H. In order to find the homography H, we separately derive two linear systems, one in the unknowns of the matrix A and the other in the unknowns of vector $\mathbf{b}$. We start with the relationship between the Euclidean and projective motion (B.16):

$$\hat{P} = PH \tag{B.18}$$

Using Equation (B.3) and Equation (B.17) we get:

$$[M|\mathbf{T}] = P[A|\mathbf{b}] \tag{B.19}$$

which gives:

$$M = PA \tag{B.20}$$

and

$$\mathbf{T} = P\,\mathbf{b} \tag{B.21}$$

Equations (B.20) and (B.21) will be a starting point for the computation of A and $\mathbf{b}$.

Computation of A. Starting from Equation (B.20) we get:

$$MM^{\top} = PAA^{\top}P^{\top} \tag{B.22}$$

Setting $Q = AA^{\top}$ allows us to solve a linear system in the unknown entries of matrix Q by taking advantage of the known constraints on the matrix $MM^{\top}$:

$$MM^{\top} = PQP^{\top}, \tag{B.23}$$

and recover matrix A by rank-3 factorization of matrix Q using SVD.

Multiplying out $MM^\top$ gives:

$$\begin{bmatrix}
\mathbf{m}_x^{1\top}\mathbf{m}_x^1 & \mathbf{m}_x^{1\top}\mathbf{m}_y^1 & \mathbf{m}_x^{1\top}\mathbf{m}_z^1 & \dots & \mathbf{m}_x^{1\top}\mathbf{m}_x^m & \mathbf{m}_x^{1\top}\mathbf{m}_y^m & \mathbf{m}_x^{1\top}\mathbf{m}_z^m \\
\mathbf{m}_y^{1\top}\mathbf{m}_x^1 & \mathbf{m}_y^{1\top}\mathbf{m}_y^1 & \mathbf{m}_y^{1\top}\mathbf{m}_z^1 & \dots & \mathbf{m}_y^{1\top}\mathbf{m}_x^m & \mathbf{m}_y^{1\top}\mathbf{m}_y^m & \mathbf{m}_y^{1\top}\mathbf{m}_z^m \\
\mathbf{m}_z^{1\top}\mathbf{m}_x^1 & \mathbf{m}_z^{1\top}\mathbf{m}_y^1 & \mathbf{m}_z^{1\top}\mathbf{m}_z^1 & \dots & \mathbf{m}_z^{1\top}\mathbf{m}_x^m & \mathbf{m}_z^{1\top}\mathbf{m}_y^m & \mathbf{m}_z^{1\top}\mathbf{m}_z^m \\
\mathbf{m}_x^{2\top}\mathbf{m}_x^1 & \mathbf{m}_x^{2\top}\mathbf{m}_y^1 & \mathbf{m}_x^{2\top}\mathbf{m}_z^1 & \dots & \mathbf{m}_x^{2\top}\mathbf{m}_x^m & \mathbf{m}_x^{2\top}\mathbf{m}_y^m & \mathbf{m}_x^{2\top}\mathbf{m}_z^m \\
\mathbf{m}_y^{2\top}\mathbf{m}_x^1 & \mathbf{m}_y^{2\top}\mathbf{m}_y^1 & \mathbf{m}_y^{2\top}\mathbf{m}_z^1 & \dots & \mathbf{m}_y^{2\top}\mathbf{m}_x^m & \mathbf{m}_y^{2\top}\mathbf{m}_y^m & \mathbf{m}_y^{2\top}\mathbf{m}_z^m \\
\mathbf{m}_z^{2\top}\mathbf{m}_x^1 & \mathbf{m}_z^{2\top}\mathbf{m}_y^1 & \mathbf{m}_z^{2\top}\mathbf{m}_z^1 & \dots & \mathbf{m}_z^{2\top}\mathbf{m}_x^m & \mathbf{m}_z^{2\top}\mathbf{m}_y^m & \mathbf{m}_z^{2\top}\mathbf{m}_z^m \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
\mathbf{m}_x^{m\top}\mathbf{m}_x^1 & \mathbf{m}_x^{m\top}\mathbf{m}_y^1 & \mathbf{m}_x^{m\top}\mathbf{m}_z^1 & \dots & \mathbf{m}_x^{m\top}\mathbf{m}_x^m & \mathbf{m}_x^{m\top}\mathbf{m}_y^m & \mathbf{m}_x^{m\top}\mathbf{m}_z^m \\
\mathbf{m}_y^{m\top}\mathbf{m}_x^1 & \mathbf{m}_y^{m\top}\mathbf{m}_y^1 & \mathbf{m}_y^{m\top}\mathbf{m}_z^1 & \dots & \mathbf{m}_y^{m\top}\mathbf{m}_x^m & \mathbf{m}_y^{m\top}\mathbf{m}_y^m & \mathbf{m}_y^{m\top}\mathbf{m}_z^m \\
\mathbf{m}_z^{m\top}\mathbf{m}_x^1 & \mathbf{m}_z^{m\top}\mathbf{m}_y^1 & \mathbf{m}_z^{m\top}\mathbf{m}_z^1 & \dots & \mathbf{m}_z^{m\top}\mathbf{m}_x^m & \mathbf{m}_z^{m\top}\mathbf{m}_y^m & \mathbf{m}_z^{m\top}\mathbf{m}_z^m
\end{bmatrix} \tag{B.24}$$

At this point we introduce the constraints on $MM^\top$. We assume square pixels, $\alpha^i = 1$, and the principal points to be known, $x_0^i = 0, y_0^i = 0$. Introducing these constraints into (B.10) gives:

$$\begin{aligned}
\mathbf{m}_x^{i\top} &= \mu^i f^i \mathbf{i}^{i\top} \\
\mathbf{m}_y^{i\top} &= \mu^i f^i \mathbf{j}^{i\top} \\
\mathbf{m}_z^{i\top} &= \mu^i \mathbf{k}^{i\top}
\end{aligned} \tag{B.25}$$

and consequently:

$$\|\mathbf{m}_x^i\|^2 = \|\mathbf{m}_y^i\|^2.$$

As the elements of the matrix M are scaled axes of rotation (see Equation (B.25)), they are mutually orthogonal and the following constraints in the elements of $MM^\top$ hold:

$$\begin{aligned}
\mathbf{m}_x^{i\top}\mathbf{m}_y^i &= 0 \\
\mathbf{m}_x^{i\top}\mathbf{m}_z^i &= 0 \\
\mathbf{m}_y^{i\top}\mathbf{m}_z^i &= 0
\end{aligned} \tag{B.26}$$

Substituting equations (B.25) and (B.26) into (B.24) yields the following matrix $MM^\top$:

$$\begin{bmatrix}
\|\mathbf{m}_x^1\|^2 & 0 & 0 & \dots & \mathbf{m}_x^{1\top}\mathbf{m}_x^m & \mathbf{m}_x^{1\top}\mathbf{m}_y^m & \mathbf{m}_x^{1\top}\mathbf{m}_z^m \\
0 & \|\mathbf{m}_x^1\|^2 & 0 & \dots & \mathbf{m}_y^{1\top}\mathbf{m}_x^m & \mathbf{m}_y^{1\top}\mathbf{m}_y^m & \mathbf{m}_y^{1\top}\mathbf{m}_z^m \\
0 & 0 & \|\mathbf{m}_z^1\|^2 & \dots & \mathbf{m}_z^{1\top}\mathbf{m}_x^m & \mathbf{m}_z^{1\top}\mathbf{m}_y^m & \mathbf{m}_z^{1\top}\mathbf{m}_z^m \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
\mathbf{m}_x^{m\top}\mathbf{m}_x^1 & \mathbf{m}_x^{m\top}\mathbf{m}_y^1 & \mathbf{m}_x^{m\top}\mathbf{m}_z^1 & \dots & \|\mathbf{m}_x^m\|^2 & 0 & 0 \\
\mathbf{m}_y^{m\top}\mathbf{m}_x^1 & \mathbf{m}_y^{m\top}\mathbf{m}_y^1 & \mathbf{m}_y^{m\top}\mathbf{m}_z^1 & \dots & 0 & \|\mathbf{m}_x^m\|^2 & 0 \\
\mathbf{m}_z^{m\top}\mathbf{m}_x^1 & \mathbf{m}_z^{m\top}\mathbf{m}_y^1 & \mathbf{m}_z^{m\top}\mathbf{m}_z^1 & \dots & 0 & 0 & \|\mathbf{m}_z^m\|^2
\end{bmatrix} \tag{B.27}$$

Let the elements of the matrix Q be denoted as q_{ij} and the rows of the matrix P^i as $\mathbf{P}_1^{i\top}, \mathbf{P}_2^{i\top}$ and $\mathbf{P}_3^{i\top}$ for the first, second and third row, respectively. The right side of (B.23) can then be written out as follows:

$$\begin{bmatrix} \mathbf{P}_1^1\mathbf{P}_2^1\dots\mathbf{P}_3^m \end{bmatrix}^\top \begin{bmatrix}
q_{11} & q_{12} & q_{13} & q_{14} \\
q_{12} & q_{22} & q_{23} & q_{24} \\
q_{13} & q_{23} & q_{33} & q_{34} \\
q_{14} & q_{24} & q_{34} & q_{44}
\end{bmatrix} \begin{bmatrix} \mathbf{P}_1^1\mathbf{P}_2^1\dots\mathbf{P}_3^m \end{bmatrix} \tag{B.28}$$

Matrix $Q = AA^\top$ is symmetric and therefore only has 10 independent elements. The linear system in the unknowns of Q can now be set up by substituting the 3×3 submatrices on the main diagonal of Equation (B.27) together with Equation (B.28) in Equation (B.23). The constraints we get from the i-th projection matrix P^i, *i.e.*, the i-th diagonal submatrix of (B.27), are the following:

$$\begin{aligned}
0 &= \mathbf{P}_1^{i\top} Q \mathbf{P}_1^i - \mathbf{P}_2^{i\top} Q \mathbf{P}_2^i \\
0 &= \mathbf{P}_1^{i\top} Q \mathbf{P}_2^i \ (= \mathbf{P}_2^{i\top} Q \mathbf{P}_1^i) \\
0 &= \mathbf{P}_1^{i\top} Q \mathbf{P}_3^i \ (= \mathbf{P}_3^{i\top} Q \mathbf{P}_1^i) \\
0 &= \mathbf{P}_2^{i\top} Q \mathbf{P}_3^i \ (= \mathbf{P}_3^{i\top} Q \mathbf{P}_2^i)
\end{aligned} \tag{B.29}$$

Each camera projection matrix contributes 4 constraints in the unknown elements of matrix Q. At least 3 views are necessary to sufficiently constrain the 10 unknowns of Q.

Constraints (B.29) can be rewritten as a homogeneous system of linear equations in the 10 unknowns of Q as follows. Let $\mathbf{x} = [x_1, x_2, x_3, x_4]^\top$ and $\mathbf{y} = [y_1, y_2, y_3, y_4]^\top$. The constraint $\mathbf{x}^\top Q \mathbf{y} = 0$ is then written as one linear equation in the unknowns of Q:

$$\begin{bmatrix} x_1y_1 \\ x_1y_2 + x_2y_1 \\ x_1y_3 + x_3y_1 \\ x_1y_4 + x_4y_1 \\ x_2y_2 \\ x_2y_3 + x_3y_2 \\ x_2y_4 + x_4y_2 \\ x_3y_3 \\ x_3y_4 + x_4y_3 \\ x_4y_4 \end{bmatrix}^\top \begin{bmatrix} q_{11} \\ q_{12} \\ q_{13} \\ q_{14} \\ q_{22} \\ q_{23} \\ q_{24} \\ q_{33} \\ q_{34} \\ q_{44} \end{bmatrix} = \mathbf{0} \tag{B.30}$$

Writing out 4 linear equations for every camera matrix P^i gives a homogeneous linear system of $4m$ equations in the unknowns of Q which can be solved using singular value decomposition (SVD). Matrix A is then recovered by rank-3 decomposition of Q.

Computation of b. Starting from Equation (B.21) we get a set of constraints on the elements of $\mathbf{b}$:

$$T_x^i = \mathbf{P}_x^{i\top}\mathbf{b}, \;\; T_y^i = \mathbf{P}_y^{i\top}\mathbf{b}, \;\; T_z^i = \mathbf{P}_z^{i\top}\mathbf{b}. \tag{B.31}$$

Further constraints on T_x^i, T_y^i and T_z^i can be obtained from the scaled measurement matrix W by calculating a sum of all image points seen by the i-th camera:

$$\begin{aligned} \sum_{j=1}^{n} \lambda_j^i \mathbf{x}_j^i &= \sum_{j=1}^{n} \begin{bmatrix} \lambda_j^i x_j^i \\ \lambda_j^i y_j^i \\ \lambda_j^i \end{bmatrix} = \sum_{j=1}^{n} \hat{P}^i \hat{\mathbf{X}}_j = \sum_{j=1}^{n} ([M_j^i | \mathbf{T}_j^i] \begin{bmatrix} \nu_j \mathbf{s}_j \\ \nu_j \end{bmatrix}) \\ &= \sum_{j=1}^{n} (\left[\begin{array}{c|c} \mathbf{m}_x^i & T_x^i \\ \mathbf{m}_y^i & T_y^i \\ \mathbf{m}_z^i & T_z^i \end{array}\right] \begin{bmatrix} \nu_j \mathbf{s}_j \\ \nu_j \end{bmatrix}) \end{aligned} \tag{B.32}$$

This gives:

$$\begin{aligned}
\sum_{j=1}^{n} \lambda_j^i x_j^i &= \sum_{j=1}^{n} (\mathbf{m}_x^i \cdot \nu_j \mathbf{s}_j + T_x^i \nu_j) = \mathbf{m}_x^i \cdot \sum_{j=1}^{n} \nu_j \mathbf{s}_j + T_x^i \sum_{j=1}^{n} \nu_j \\
\sum_{j=1}^{n} \lambda_j^i y_j^i &= \sum_{j=1}^{n} (\mathbf{m}_y^i \cdot \nu_j \mathbf{s}_j + T_y^i \nu_j) = \mathbf{m}_y^i \cdot \sum_{j=1}^{n} \nu_j \mathbf{s}_j + T_y^i \sum_{j=1}^{n} \nu_j \qquad \text{(B.33)} \\
\sum_{j=1}^{n} \lambda_j^i &= \sum_{j=1}^{n} (\mathbf{m}_z^i \cdot \nu_j \mathbf{s}_j + T_z^i \nu_j) = \mathbf{m}_z^i \cdot \sum_{j=1}^{n} \nu_j \mathbf{s}_j + T_z^i \sum_{j=1}^{n} \nu_j
\end{aligned}$$

Assuming that the origin of the world frame is in the centroid of the scaled 3D points, we get:

$$\sum_{j=1}^{n} \nu_j \mathbf{s}_j = 0, \qquad \text{(B.34)}$$

and using this assumption in (B.33) gives:

$$\begin{aligned}
\sum_{j=1}^{n} \lambda_j^i x_j^i &= T_x^i \sum_{j=1}^{n} \nu_j \\
\sum_{j=1}^{n} \lambda_j^i y_j^i &= T_y^i \sum_{j=1}^{n} \nu_j \\
\sum_{j=1}^{n} \lambda_j^i &= T_z^i \sum_{j=1}^{n} \nu_j \qquad \text{(B.35)}
\end{aligned}$$

Writing out the ratios of (B.35) we get:

$$\frac{T_x^i}{T_z^i} = \frac{\sum_{j=1}^{n} \lambda_j^i x_j^i}{\sum_{j=1}^{n} \lambda_j^i} \quad \textit{and} \quad \frac{T_y^i}{T_z^i} = \frac{\sum_{j=1}^{n} \lambda_j^i y_j^i}{\sum_{j=1}^{n} \lambda_j^i} \qquad \text{(B.36)}$$

From (B.36) follows:

$$\begin{aligned}
T_x^i \sum_{j=1}^{n} \lambda_j^i &- T_z^i \sum_{j=1}^{n} \lambda_j^i x_j^i = 0 \\
T_y^i \sum_{j=1}^{n} \lambda_j^i &- T_z^i \sum_{j=1}^{n} \lambda_j^i y_j^i = 0 \qquad \text{(B.37)}
\end{aligned}$$

and using (B.31):

$$\begin{aligned} \mathbf{P}_x^{i\top}\mathbf{b}\sum_{j=1}^{n}\lambda_j^i - \mathbf{P}_z^{i\top}\mathbf{b}\sum_{j=1}^{n}\lambda_j^i x_j^i &= 0 \\ \mathbf{P}_y^{i\top}\mathbf{b}\sum_{j=1}^{n}\lambda_j^i - \mathbf{P}_z^{i\top}\mathbf{b}\sum_{j=1}^{n}\lambda_j^i y_j^i &= 0 \end{aligned} \tag{B.38}$$

From Equation (B.38) we can set up a system of $2m$ homogeneous linear equations in the 4 unknown elements of $\mathbf{b}$:

$$B\,\mathbf{b} = 0 \tag{B.39}$$

where

$$B = \begin{bmatrix} \mathbf{P}_x^1\sum_{j=1}^{n}\lambda_j^1 & - & \mathbf{P}_z^1\sum_{j=1}^{n}\lambda_j^1 x_j^1 \\ \mathbf{P}_y^1\sum_{j=1}^{n}\lambda_j^1 & - & \mathbf{P}_z^1\sum_{j=1}^{n}\lambda_j^1 y_j^1 \\ & \vdots & \\ \mathbf{P}_x^m\sum_{j=1}^{n}\lambda_j^n & - & \mathbf{P}_z^m\sum_{j=1}^{n}\lambda_j^n x_j^n \\ \mathbf{P}_y^m\sum_{j=1}^{n}\lambda_j^n & - & \mathbf{P}_z^m\sum_{j=1}^{n}\lambda_j^n y_j^n \end{bmatrix} \tag{B.40}$$

and $\mathbf{b}$ is the singular vector corresponding to the smallest singular value in the SVD decomposition of B. The unknown homography H is then composed as in Equation (B.17) and the Euclidean motion and structure recovered as in Equation(B.16).

Appendix C

The ε_{ijk} Tensor

The tensor ε_{ijk} is defined for $i, j, k = 1, \ldots, 3$ as follows:

$$\varepsilon_{i,j,k} = \begin{cases} 0 & \text{unless } i,\ j \text{ and } k \text{ are distinct} \\ +1 & \text{if } ijk \text{ is an even permutation of } 123 \\ -1 & \text{if } ijk \text{ is an odd permutation of } 123 \end{cases} \tag{C.1}$$

The tensor ε_{ijk} is connected with the cross product of two vectors. If $\mathbf{a}$ and $\mathbf{b}$ are two vectors, and $\mathbf{c} = \mathbf{a} \times \mathbf{b}$ is their cross product, then

$$c_i = (\mathbf{a} \times \mathbf{b})_i = \varepsilon_{ijk} a^j b^k \tag{C.2}$$

and the skew-symmetric matrix $[a]_\times$ in tensor notation is written as

$$([a]_\times)_{ik} = \varepsilon_{ijk} a^j \tag{C.3}$$

Appendix D

View Synthesis Comparison Tables

In Chapter 8, we compared the quality of the novel views synthesised using disparity maps which were completed with a generic and undeformed disparity space body model with novel views synthesised using disparity maps completed with disparity space body models which were deformed to better fit the disparity data. We compared the resulting novel views with the ground truth views available in our camera setup and showed the results in comparison graphs in Figures 8.1, 8.2 and 8.3.

The tables in this appendix list the numbers which were used to plot these graphs. Every entry in the table represents an average RGB distance, as defined in Equation 8.1, for one view of a particular pose using a particular completion method. For example, the first entry in Table D.1 represents the average RGB distance between the ground truth view of camera 1 and the equivalent novel view of pose 1 synthesised using the original disparity map and using camera 1 as the virtual camera.

Table D.1: Subject S - average RGB distance from GT for camera 1

	Original DM	Undeformed Model	Deformed Model
Pose 1	57.978260	26.158072	26.157367
Pose 2	55.699688	23.348503	23.016029
Pose 3	68.347527	26.933405	24.915596
Pose 4	41.030621	16.490496	16.440322
Pose 5	57.331276	19.262732	18.637751
Pose 6	48.955353	14.811069	14.811027
Pose 7	58.428795	20.459679	20.448526
Pose 8	54.643814	16.110243	15.981818
Pose 9	49.125759	15.504137	15.411086
Pose 10	57.998737	22.464500	21.330364
Pose 11	66.999466	24.640265	24.090151
Pose 12	62.861015	25.520969	24.611042

Table D.2: Subject S - average RGB distance from GT for camera 3

	Original DM	Undeformed Model	Deformed Model
Pose 1	73.996101	24.939896	24.353365
Pose 2	69.901886	22.992050	22.302734
Pose 3	70.943695	25.634468	23.748228
Pose 4	52.615974	15.471055	15.470680
Pose 5	68.581573	19.088869	18.114635
Pose 6	61.893013	19.148748	17.944260
Pose 7	79.429413	22.934198	21.465199
Pose 8	69.101997	17.913002	17.531750
Pose 9	58.229519	17.774149	17.270882
Pose 10	75.150276	23.675049	23.397133
Pose 11	75.737358	24.799362	24.180780
Pose 12	70.619522	24.039560	22.833471

Table D.3: Subject S - average RGB distance from GT for camera 4

	Original DM	Undeformed Model	Deformed Model
Pose 1	60.764526	23.363453	23.215313
Pose 2	57.790417	22.539692	21.691345
Pose 3	55.340195	22.470785	19.845100
Pose 4	42.484135	14.802647	14.696232
Pose 5	55.329136	18.392832	17.324724
Pose 6	50.231300	17.183044	16.334940
Pose 7	63.489571	21.577450	19.732044
Pose 8	56.492722	15.799367	15.602816
Pose 9	49.745785	16.899014	16.646645
Pose 10	59.110195	20.544666	20.011791
Pose 11	63.949596	24.749189	24.188799
Pose 12	61.170898	23.697542	22.374830

Table D.4: Subject D - average RGB distance from GT for camera 1

	Original DM	Undeformed Model	Deformed Model
Pose 1	43.476494	25.316141	24.021002
Pose 2	51.153233	28.948597	28.839231
Pose 3	47.450253	28.295967	27.393982
Pose 4	47.109940	25.719717	24.751745
Pose 5	37.602493	20.756750	20.176121
Pose 6	39.852085	16.380753	15.360172
Pose 7	41.081612	19.345428	18.027454
Pose 8	43.232914	25.709339	24.522863
Pose 9	48.398876	21.491117	21.181973
Pose 10	50.957516	29.612055	28.955921
Pose 11	44.122429	24.608479	23.552496
Pose 12	34.671280	14.987300	14.062318

Table D.5: Subject D - average RGB distance from GT for camera 3

	Original DM	Undeformed Model	Deformed Model
Pose 1	62.009930	30.484936	28.777449
Pose 2	55.104713	29.272455	29.541388
Pose 3	53.936695	28.395403	25.711342
Pose 4	54.161472	27.852018	28.243841
Pose 5	53.253143	28.682835	28.533949
Pose 6	53.156864	22.597343	22.074007
Pose 7	57.084106	25.912823	24.462458
Pose 8	56.339256	28.859148	28.323353
Pose 9	54.582863	26.520191	26.238401
Pose 10	52.055447	25.740263	24.617180
Pose 11	55.406841	22.656012	21.721025
Pose 12	40.565155	16.271175	15.427202

Table D.6: Subject D - average RGB distance from GT for camera 4

	Original DM	Undeformed Model	Deformed Model
Pose 1	51.147377	24.230742	22.788300
Pose 2	47.028877	26.367048	26.299725
Pose 3	44.929386	26.834433	24.171257
Pose 4	46.652874	25.210871	25.236000
Pose 5	43.963921	22.704746	21.666199
Pose 6	42.066616	14.747580	14.079642
Pose 7	45.233456	20.649014	19.469257
Pose 8	44.694801	25.556385	25.287367
Pose 9	46.377396	20.907297	20.989653
Pose 10	45.392723	22.053635	21.267849
Pose 11	50.600250	25.424776	25.262171
Pose 12	34.705750	12.717072	11.815187

Table D.7: Subject G - average RGB distance from GT for camera 1

	Original DM	Undeformed Model	Deformed Model
Pose 1	67.331581	29.295992	27.352051
Pose 2	73.039299	33.743046	33.045753
Pose 3	71.491241	36.279533	35.255360
Pose 4	61.214146	33.276581	30.226803
Pose 5	62.882812	29.251049	27.724245
Pose 6	59.368855	15.991247	15.940762
Pose 7	59.066441	25.186464	24.155813
Pose 8	65.906532	19.695986	19.148224
Pose 9	71.673660	31.476734	28.583054
Pose 10	66.597855	26.139074	24.045439
Pose 11	61.658504	27.323616	26.907904
Pose 12	60.963768	19.900568	18.879946
Pose 13	60.582012	28.622744	28.308945
Pose 14	86.416534	35.190060	34.665215
Pose 15	78.686363	33.358498	32.700809

Table D.8: Subject G - average RGB distance from GT for camera 3

	Original DM	Undeformed Model	Deformed Model
Pose 1	94.961449	32.261337	30.058060
Pose 2	81.175446	32.703899	31.325933
Pose 3	89.875961	32.719181	31.853958
Pose 4	83.965797	34.390949	32.526886
Pose 5	92.279541	32.584915	31.201973
Pose 6	73.280769	16.680302	15.869398
Pose 7	77.421364	22.437611	21.620115
Pose 8	84.268227	25.197594	23.077227
Pose 9	86.185989	36.419487	33.531635
Pose 10	76.352364	28.828327	26.709644
Pose 11	87.819611	35.118511	32.659035
Pose 12	77.356819	25.359468	22.755095
Pose 13	82.377640	28.577803	28.256517
Pose 14	91.360680	35.230103	34.768784
Pose 15	91.188179	41.227070	40.343407

Table D.9: Subject G - average RGB distance from GT for camera 4

	Original DM	Undeformed Model	Deformed Model
Pose 1	76.101585	29.159929	27.554693
Pose 2	61.350391	30.280273	28.976038
Pose 3	70.460091	28.722569	27.519636
Pose 4	72.455185	34.488205	32.533405
Pose 5	76.986572	31.035990	30.187012
Pose 6	59.820431	14.473685	13.623223
Pose 7	64.379463	23.925940	23.209015
Pose 8	65.754158	20.510227	19.869896
Pose 9	69.925735	29.276943	26.623911
Pose 10	63.204739	24.436129	22.487394
Pose 11	70.971626	32.334068	30.330544
Pose 12	61.515839	22.671436	20.846895
Pose 13	71.413666	32.223949	31.946318
Pose 14	82.540413	33.656895	33.153072
Pose 15	76.525017	34.946419	34.392899

Bibliography

[1] E.H. Adelson, C.H. Anderson, J.R. Bergen, P.J. Burt, and J.M. Ogden. Pyramid methods in image processing. *RCA Engineer*, 29(6):33–41, 1984.

[2] E.H. Adelson and J.R. Bergen. *The plenoptic function and the elements of early vision*, section 1. MIT Press, Editors: Michael Landy and J. Anthony Movshon, 1991.

[3] A. Agarwal and B. Triggs. Tracking articulated motion using a mixture of autoregressive models. In *Proceedings of the 8th European Conference on Computer Vision*, pages 54 – 65, 2004.

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

[5] J.K. Aggarwal and Q. Cai. Human motion analysis : A review. *Computer Vision and Image Understanding*, 73(3):428 – 440, 1999.

[6] J.K. Aggarwal, Q. Cai, W. Liao, and B. Sabata. Articulated and elastic non-rigid motion: a review. In *Proceedings of the Workshop on Motion of Non-Rigid and Articulated Objects*, pages 2 – 14, 1994.

[7] J.K. Aggarwal and S. Park. Human motion: modeling and recognition of actions and interactions. In *Proceedings of the Second International Symposium on 3D Data Processing, Visualization and Transmission*, pages 640 – 647, 2004.

[8] M. Agrawal and K. Konolige. Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *International Conference on Pattern Recognition*, pages 1063 – 1068, 2006.

[9] A.I.M.A.T.S.H.A.P.E. Advanced and innovative models and tools for the development of semantic-based systems for handling, acquiring, and processing knowledge embedded in multidimensional digital objects. `http://www.aim-at-shape.net`, 2007 (accessed 7. August 2007).

[10] B. Allen, B. Curless, and Z. Popović. Articulated body deformation from range scan data. *ACM Transactions on Graphics (ACM SIGGRAPH 2002)*, 21(3):612 – 619, 2002.

[11] D. Anguelov, P. Srinivasan, D. Koller, Thrun S., J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. *ACM Transactions on Graphics*, 24(3):408 – 416, 2005.

[12] Roland ASD. Lpx-60/600 3D laser scanners. www.rolanddga.com/asd/products/scanners/LPX600, April 2008.

[13] N. Atzpadin, P. Kauff, and O. Schreer. Stereo analysis by hybrid recursive matching for real-time immersive video conferencing. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(3):321 – 334, 2004.

[14] N. Atzpadin and J. Mulligan. *3D Videocommunication*, chapter 7. John Wiley and Sons, Editors: Oliver Schreer, Peter Kauff and Thomas Sikora, 2005.

[15] S. Avidan and A. Shashua. Novel view synthesis in tensor space. In *Proceedings of CVPR 1997*, pages 1034 – 1040, 1997.

[16] Blue sky studios. `http://www.blueskystudios.com`, 2007 (accessed 7. August 2007).

[17] H. Baker and T. Binford. Depth from edge and intensity based stereo. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 631 – 636, 1981.

[18] H. H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, J. MacCormick, W. B. Culbertson, and T. Malzbender. Computation and performance issues in coliseum, an immersive videoconferencing system. In *ACM International Conference on Multimedia*, pages 470 – 479, 2003.

[19] A. O. Balan, L. Sigal, M. J. Black, J. E. Davis, and H. W. Haussecker. Detailed human shape and pose from images. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[20] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice Hall, 1982.

[21] C. Barron and I. Kakadiaris. Estimating anthropometry and pose from a single uncalibrated image. *Computer Vision and Image Understanding*, 81, 2001.

[22] R.H. Bartels, J.C. Beatty, and B.A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.

[23] L. Bo, C. Sminchisescu, A. Kanaujia, and D. Metaxas. Fast algorithms for large scale conditional 3d prediction. In *Proceedings IEEE Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[24] A.F. Bobick and S.S. Intille. Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200, 1999.

[25] J.-Y. Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/. camera calibration.

[26] D.J. Braunegg. Stereo feature matching in disparity space. In *Proceedings IEEE International Conference on Robotics and Automation*, volume 2, pages 796 – 803, 1990.

[27] P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532 – 540, 1983.

[28] H. Buxton. Learning and understanding dynamic scene activity: a review. *Image and Vision Computing*, 21(1):125 – 136, 2003.

[29] W.A.P. Buxton. Telepresence: integrating shared task and person spaces. In *Proceedings of Graphics Interface*, pages 123 – 129, 1992.

[30] F. Caillette, A. Galata, and T. Howard. Real-time 3-d human body tracking using learnt models of behaviour. *Computer Vision and Image Understanding*, 109(2):112–125, 2008.

[31] J. Carranza, C. Theobalt, M.A. Magnor, and H.P. Seidel. Free-viewpoint video of human actors. In *Proceedings of ACM SIGGRAPH*, pages 569 – 577, 2003.

[32] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10, 1978.

[33] C. Cedras and M. Shah. Motion-based recognition: a survey. *Image and Vision Computing*, 13(2):129 – 155, 1995.

[34] S.E. Chen and L. Williams. View interpolation for image synthesis. In *Proceedings of SIGGRAPH 1993*, pages 279 – 288, 1993.

[35] K.-S. D. Cheng, W. Wang, H. Qin, Wong K.-Y. K., H. Yang, and Y. Liu. Fitting subdivision surfaces to unorganised point data using sdm. In *Proceedings of Pacific Conference on Computer Graphics and Applications*, pages 16 – 24, 2004.

[36] K.-S. D. Cheng, W. Wang, H. Qin, K. Y. K. Wong, H. Yang, and Y. Liu. Design and analysis of optimization methods for subdivision surface fitting. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):878 – 890, 2007.

[37] S. Y. Cheng and M. M. Trivedi. Articulated human body pose inference from voxel data using a kinematically constrained gaussian mixture model. In *Proceedings of EHuM2: 2-nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation, CVPR Workshop*, 2007.

[38] G.K.M. Cheung, S. Baker, and T. Kanade. Shape from silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *Proceedings of IEEE Computer Vision and Pattern Recognition, Volume I*, pages 77 – 84, 2003.

[39] G.K.M. Cheung, S. Baker, and T. Kanade. Visual hull alignment and refinement across time: A 3d reconstruction algorithm combining shape-from-silhouette with stereo. In *Proceedings of IEEE Computer Vision and Pattern Recognition, Volume II*, pages 375 – 382, 2003.

[40] C.-W. Chun, O.C. Jenkins, and M.J. Matarič. Markerless kinematic model and motion capture from volume sequences. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 16 – 22, 2003.

[41] K. Connor and I. Reid. Novel view specification and synthesis. In *Proceedings of the 13th British Machine Vision Conference*, 2002.

[42] K. Connor and I. Reid. A multiple view layered representation for dynamic novel view synthesis. In *Proceedings of the 14th British Machine Vision Conference*, 2003.

[43] A. Criminisi, J. Shotton, A. Blake, C. Rother, and P.H.S. Torr. Efficient dense stereo with occlusions for new view-synthesis by four-state dynamic programming. *International Journal of Computer Vision*, 71(1):89–110, 2007.

[44] Cyberware. Whole body x 3d scanner. www.cyberware.com/products/scanners/wbx.html, April 2008.

[45] K. Daniilidis, J. Mulligan, R. McKendall, D. Schmid, G. Kamberova, and R. Bajcsy. *Confluence of Computer Vision and Computer Graphics*, chapter 14, pages 253 – 265. NATO Science Series, Editors: Aleš Leonardis, Franc Solina and Ruzena Bajcsy, 2000.

[46] C. de Boor. *A Practical Guide to Splines*. Springer Verlag New York, 1978.

[47] D. Demirdjian and T. Darrell. Using multiple-hypothesis disparity maps and image velocity for 3-d motion estimation. *International Journal of Computer Vision*, 47(1/2/3):219–228, 2002.

[48] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *Proceedings of ACM SIGGRAPH*, pages 85 – 94, 1998.

[49] K.G. Derpanis and P. Chang. Closed-form linear solution to motion estimation in disparity space. In *Procedings of the IEEE Intelligent Vehicles Symposium*, pages 268 – 275, 2006.

[50] J. Deutscher, A. Blake, and I Reid. Articulated body motion capture by annealed particle filtering. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2000*, volume 2, pages 21 – 26, 2000.

[51] J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61(2):185 – 205, 2005.

[52] D. Doo and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10, 1978.

[53] O. Drummond and R. Cipolla. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *Proceedings of International Conference on Computer Vision*, pages 315 – 320, 2001.

[54] R. C. Eberhart and Y. H. Shi. Guest editorial special issue on particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3), 2004.

[55] G. Farin. *Curves and Surfaces for CAGD*. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann, 2002.

[56] O. Faugeras. *Three-Dimensional Computer Vision, A Geometric Viewpoint*. The MIT Press, 1993.

[57] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[58] A. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. *International Journal of Computer Vision*, 63(2):141–151, 2005.

[59] R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, 2006.

[60] D.A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[61] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[62] D.M. Gavrila. Visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73, 1999.

[63] S. J. Gibbs, C. Arapis, and C. J. Breiteneder. Teleport towards immersive copresence. *Multimedia Systems*, 7(3):214 – 221, 1999.

[64] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Prentice Hall, 2nd international edition, 2002.

[65] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The lumigraph. In *Proceedings of SIGGRAPH 1996*, pages 43 – 54, 1996.

[66] W.E.L. Grimson. *From Images to Surfaces: A Computational Study of the Human Early Visual System*. The MIT Press Classics. MIT Press, 1981.

[67] H. Gross, J. Richarz, S. Mueller, A. Scheidig, and C. Martin. Probabilistic multimodal people tracker and monocular pointing pose estimator for visual instruction of mobile robot assistants. In *International Joint Conference on Neural Networks, 2006*, pages 4209 – 4217, 2006.

[68] M. Gross, S. Wuermlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, Vande Moere A., and O. Staadt. blue-c: A spatially immersive display and 3d video portal for telepresence. In *Proceedings of ACM SIGGRAPH 2003*, pages 819 – 827, 2003.

[69] Y. Guo, G. Xu, and S. Tsuji. Understanding human motion patterns. In *Proceedings of the 12th IAPR International Conference on Computer Vision Image Processing*, pages 325 – 329, 1994.

[70] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2003.

[71] H. Hattori and N. Takeda. Dense stereo matching in restricted disparity space. In *Intelligent Vehicles Symposium*, pages 118 – 123, 2005.

[72] L. Herda, R. Urtasun, and P. Fua. Hierarchical implicit surface joint limits for human body tracking. *Computer Vision and Image Understanding*, 99(2):189 – 209, 2005.

[73] A. Hilton, D. Beresford, T. Gentils, R.J. Smith, W. Sun, and J. Illingworth. Whole-body modelling of people from multi-view images to populate virtual worlds. *The Visual Computer*, 16(7):411 – 436, 2000.

[74] L. Hong and G. Chen. Segment-based stereo matching using graph cuts. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 74 – 81, 2004.

[75] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of ACM SIGGRAPH*, pages 295 – 302, 1994.

[76] S. Hou, A. Galata, F. Caillette, N. Thacker, and P. Bromiley. Real-time body tracking using a gaussian process latent variable model. In *Proceedings of 11th International Conference on Computer Vision*, pages 1 – 8, 2007.

[77] H.-H. Hsu, S.-W Hsieh, W.-C. Chen, C.-J. Chen, and C.-Y. Yang. Motion analysis for the standing long jump. In *26th IEEE International Conference on Distributed Computing Systems Workshops*, page 47, 2006.

[78] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviours. *Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews*, 34(3):334 – 352, 2004.

[79] Z. Husz and A. M. Wallace. Evaluation of a hierarchical partitioned particle filter with action primitives. In *EHuM2: 2-nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation, CVPR Workshop*, 2007.

[80] S. Ilić. Using subdivision surfaces for 3-d reconstruction from noisy data. In *Proceedings of Deform'06 - Workshop on Image Registration in Deformable Environments, in conjunction with BMVC 2006*, pages 1 – 10, 2006.

[81] S.S. Intille and A.F. Bobick. Disparity-space images and large occlusion stereo. In *Proceedings of European Conference on Computer Vision*, volume 2, pages 179 – 186, 1994.

[82] F. Isgrò, E. Trucco, P. Kauff, and O. Schreer. 3-d image processing in the future of immersive media. *Transactions on Circuits and Systems for Video Technology*, 14(3):288 – 303, 2004.

[83] W. K. Jeong and C. H. Kim. Direct reconstruction of displaced subdivision surface from unorganised points. In *Proceedings of Pacific Conference on Computer Graphics and Applications*, pages 160 – 169, 2001.

[84] T. Kanade, P. Rander, and P.J. Narayanan. Virtualized reality: Contructing virtual worlds from real scenes. *IEEE Multimedia*, 4(1):34 – 47, 1997.

[85] J. Kannala and S. S. Brandt. Quasi-dense wide baseline matching using match propagation. In *Proceedings of CVPR 2007*, pages 1 – 8, 2007.

[86] P. Kauff, N. Atzpadin, C. Fehn, M. Mller, O. Schreer, A. Smolic, and R. Tanger. Depth map creation and image-based rendering for advanced 3dtv services providing interoperability and scalability. *Elsevier Signal Processing: Image Communication*, 22(2):217 – 234, 2007.

[87] P. Kauff and O. Schreer. *3D Videocommunication*, chapter 5. John Wiley and Sons, Editors: Oliver Schreer, Peter Kauff and Thomas Sikora, 2005.

[88] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[89] J. Kennedy, R. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[90] J. Lanier, A. Sadagic, D. Ondayko, and M. Brown. Us national tele-immersion initiative. http://www.advanced.org/teleimmersion2.html, 1997. Accessed on 15. July 2008.

[91] S. Laveau and O. Faugeras. 3D scene representation as a collection of images. In *Proceedings of the 12th International Conference on Pattern Recognition*, pages 689–691, 1994.

[92] M. W. Lee and I. Cohen. A model-based approach for estimating human 3d poses in static images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(6):905 – 916, 2006.

[93] B. J. Lei and E. A. Hendriks. A real-time realization of geometrical valid view synthesis for tele-conferencing with viewpoint adaptation. In *Proceedings of the First International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'02)*, pages 327 – 336, 2002.

[94] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH 1996*, pages 31 – 42, 1996.

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

[96] N. Litke, A. Levin, and P. Schroeder. Fitting subdivision surfaces. In *Proceedings of IEEE Visualization 2001*, pages 319 – 324, 2001.

[97] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, 1987.

[98] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 20(2):91 – 110, 2004.

[99] W. Ma. Subdivision surfaces for CAD - an overview. *Computer-Aided Design*, 37(7):693 – 709, 2005.

[100] W. Ma and N. Zhao. Catmull-clark surface fitting for reverse engineering applications. In *Proceedings of Geometric Modeling and Processing 2000*, pages 274 – 283, 2000.

[101] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface quality hand tracking. In *European Conference on Computer Vision, Part II*, pages 3 – 19, 2000.

[102] J. Maillot and J. Stam. A unified subdivision scheme for polygonal modeling. *Computer Graphics Forum*, 20(3):471–479, 2001.

[103] D. Marr and T. Poggio. A computational theory of human stereo vision. In *Proc. Royal Society of London*, pages 301 – 328, 1979.

[104] D. Martinec and T. Pajdla. Structure from many perspective images with occlusions. In *Proceedings of the ECCV'02*, pages 355 – 369, 2002.

[105] B. Matei and P. Meer. A general method for errors-in-variables problems in computer vision. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 18 – 25, 2000.

[106] W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 115 – 126, 2001.

[107] D. McLeod, U. Neumann, C.L. Nikias, and A.A. Sawchuk. Integrated media systems. *IEEE Signal Processing Magazine*, 16(1):33–43, 1999.

[108] L. McMillan and G. Bishop. Plenoptic modeling: an image-based rendering system. In *Proceedings of SIGGRAPH 1995*, pages 39 – 46, 1995.

[109] A. Micilotta, E. Ong, and R. Bowden. Detection and tracking of humans by probabilistic body part assembly. In *Proceedings of British Machine Vision Conference*, pages 429 – 438, 2005.

[110] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *International Journal on Computer Vision*, 53(3):199 – 223, 2003.

[111] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615 – 1630, 2005.

[112] Konica Minolta. Vivid 910. www.konicaminolta.com/instruments/products/3d/index.html, April 2008.

[113] T.B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81, 2001.

[114] T.B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3), 2006.

[115] Robyn Owens. Geometric transformations. CVonline: On-Line Compendium of Computer Vision, http://homepages.inf.ed.ac.uk/rbf/CVonline/, 22 July 2007.

[116] V. Parameswaran and R. Chellappa. View independent human body pose estimation from a single perspective image. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages II–16 – II–22, 2004.

[117] R. Plaenkers and P. Fua. Articulated soft objects for multi-view shape and motion capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10), 2003.

[118] R. Poli. An analysis of publications on particle swarm optimisation applications. Technical Report CSM-649, University of Essex, Department of Computer Science, November 2007.

[119] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, 2007.

[120] R. Poli, J. Kennedy, T. Blackwell, and A. Freitas. Editorial for particle swarms: The second decade. *Journal of Artificial Evolution and Applications*, 1(1):1–3, 2008.

[121] R. Poppe. ”vision-based human motion analysis: An overview”. *Computer Vision and Image Understanding*, pages 4–18, 2007.

[122] W. H. Press, S. A. Teukolsky, W.T. Vetterling, and B.P Flannery. *Numerical Recipes in C++*. Cambridge University Press, 2002.

[123] H. Qin, C. Mandal, and B.C. Vemuri. Dynamic Catmull-Clark Subdivision Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):215 – 229, 1998.

[124] Poppe R. Evaluating example-based pose estimation: Experiments on the humaneva sets. In *Proceedings of EHuM2: 2-nd Workshop on Evaluation of Articulated Human Motion and Pose Estimation, CVPR Workshop*, 2007.

[125] Poppe R., Heylen D., Nijholt A., and Poel M. Towards real-time body pose estimation for presenters in meeting environments. In *Proceedings of the 13-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2005*, 2005.

[126] T. Roberts, S. J. McKenna, and I. W. Ricketts. Human pose estimation using partial configurations and probabilistic regions. *International Journal of Computer Vision*, 73(3):285 – 306, 2007.

[127] C. Robertson and E. Trucco. Human body posture via hierarchical evolutionary optimization. In *British Machine Vision Conference 2006*, pages 999 – 1008, 2006.

[128] C. Robertson, E. Trucco, and S. Ivekovic. Dynamic body posture tracking using evolutionary optimisation. *Electronic Letters*, 41:1370 – 1371, 2005.

[129] N. Roma, J. Santos-Victor, and J.A.B. Tome. A comparative analysis of cross-correlation matching algorithms using a pyramidal resolution approach. In *Proceedings of the Second Workshop on Empirical Evaluation Methods in Computer Vision in conjunction with the 6th European Conference on Computer Vision*, pages 1–23, 2000.

[130] R. Rosales, M. Siddiqui, J. Alon, and S. Sclaroff. Estimating 3D body pose using uncalibrated cameras. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages I–821 – I–827, 2001.

[131] P. Salamon, P. Sibani, and R. Frost. *Facts, Conjectures and Improvements for Simulated Annealing*. SIAM Monographs on Mathematical Modeling and Computation, 2002.

[132] D. Scharstein and R. Szeliski. Stereo matching with non-linear diffusion. *International Journal of Computer Vision*, 28(2):155 – 174, 1998.

[133] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7 – 42, 2002.

[134] D. Scharstein and R. Szeliski. Middlebury stereo vision page. `http://vision.middlebury.edu/stereo/`, 2008 (accessed 8. July 2008).

[135] O. Schreer, N. Brandenburg, S. Askar, and P. Kauff. Hybrid recursive matching and segmentation-based postprocessing in real-time immersive video conferencing. In *Proceedings of Vision, Modeling and Visualization 2001*, 2001.

[136] O. Schreer and P. Sheppard. Virtue - the step towards immersive tele-presence in virtual video conference systems. In *Proceedings of eWorks 2000*, 2000.

[137] P. Schröder. Subdivision as a fundamental building block of digital geometry processing algorithms. *Journal of Computational and Applied Mathematics*, 149(1):207 – 219, 2002.

[138] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, and A. D. George. Parallel global optimization with the particle swarm algorithm. *International Journal for Numerical Methods in Engineering*, 61(13), 2004.

[139] S.M. Seitz and C.M. Dyer. View morphing. In *Proceedings of SIGGRAPH 1996*, pages 21 – 30, 1996.

[140] A. Sellen, W. Buxton, and J. Arnott. Using spatial cues to improve videoconferencing. In *Proceedings of CHI 1992*, pages 651 – 652, 1992.

[141] A. Shahrokni, O. Woodford, and I. Reid. Temporal priors for novel video synthesis. In *Proceedings of ACCV*, pages 601–610, 2007.

[142] Y. Shi. Particle swarm optimization. *IEEE Neural Networks Society Magazine (Feature Article)*, pages 8 – 13, February 2004.

[143] Y. H. Shi and R. C. Eberhart. A modified particle swarm optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 69 – 73, 1998.

[144] K. Shoji, A. Mito, and F. Toyama. Pose estimation of a 2d articulated object from its silhouette using a ga. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 3, pages 713 – 717, 2000.

[145] H. Sidenbladh, M.J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *Proceedings of European Conference on Computer Vision*, pages 784 – 800, 2002.

[146] L. Sigal. Humaneva database of multi-view video sequences with ground truth for articulated body tracking. http://vision.cs.brown.edu/humaneva/, July 2008.

[147] L. Sigal, A. Balan, and M.J. Black. Combined discriminative and generative articulated pose and non-rigid shape estimation. In *Proceedings of NIPS 2007*, 2007.

[148] L. Sigal, S. Bhatia, S. Roth, M. J. Black, and M. Isard. Tracking loose-limbed people. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 421 – 428, 2004.

[149] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research*, 22(6):371 – 391, 2003.

[150] J. Starck and A. Hilton. Model-based multiple view reconstruction of people. In *IEEE International Conference on Computer Vision*, pages 915 – 922, 2003.

[151] J. Starck, G. Miller, and A. Hilton. Video-based character animation. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 49 – 60, 2005.

[152] N. Stefanov, A. Galata, and R. Hubbold. Real-time hand tracker using variable-length markov models of behaviour. *Computer Vision and Image Understanding*, 108(1-2):98 – 115, 2007.

[153] G. Strang. *Linear algebra and its applications*. Harcourt Brace Jovanovich College publishers, 2nd edition, 1988.

[154] C. Strecha, R. Fransens, and L. Van Gool. Wide-baseline stereo from multiple views: a probabilistic account. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2004*, pages 552 – 559, 2004.

[155] C. Strecha, T. Tuytelaars, and L. Van Gool. Dense matching of multiple wide-baseline views. In *Proceedings of IEEE International Conference on Computer Vision 2003*, pages 1194 – 1201, 2003.

[156] Pixar Animation Studios. http://www.pixar.com.

[157] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proceedings of ECCV 1996*, pages 709 – 720, 1996.

[158] H. Suzuki, S. Takeuchi, F. Kimura, and T. Kanai. Subdivision surface fitting to a range of points. In *Proceedings of IEEE Computer Society Pacific Conference on Computer Graphics and Applications*, pages 158 – 167, 1999.

[159] T. Svoboda, D. Martinec, and T. Pajdla. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407 – 422, 2005.

[160] R. Szeliski and P. Golland. Stereo matching with transparency and matting. *International Journal of Computer Vision*, 32(1):45–61, 1999.

[161] R. Szeliski and D. Scharstein. Sampling the disparity space image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):419 – 425, 2004.

[162] R. Tanger, P. Kauff, and O. Schreer. Immersive meeting point (im.point) - an approach towards immersive media portals. In *Proceedings of Pacific-Rim Conference on Multimedia*, pages 89–96, 2004.

[163] C.J. Taylor. Reconstruction of articulated objects from point correspondences in a single image. *Computer Vision and Image Understanding*, 80(3):349 – 363, 2000.

[164] TELEPORTEC. The ultimate in global face-to-face communication. `http://www.teleportec.com`, 2007 (accessed 7. August 2007).

[165] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[166] E. Trucco, F. Isgro, and F. Bracchi. Plane detection in disparity space. In *Proceedings of International Conference on Visual Information Engineering*, pages 73 – 76, 2003.

[167] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[168] C.-J. Tsai and A. K. Katsaggelos. Dense disparity estimation with a divide-and-conquer disparity space image technique. *IEEE Transactions on Multimedia*, 1(1):18 – 29, 1999.

[169] R.Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf cameras and lenses. *Journal of Robotics and Automation*, 3(4):323 – 344, 1987.

[170] R. Urtasun and T. Darrell. Sparse probabilistic regression for activity-independent human pose inference. In *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 1 – 8, 2008.

[171] R. Urtasun, D. J. Fleet, and P. Fua. 3D people tracking with gaussian process dynamical models. In *Proceedings of Computer Vision and Pattern Recognition*, pages 238 – 245, 2006.

[172] R. Urtasun, D. J. Fleet, and P. Fua. Temporal motion models for monocular and multiview 3D human body tracking. *Computer Vision and Image Understanding*, 104(2):157 – 177, 2006.

[173] L. Wang, W.M. Hu, and T.N. Tan. Recent developments in human motion analysis. *Pattern Recognition*, 36(3):585 – 601, 2003.

[174] J. Warren and S. Schaefer. A factored approach to subdivision surfaces. *Computer Graphics and Applications*, 24(3):74 – 81, 2004.

[175] J. Warren and H. Weimer. *Subdivision Methods for Geometric Design*. Morgan Kaufmann Publishers, 2002.

[176] M. Waschbüsch, S. Würmlin, D. Cotting, and M. Gross. Point-sampled 3D video of real-world scenes. *Image Communication*, 22(2):203 – 216, 2007.

[177] O. J. Woodford, I. D. Reid, and A. W. Fitzgibbon. Efficient new-view synthesis using pairwise dictionary priors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 – 8, 2007.

[178] O. J. Woodford, I. D. Reid, P. H. S. Torr, and A. W. Fitzgibbon. Fields of experts for image-based rendering. In *Proceedings of British Machine Vision Conference*, volume 3, pages 1119 – 1128, 2006.

[179] M. Xie. *Fundamentals of Robotics: Linking Perception to Action*, volume 54 of *Series in Machine Perception and Artificial Intelligence*. World Scientific Publishing, 2003.

[180] Y. Yang, A. Yuille, and J. Lu. Local, global, and multilevel stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 274–279, 1993.

[181] Z. Ye and Z.-Q. Liu. Genetic condensation for motion tracking. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, volume 9, pages 5542 – 5547, 2005.

[182] Cha Zhang and Tsuhan Chen. A survey on image-based rendering - representation, sampling and compression. *Elsevier Signal Processing: Image Communication*, 19(1):1 – 28, 2004.

[183] Z. Zhang. A flexible new technique for camera calibration. *Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330 – 1334, 2000.

[184] C.L. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):675 – 684, 2000.

[185] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *ACM Transactions on Graphics 23(3), Proceedings of SIGGRAPH 2004*, pages 600 – 608, 2004.

[186] D. Zorin, P. Schroeder, T. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. Subdivision for modeling and animation, course notes. In *SIGGRAPH 2000*, 2000.