Markerless Pose Tracking of a Human Subject

by

Neil Hendry

Submitted in fulfilment of the academic requirements
for the degree of Master of Science in Engineering
in the School of Electrical, Electronic and Computer Engineering at the
University of KwaZulu–Natal, Durban, South Africa

Supervisor:
Mr. Bashan Naidoo

2012

To my parents, Keith and Gill,
for all the sacrifices they have made
and opportunities they have given me.

Preface

I, Neil Hendry, declare that

1. The research reported in this dissertation, except where otherwise indicated, is my original work.
2. This dissertation has not been submitted for any degree or examination at any other university.
3. This dissertation does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This dissertation does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
   (a) their words have been re-written but the general information attributed to them has been referenced;
   (b) where their exact words have been used, their writing has been placed inside quotation marks, and referenced.
5. Where I have reproduced a publication of which I am an author, co-author or editor, I have indicated in detail which part of the publication was actually written by me alone and have fully referenced such publications.
6. This dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References sections.

Signed:

As the candidate's Supervisor I agree/do not agree to the submission of this thesis.

Signed:

Name:

Acknowledgements

I would like to thank my supervisor, Mr Bashan Naidoo, for the guidance and advice he has given me during my research and for always being willing to assist in a friendly manner whenever a problem arose. I am very grateful for the time and effort he spent proof-reading my conference and journal papers and the invaluable corrections and suggestions he gave.

I would also like to thank my sponsors, SAAB Grintek Communications and Armscor, for their financial support and interest in my project. In particular I would like to mention Mrs Franzette Vorster, Mr Danny Ignatov, Mr Preshon Jeebodh and Mr Francois van der Merwe for their involvement in this sponsorship.

The financial assistance of the National Research Foundation (NRF) towards this research is also gratefully acknowledged. Opinions expressed and conclusions arrived at are those of the author and are not necessarily to be attributed to the NRF.

A big thanks must go to my mother, Gill, for her help with the statistical analysis of the results and for always being willing to 'lend an ear' when things were not working as expected. The process of talking about the problem out loud and going through the code helped to solve many errors. I am also very grateful for all her time spent checking my papers for grammar and spelling mistakes. Thank you also to my Aunt, Ethel Ross, for doing an amazing job proof reading my dissertation.

The time spent on my masters would not have been nearly as enjoyable without my fellow postgraduate students. I wish to thank my office-mate, Naren, for his friendship during this process and mutual help in solving problems. To all my 'Mat-Sci' friends, in particular Jono, Brett and Peter, thanks for the memorable lunch times and the always-available use of your toaster.

Lastly, thank you to my family, parents and sister, Liesl, for their encouragement and support during my masters and for allowing me to practise my talks on them; and to my close friends, thanks for always being interested in what I was doing.

Abstract

High capacity wireless and fixed-line broadband services have a relatively small footprint over South Africa's vast expanse. This results in many rural areas, as well as military communication when deployed, relying on low-bandwidth communication networks instead, making live video communication over these links impractical. Traditional and advanced data compression methods cannot produce the payload reduction required for video use over these bandwidths. Instead, a model-based vision system is used to address this problem. This is not video compression but rather image understanding and representation in the context of prior models of the observed object. Markerless human tracking and pose recovery are the specific interests of this research.

Markerless human pose tracking is a relatively new and growing field of image processing. It has many potential areas of application apart from low-bandwidth video communication, including the medical field, sporting arena, security and surveillance and human-machine interaction. As multimedia technologies continue to grow and improve, pose tracking systems have the potential to be used more and more. While a few markerless tracking devices are beginning to emerge, many currently available commercial motion capture systems require the use of a special suit and markers or sensors. This makes them very impractical for easy everyday, anywhere use. Current research in computer vision and image processing incorporates a significant focus on the development of markerless approaches to human motion capture.

This dissertation looks at a complete markerless human pose tracking system which can be split into four distinct but interlinking stages: the image capture, image processing, body model and optimisation stages. After video data from multiple camera views is captured, the processing stage extracts image cues such as silhouettes, 2-D edges and 3-D colour volumetric reconstruction. Following the basic principle of a model-based approach, a 24 degree-of-freedom superellipsoid body model is fitted to the observed image cue data. An objective function is used to measure the closeness of this match.

A number of different optimisation approaches are examined for use in refining and finding the best fitting body pose for each image frame. These approaches are all based around Stochastic Meta Descent (SMD) optimisation with SMD by itself, SMD in a

hierarchical approach, SMD with pose prediction and Smart Particle Filtering, SMD inside a particle filter framework, all explored.

The performance of the system with the various optimisation approaches is tested using the HumanEvaII datasets. These datasets contain a number of different subjects performing a variety of actions while wearing ordinary clothes. They contain marker-based ground-truth data obtained using a ViconPeak motion capture system. This allows a relative error measurement of the predicted poses to be calculated. With its robustness to clutter and occlusion, the Smart Particle Filter approach is shown to give the best results.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large areas of South Africa are only covered by low-bandwidth systems. This is mainly confined to rural areas where fixed-line and high speed HSDPA mobile networks are not available. Many military applications also use low-bandwidth systems due to their robustness, flexibility and ease-of-use, especially when deploying to remote rural areas. This makes live video communication over these links very impractical, with traditional and advanced compression methods rarely able to provide the bandwidth reduction that is required.

Therefore, a model-based vision system, which does not use data compression but rather image understanding and representation in the context of prior models of the observed object, is used to address this problem. This is achieved with low-bandwidth modelling and tracking techniques for objects in the video scene and allows model information about the video sequence to be sent instead of the actual video itself. The information can then be used on the receiver side as required and the scene, if needed, can be virtually reconstructed. While this is not photo-realistic, there are many cases when the knowledge of what is occurring in a scene is more important than photo-realism. The cost of greatly reducing the amount of data required to be sent is that more processing power is required — both at the video capture and video display ends. However, with the rapid increase in processing power of new devices, this is less of a problem than bandwidth. Markerless human tracking and pose recovery are the specific interests of this study.

1.1 Bandwidth Reduction

To get an idea of the bandwidth reduction achieved, a usual standard definition (640 x 480 pixels), 24-bit colour video running at 25 frames/sec with no compression has a data rate of about

$$24\frac{bits}{pixel} \times (640 \times 480)\frac{pixels}{frame} \times 25\frac{frames}{sec} = 175.8Mbps$$

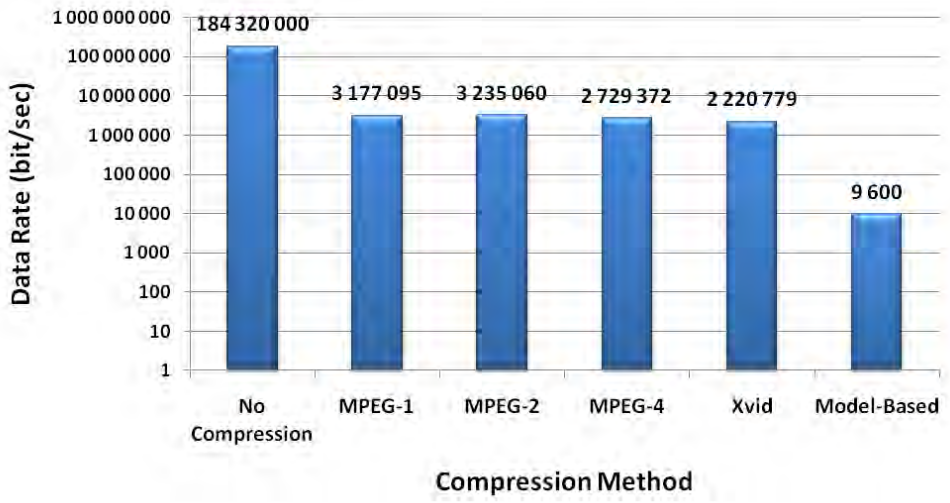


Figure 1.1: Graph comparing the bandwidth requirements for various video compression methods.

Various video compression techniques have an average compression ratio of 1:60 – 1:80 [1] which reduces this data rate to between 2 and 4 Mbps.

Transmission of model data, on the other hand, is very small. In this dissertation a 24-parameter human model is used which means that a data rate of only

$$16\frac{bits}{parameter} \times 24\frac{parameters}{frame} \times 25\frac{frames}{sec} = 9600 bits/sec$$

is required. This is a huge reduction and the comparison of these compression options is displayed in Figure 1.1 — note the logarithmic vertical scale. This can further be reduced by employing a scheme which only sends model update information when significant change or movement occurs.

1.2 Markerless Pose Tracking

Human pose tracking is not a new subject. It has been around for years with people wearing special suits containing markers and sensors which allow a sophisticated system to capture the subjects' movements. These techniques, used in many cinematic productions including *The Lord of the Rings*, *Avatar* and recently *Rise of the Planet of the Apes*, allow an actor's movements and facial expressions to be captured and used to animate their digital counterparts.

As multimedia technologies continue to grow and improve, pose tracking systems have the potential to be used more and more; however, not with the restrictions of a special suit. Current research in computer vision and image processing is leading to the development of many markerless approaches to human motion capture. Microsoft

[1] compression ratios tested using a few different video sequences but may vary slightly depending on video clip and/or codec settings used

is presently using basic markerless tracking in the form of their Kinect product. This device links with an Xbox 360 gaming console and allows a player to interactively control games through physical movements which are captured by the Kinect's camera and sensors.

As with all tracking, markerless human pose tracking requires the use of some technique to find the most likely pose given all available information. However, due to the high dimensional state space of a human body, finding a suitable optimisation or filtering method to estimate the pose can be difficult. This high dimensionality is a result of the body not being a rigid entity but rather consisting of numerous dependent articulated motions.

Traditionally the use of optimisation techniques such as the Kalman filter [99] and Condensation [42] algorithms were favoured. Powell's method [17, 24] was also frequently used, primarily since it does not require a gradient or derivative calculation. However, for non-linear objective functions it can take many iteration steps to converge on a solution and can thus be very slow.

Popular gradient-based techniques for pose tracking systems include the Levenberg-Marquardt [57, 61] and Broyden-Fletcher-Goldfarb-Shanno [24] algorithms. These methods handle non-linear optimisation problems well but require the calculation of the first and second-order derivatives — which may not be available or easily calculated. A simple first-order Gradient Descent method has also been used [74, 87] but again shows poor convergence for non-linear problems.

Stochastic Meta Descent (SMD) is another method recently introduced to tracking as outlined in [11, 50]. It is based on gradient descent but is able to achieve faster convergence by using adaptive and parameter specific step size.

These optimisation approaches to tracking generally perform well in that they find the best fitting local model. Stochastic Meta-Descent optimisation, in particular, has been shown to perform well in high-dimensional spaces [11] as it is able to use fewer sample points than many other optimisation schemes due to its stochastic sampling. However, in non-linear spaces with many local optima, optimisation (including SMD) does not guarantee that the global optimum will be reached. This is compounded when clutter and occlusion is experienced.

Human pose tracking is a classic filtering problem since it requires an estimate of the state of the body pose, based on given observations, for each time-step or frame. There exist many methods to solve this sort of problem but particle filters, in particular, have become popular for solving human motion tracking problems. This is because they are easy to implement and do not assume that signal or observation probabilities follow a linear or Gaussian model — as is the case with many other filters.

Particle filters are specially designed to examine multiple hypotheses which improves the chance of the global optimum being reached. However, with high dimensional problems this is computationally very expensive as the number of particles used must drastically increase with the dimensionality to sufficiently cover the entire search space. To overcome this, either the dimensionality has to be reduced, or a scheme which uses fewer samples must be employed.

The advantages of both SMD optimisation and particle filtering can be combined to form a multiple hypotheses framework, in which SMD is wrapped. This makes use of the particle filter's ability to improve the likelihood of reaching the global minimum while the SMD optimisation handles the high dimensionality. This is known as a Smart Particle Filter (SPF) and was employed by Bray *et al.* [13] for articulated hand tracking. In this paper, the method is taken a step further and applied to full body markerless human pose tracking with some additional features from Gall *et al.*'s [29]'s interacting simulated annealing filter also included in the algorithm.

1.3 Application Possibilities

Apart from low-bandwidth video communication, there are other potential applications for the modelling and pose tracking of human beings. These include:

1. **Medical Field** A benefit would be to use it for diagnostic and preventative analysis of incorrect body motion and posture which can lead to future problems. Testing and refining prosthetic limps/joints is another area of medicine which could be assisted with this type of modelling.

2. **Sport** This could be especially useful for the modelling of sporting movements for performance analysis which would lead to the correction and improvement of the action (eg. swimming stroke, golf shot, cricket bowling action). In particular it could be used for the study of suspicious bowling actions in cricket. It would not require the player to go for special testing wearing markers but could instead be performed while he was playing an ordinary cricket match — thus greatly improving on the current method of testing.

3. **Surveillance** Once modelled, human actions could be automatically monitored to determine suspicious behaviour which might then alert appropriate authorities to monitor or deal with the culprit(s).

4. **Communication** Linked to the low-bandwidth video transmission, video cellphones can be used by the deaf to communicate via modelling and analysis of spoken words (lip-reading) and/or sign language, relatively cheaply. This type of modelling can also be used to improve HMI (Human Machine Interface) com-

munication systems — which is of course related to video game devices such as Microsoft's Kinect.

1.4 Contributions and Publications

Parts of this work have been presented at local conferences and published in the conference proceedings, including [36, 37]. Another paper is currently in the process of being reviewed for acceptance in the IEEE Transactions on Pattern Analysis and Machine Intelligence journal .

This thesis is based on the work by Kehl and Van Gool [50] but a number of modifications and improvements have been added.

1. A modular system has been implemented which provides a good basis for future study in this area. Any of the stages can very easily be removed or updated and new stages added, with quick and easy testing of the modifications.

2. Smart Particle Filtering has been introduced to the optimisation stage of the system. This still makes use of the SMD optimisation but implements it in a particle filter framework. Multiple hypotheses are thus examined providing robustness to the tracking system, especially in situations of high clutter and occlusion. This is similar to the work by Bray *et al.* [13], but extended to full human pose tracking.

3. The resampling section of the SPF has been adjusted to more closely follow the work of Gall *et al.* [29]. This improves the selection and representation of the states (or poses) to better cover the search space and include the more closely aligned poses for the next frame.

4. Testing is performed using the HumanEva datasets which have recently been used by numerous researches and become a kind of standard dataset for markerless pose tracking algorithm evaluation. Both image and ground-truth data are available for testing and evaluation purposes.

1.5 Outline of Dissertation

This dissertation starts off by examining the related work in Chapter 2. Model-based and model-free approaches are explored, with the main focus being on the model-based. Here various image processing options are examined under the three identified key components of body models, image descriptors and optimisation techniques.

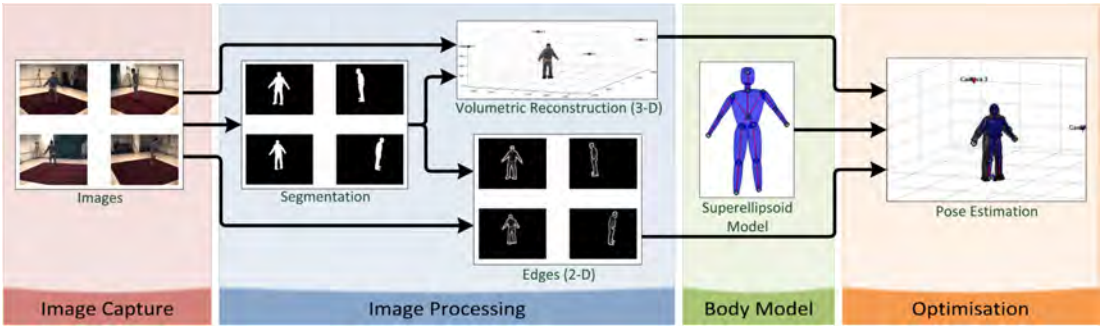


Figure 1.2: Overall functional diagram showing the 4 stages of the model-based human pose tracking system.

The human pose tracking system can be spit into four distinct but interlinked stages as depicted in Figure 1.2. These are the image capture, image processing, body model and optimisation stages and form the main body of the dissertation.

The first stage, Image Capture, is dealt with in Chapter 3. This pertains to the collection and input of image data to the system and looks at the epipolar geometry and camera calibration involved in the process. The HumanEva datasets used for testing are also introduced.

The image processing stage, presented in Chapter 4, converts the observed data into some usable form for pose tracking. In the chapter, segmentation, edge detection and volumetric reconstruction are the main focus with results from this processing given at the end. The segmentation is performed using an illumination invariant method with *a priori* probabilities, colour collinearity testing and a darkness compensation component added. An RGB colour space Difference Vector Edge Detector is used which helps reduce the unwanted detection of shadow edges. For the volumetric reconstruction a voxel-based visual hull method is implemented with a pre-computed look-up-table for improved processing performance.

Chapter 5 examines the body model used in the tracking system. A superellipsoid model controlled by an underlying 24 degrees-of-freedom skeleton is used. This allows the human shape to be reasonably approximated without excessive processing required. Basic kinematic constraints are also included to eliminate invalid poses.

In Chapter 6 the optimisation framework is presented. This involves various evolutions, initially introducing just SMD optimisation, then SMD with first a hierarchical approach and then with pose prediction added and finally Smart Particle Filtering — SMD inside a particle filter framework. In the SPF the SMD optimisation handles the high dimensionality of the human pose tracking while the particle filter improves the robustness by allowing multiple hypotheses to be examined at each frame.

Chapter 7 examines the tracking results for the different optimisation approaches. A couple of subjects performing a number of actions are used for this testing. The benefit of the SPF framework over the other optimisation approaches is shown, with the various results, graphs and images being discussed.

Lastly, in Chapter 8, final comments on the human pose tracking system are offered and a few options for future work and improvement are presented.

Appendix A outlines the contents of the attached CD which contains files and Matlab code related to this dissertation.

Chapter 2

Related Work

Tracking of a human can be classified in two ways: by identifying the position of a human in some scene, or determining the motion of a human and his various body parts — known as the body pose. The simple 'blob-tracking' of a human is a relatively simple task to accomplish and, except in a few specific cases, does not provide any meaningful information to a system. The tracking of interest here is pose tracking where specific body points (joints) and regions (forearm, upper arm, etc.) are tracked. This allows the complete movement and posture of a human to be recovered and modelled.

While many pose tracking methods use a model-based approach (Section 2.1), where a body model is available and fitted to the observed information, there has recently been more research into the use of a model-free approach (Section 2.2). As the name suggests, this approach does not use body models to recover pose but instead employs other methods to achieve this — such as learning observation–pose functions or searching exemplar databases for a match. Both approaches are examined for their benefits and possible use in a human pose tracking system.

2.1 Model-based Approach

Model-based approaches assume some prior knowledge of the object to be tracked and therefore attempt to fit models to the given image or identified image cues. These models may be simple stick-figure models, 2-D contours or even more elaborate volumetric models [5, 18, 41, 71]. The models usually incorporate information such as pose, anthropology and kinematic constraints.

Most 'human motion tracking' systems follow three basic steps in order to achieve this tracking and human pose recovery. They are:

1. feature identification and tracking
2. body structure analysis
3. pose estimation and optimisation

However, these are not the best steps to follow when examining and comparing various motion analysis systems. This is because many of the systems' features fall into more than a single step. For example image descriptors are used for *feature identification and tracking* as well as *body structure analysis*. As a result, the discussion in this section has the following components:

- Body Models
- Image Descriptors
- Optimisation Techniques

2.1.1 Body Models

Body models are used to describe the body of a human according to certain properties. These properties are both the kinematic properties (the underlying skeletal structure and motion) and the shape and appearance of the body (flesh and skin). As much (or as little) detail as required can be collected and used in order to obtain a desired model (anywhere from a plain, rigid stick figure to a flexible, more realistic-looking character).

2.1.1.1 Kinematic Properties

The kinematic properties of the human body consist of the skeletal structure and body motion constraint or kinematic constraint.

The **skeletal structure** is a simple stick figure model which controls the underlying movement of the body. It consists of joints and segments fitted together to give the body the required structure and movement. The different models used vary by the number of joints and segments as well as the degrees-of-freedom allowed.

Lee and Chen's [56] stick figure model, Figure 2.1a, contains 14 joints and 17 segments. The joints are the neck, the left and right shoulders, elbows, wrists, hips, knees and ankles and the pelvis. The segments are the left and right upper arms, forearms, thighs and lower legs, the two shoulder and two hip joining joint segments, the segments joining the neck with the two shoulders and with the pelvis as well as the segments joining the pelvis to the two hip joints. Another model is Kehl and Van Gool's [50] which is a more simplified version, Figure 2.1b, and consists of 10 joints and 14 segments with a total of 24 degrees of freedom.

One of the main types of constraints used in model-based tracking are **kinematic constraints**. These are limitations based on the motion of an object, and in our case that of the human body. An example of this would be the limitations or constraints of the angle and direction a knee can bend. In both body structure analysis and feature tracking, kinematic constraints are very useful as they have the ability to increase the

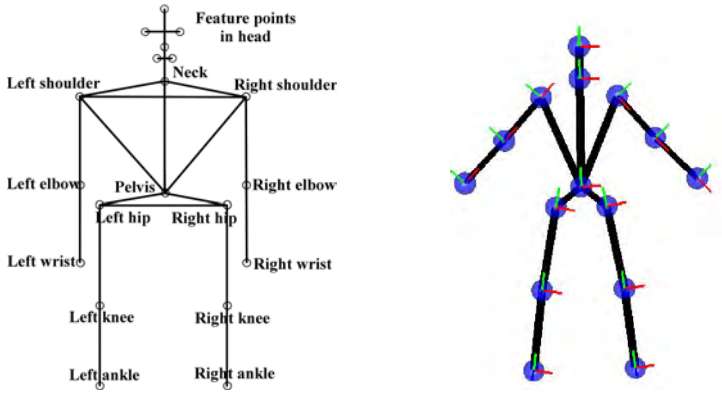


Figure 2.1: Stick figures: (a) Lee and Chen's [56] model and (b) Kehl and Van Gool's [50] simple skeletal

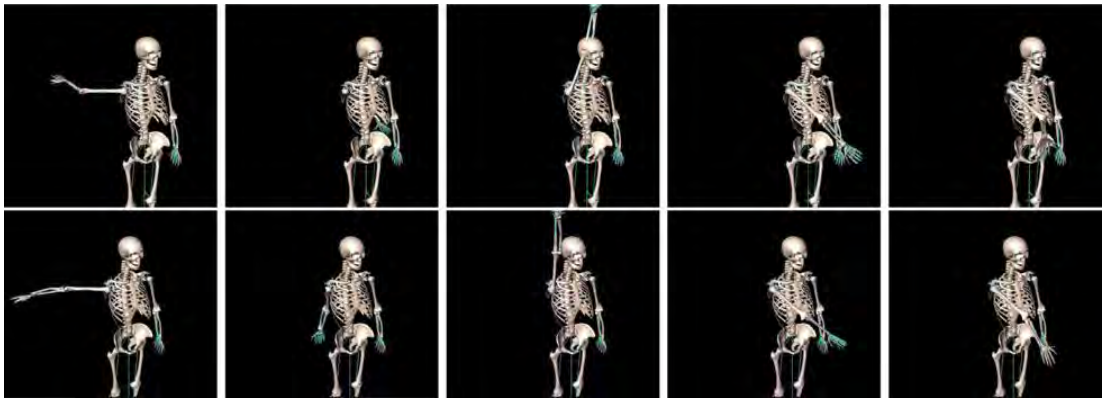


Figure 2.2: Applying hierarchical joint limits to an arbitrary motion — unconstrained (top) and constrained (bottom) motion [38]

reliability of existing motion tracking algorithms. This is done by eliminating anatomically invalid postures which allows the pose estimation stage to more reliably converge on the correct pose. The downside, however, is that they can be very complex and therefore difficult to implement.

There are a number of existing vision systems [15, 21, 75, 88] which have attempted to implement kinematic constraints - but in a very limited and oversimplified manner. In most cases, only hard limits are imposed on the individual Euler angles used to represent the joint rotations or bends in a body model. This gives a minimum and maximum value in which the bend or rotation of a joint can fall. However, this type of model never accounts for the interdependency of joints. A good example to illustrate this is the elbow-shoulder interdependency. If your arm is in front of your body, you can bend your elbow almost 180° from its straight position. However, move your arm behind your back and your elbow can now only bend a maximum of about 90°.

Joint interdependency has been known for a long time and sophisticated models have been developed to account for it. However, these are generally impractical to implement due to the large number of parameters used. Herda *et al.* [38] proposed a model which accounts for joint interdependency while keeping it as simple as possible (see Figure 2.2). To achieve this, they employ a simple parent-child hierarchy system of joints. The range of valid configurations for the child joints is then expressed as a function of the parent joints. So, going back to the elbow-shoulder example, the range of acceptable elbow angles depends on the position and angle of the shoulder joint. Herda *et al.* [38] also use quaternions[1] to enforce the joint-angle constraints. Compared to Euler angles, quaternions are simpler to compose and avoid the problem of *gimbal lock* which can lead to singularities. Thus, they are more numerically stable and possibly more efficient.

2.1.1.2 Shape information

While the resulting human pose and tracking can display the output as a stick figure, it is also possible to use shape information about the subject to create a more realistic appearing model. Most of this shape information can be collected beforehand in an initialisation step and then be fitted over or deformed by the skeleton.

A number of methods have been used to represent this shape information. *Cylinders* were used by Lin [59] while Hogg [39] and Rohr [77] used elliptical cylinders.

Superquadratics [2] are used when more accurate shape modelling is desired or required. These shapes are some distortion of a sphere and have parameters which define their 3D scaling (x, y, z directions) as well as their roundness/squareness. Superquadratics are attractive to use because of their flexibility and relative simplicity. Many people, including Gavrila and Davis [30], have used these shapes to model the shape information (see Figure 2.3a).

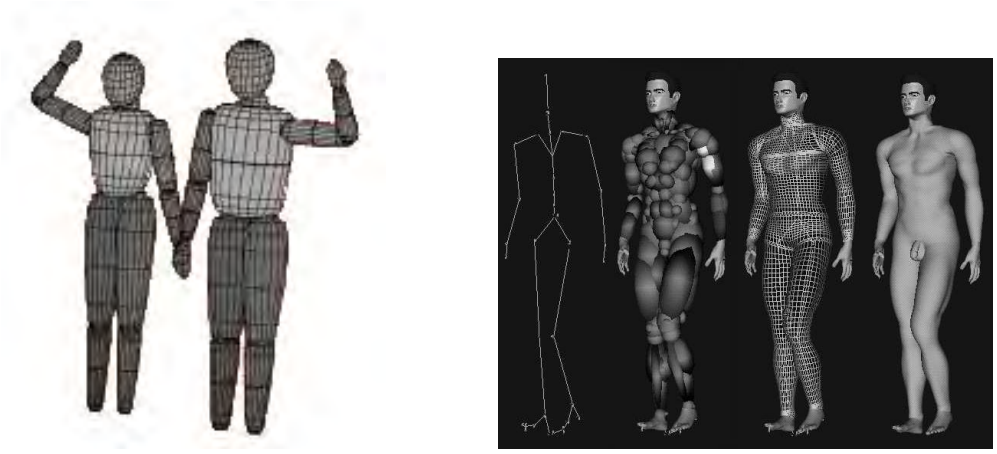
A relatively new concept is that of *metaballs*, from Fua *et al.* [26], which are used to create body models with a more realistic appearance. The metaballs are used to represent the muscle and fat tissue of a human body and are attached to a stick-like skeleton in order to give the model its more realistic anatomical appearance (see Figure 2.3b). A mesh or skin is then overlaid on top of this. The dimensions of the metaballs can be obtained by visual analysis of the subject in an initialisation stage.

2.1.2 Image Descriptors

People in a scene or image are not all the same. They wear different clothes, there may be different lighting conditions and the background may also be changing. Therefore, in order to extract the poses of these changing people, a generalised approach is needed.

[1] a non-commutative number system that extends the complex numbers and provides a convenient mathematical notation for representing orientations and rotations of objects in three dimensions

[2] also know as superquadratic ellipsoids



(a) (b)

Figure 2.3: (a) 3-D superquadric models [30] and (b) use of metaballs to represent muscles and fat tissue along with surface representation of the skin and shaded rendering [72]

This requires image descriptors, such as silhouettes, edges, motion, colour, 3-D data and volumetric properties, to be extracted from the original image and used in the system. A good set of image descriptors allows one to configure the body model accurately and without ambiguity, given the observed data.

Silhouettes can be extracted relatively easily from images, especially when the background is virtually static. Silhouettes are not affected by lighting or colour changes. However, they contain no depth information (see Figure 2.4) which can lead to ambiguities and have limited performance with noisy backgrounds.

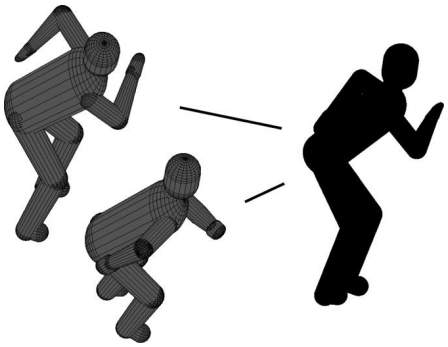


Figure 2.4: Silhouette depth ambiguity illustrated with two different model poses producing the same silhouette image [73]

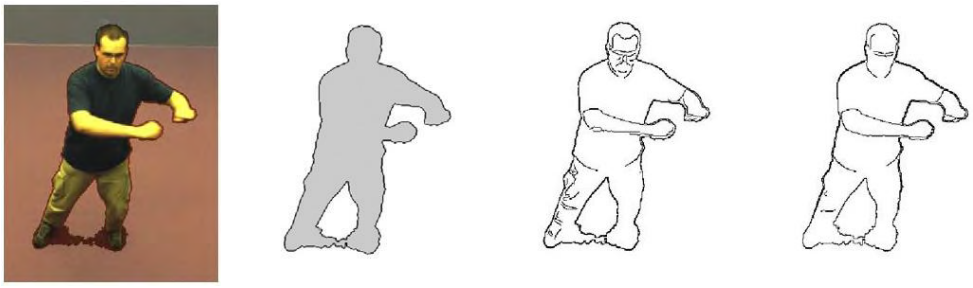
Edges appear all over images wherever there is a difference in intensity between two pixels. Because of this large amount of potential information, edges are often only located within an extracted silhouette. This allows features usually lost in a silhouette to be identified. For example, if an arm is in front of a body it would not be identified

in a silhouette but it could be seen when looking at the edges (see Figure 2.5).

Motion can be used to help with silhouette extraction in that it can help to segment the relatively static background from the moving person in the foreground. Another use is in the area of optical flow and motion vectors. With this, the motion of the pixels in an image is determined and then used to extract or identify objects in scenes. Optical flow is used by Bregler and Malik [14] in their monocular body tracking.

Colour and/or texture are commonly used cues, especially in *no a priori shape model* cases where it is important to track points and small surface regions individually instead of fitting a whole model. In model-based methods, colour and texture may also be used to track or identify certain regions. For example skin colour can be used as a good cue to find the head and hands of a subject. One possible use for this could be an automatic model initialisation sequence. However, colour can be susceptible to changes in illumination of the scene which can lead to problems. On the other hand, texture tends to be more resistant to such changes and is used by Theobalt *et al.* [95] to enhance their silhouette-based tracking method.

Colour can also be used to improve edge detection (see Figure 2.5) by suppressing unwanted clutter edges which are generally caused by shadows or wrinkles/texture of clothing. The use of colour allows this to occur while still retaining the important structural edges.



(a) Original (b) Silhouette (c) Canny (d) RGB

Figure 2.5: The benefit of edge detection within silhouettes. It displays the advantage of (d) RGB edge detection over (c) greyscale Canny edge detection. [50]

Volumetric properties can be extremely useful and have the potential to be a very strong cue as everything has a volume. Voxel-based methods for volumetric reconstruction have increased lately and are used by both Cheung *et al.* [19] and Mikić *et al.* [66] in their work. Voxels are sets of "cubic volume elements which contain information such as which object belongs to this pixel and the history of this object" [4]. While they have become popular, voxels do have a down-side in that they can be very computationally expensive. However, Kehl and Van Gool [50] used look-up-tables for voxel updating,

instead of recalculating each voxel for every frame, which improved the performance time of voxel-based volumetric reconstruction.

The visual hull technique has also been used in human motion analysis. This involves 'stitching together' multiple viewpoint silhouette information to create a 3D image of the subject. This was used by Michoud *et al.* [64], together with voxels, to create a fast, real-time motion capture system. This system ran in real-time at 30 fps using a single computer and four low-cost cameras (webcams).

Menier *et al.* [62] also use the visual hull; this time to estimate the medial axis points of the subject. The medial axis points are a cloud of points which approximate the middle of an object and, hence, on which the skeleton pose is fitted. This method can be sensitive to noise and thus 'pruning' may be used to reduce the size of the cloud and eliminate outliers.

3-D data is another method of acquiring information about the pose of people in an image. Because of its purpose, the use of 3-D data has been classified under *Image Descriptors* even though it is more of a hardware dependent information source. To acquire 3-D spatial information, multiple cameras (at least 2) are needed. With this, spatial information is extracted and used — including the volumetric information described above. Knoop *et al.* [54] discovered that 3-D tracking works very well for large movements, such as a person bowing. However, for smaller quicker moves, such as a quick wave, tracking fails when only 3-D data is used; but when 2-D data is used these smaller, sharper movements can be tracked. This is because the 2-D cues offer better localisation compared to the 3-D cues. Knoop *et al.* [54] showed that a fusion of the 2-D and 3-D tracking gave good results for both the large body movements as well as the fine, fast movements mainly seen on extremities.

Multiple cameras can also help with occlusion handling if employed at specially selected locations around the target area. They also have the benefit of achieving tighter 3-D pose recovery and reduce ambiguity because of the multiple views and hence multiple checks [30, 47]. The disadvantage, apart from the added cost, is the extra complexity and more challenging perspective matching between the cameras [16, 44, 53, 80].

It is unlikely that any of these descriptors will be used independently. Instead, a likelihood function which uses a **combination of descriptors** is commonly used to make the identification and tracking much more robust. Care must be taken when constructing the likelihood function to ensure that it operates effectively and efficiently in the more likely situations. Another two image cues which can also be used (and are fairly self explanatory) are hue and intensity.

When selecting points, areas or features to track, it is important to select those that are robust to size, brightness and contrast changes. They should also be well defined so that poor or incorrect matching does not occur — even though this can be improved through the use of multiple cues. The benefit of multiple cues to track points was

examined and employed by Kahn *et al.* [46]. It was also used by Kehl and Van Gool [50]. In their algorithm they used silhouette methods to extract the subject from the background, edge detectors to extract additional, structural information not seen in the silhouette and colour properties to help reduce unwanted clutter of edges caused by shadows and wrinkles in clothing (see Figure 2.5).

2.1.3 Optimisation Techniques

Optimisation techniques are based around finding the most likely pose, given the available information, in order to minimise the error between the actual observation or image and the human body model.

A number of different optimisation methods, and their variations, are used in human pose estimation. The techniques uses should be able to minimise multidimensional functions since the body pose is not a single entity but made up of numerous joint angle rotations and body part translations — the exact number being determined by the degrees-of-freedom of the body model used.

Traditionally the Kalman filter [99] and Condensation [42] algorithms were popular. Powell's method [17, 24] has also been frequently used, primarily since it does not require a gradient or derivative calculation. However, for non-linear objective functions it can take many iteration steps to converge on a solution and can thus be very slow.

Other gradient techniques used include the Levenberg-Marquardt [57, 61] and Broyden-Fletcher-Goldfarb-Shanno [24] algorithms. These methods handle non-linear optimisation problems well but require the calculation of the first- and second-order derivatives, which may not be available or easily calculated. A simple first-order Gradient Descent method has also been used for pose tracking [74, 87], but again also shows poor convergence for non-linear problems.

Stochastic Meta Descent (SMD) is another method recently used [11, 50]. It is also based on gradient descent but is able to achieve faster convergence by using adaptive and parameter specific step size.

More sophisticated particle filters, such as the Annealed Particle Filter [22, 23], have been used as they reduce the required number of samples and increase the chance of finding the global minimum. This method is based on simulated annealing and can handle multiple hypotheses simultaneously. However, it can be computationally expensive.

2.1.3.1 Top-down and Bottom-up Approaches

Apart from the actual objective function optimisation, there are also various approaches which can be used when performing the optimisation. In the **top-down** approach a projection of the human body is matched with the observed image. This is done from some starting point, such as the head or torso, and then moves out to other parts of the

body. The disadvantage is that errors are propagated throughout the model if there is an error in any of the previous 'fittings'. Computation wise, it does not allow any parallel processing as parts are fitted one after the other. Its advantage is that it is able to make use of the fact that the body parts are not independent but related to each other in known models.

On the other hand, the **bottom-up** approach finds the individual body parts separately and then assembles them into a human body. This has the disadvantage of resulting in more false positives as there can be many 'limb-like' regions in an image. It also does not make use of any inter-dependency properties of the human body. However, it may take advantage of parallel processing which can increase the convergence speed of the estimation and optimisation.

As with many other things, it is once again beneficial to combine the two approaches and take advantage of their respective benefits. This is done to some extent by Kehl and Van Gool [50] where first the whole body model is fitted with a relatively large tolerance, in a top-down fashion, and then the individual limb sections are refined separately with a much smaller tolerance in a hierarchical scheme, much like the bottom-up manner, where parallel processing can be exploited.

2.2 Model-free Approach

The model-free approach is different from that of the model-based approach in that no explicit human body model is either available or used. Instead, some sort of mapping or direct relation needs to be established between the observed image and the pose state. Poppe [73] identifies two main classes which allow this pose estimation to be achieved: *learning-based* and *example-based*.

The model-free approach is relatively new to human pose tracking. While it does not use any form of body model, as described in Section 2.1.1, it still has the same visual input and therefore makes uses of the same image descriptors detailed in Section 2.1.2. It is the way in which the obtained visual information is used that changes.

2.2.1 Learning-based

In learning-based approaches a function or relationship between the observed image (or image space) and the pose (or pose space) is learned by using training data.

A number of researchers including Agarwal *et al.* [3], Brand [10], Rosales and Sclaroff [78] and Sminchisescu *et al.* [89] have done work in this field. In a number of cases single- and multi-view silhouettes and 3D joint locations are used as the input image descriptors to the process. A range of different techniques have been used to model the relationship between the image space and pose space. These include non-linear

regression, the Viterbi algorithm, neural networks and a mixture of expert models.

Recently, Taycher *et al.* [93], looked at transforming the continuous state estimation problem into a discrete problem. This allows the modelling function to be learned off-line from a large number of examples — meaning a large amount of processing work can be performed beforehand. This leads to the potential for the poses to be recovered in real-time.

2.2.2 Example-based

Conversely, the example-based approach stores a database of exemplars that describe poses in both image and pose space. It therefore avoids learning any mapping function and instead searches of the database are performed and the resulting poses interpolated to obtain the pose estimate.

Once again, a number of researchers including Mori and Malik [68], Bowden *et al.* [9], Howe [40] and Shakhnarovich *et al.* [82] have done research in this area. The main difference between the various methods is the manner in which a stored image is searched for and fitted to an observed image. Methods and techniques such as a non-linear point distribution model, deformation of exemplars, Markov Chaining with smoothing, Nearest Neighbour search and Parameter Sensitive Hashing are some of those used.

In both cases a large amount of memory space is needed to store either the training data or exemplar database — which is a big drawback to the model-free approach.

However, they do have the important advantage that, in most cases, the pose can be estimated independently in each image frame (instead of requiring previous information as in the model-based approach). This independent pose recovery allows both estimation of rapid movements and recovery from self-occlusion (and other tracking failures) since there is essentially no reinitialisation problem as initialisation does not exist.

This benefit is used by Micilotta *et al.* [65] where model-based and model-free approaches have been combined to enable automatic initialisation as well as recovery if a tracking failure is experienced.

2.3 Summary

Initially pose tracking of humans was performed using markers, such as MLDs (moving light displays), and sensors. These systems are fairly accurate but require special suits to be worn and operate in special environments. The systems can also be rather expensive. With the increase in processing power of computers, markerless pose tracking systems are becoming more popular.

While both monocular (single camera) and multi-view (two or more cameras) setups are used, monocular methods experience problems with occlusion and depth measurement. To overcome these difficulties more sophisticated tracking methods need to be used which can be very processor intensive. Constraints are also placed on the systems to improve their performance but this makes them less general and more motion and application specific.

On the other hand, the use of multiple views helps to combat occlusion problems and allows information from multiple angles to be used to obtain depth information and combat ambiguity. The use of multiple cameras allows a 3-D reconstruction of the subject to be made — which is a very strong image cue. While the increase in views may seem to lead to increased processing, that is not necessarily the case as only certain cues from the images are used. In fact various 3-D reconstruction methods are able to run in real-time.

Model-free approaches appear to work very well and can operate in real-time. However, they are very limited by the poses and actions in their databases. This places constraints on them and makes them more specifically focussed. This approach also has the potential for reduced accuracy as interpolation is used to determine poses. Without prohibitively large databases, this method does not allow unusual or diverse poses to be recovered either.

Therefore, in order to create a markerless pose tracking system which is fairly accurate yet can be used very generally, a multi-view model-based approach is used. Multiple image cues are combined to improve the robustness. The strong 3-D reconstruction cue forms a base while the 2-D silhouettes and colour edges help to refine the tracking. The 2-D cues also help when the 3-D reconstruction suffers from bulkiness and/or low accuracy — as it has the potential to. The multiple views, as well as use of colour, help with ambiguities and occlusion.

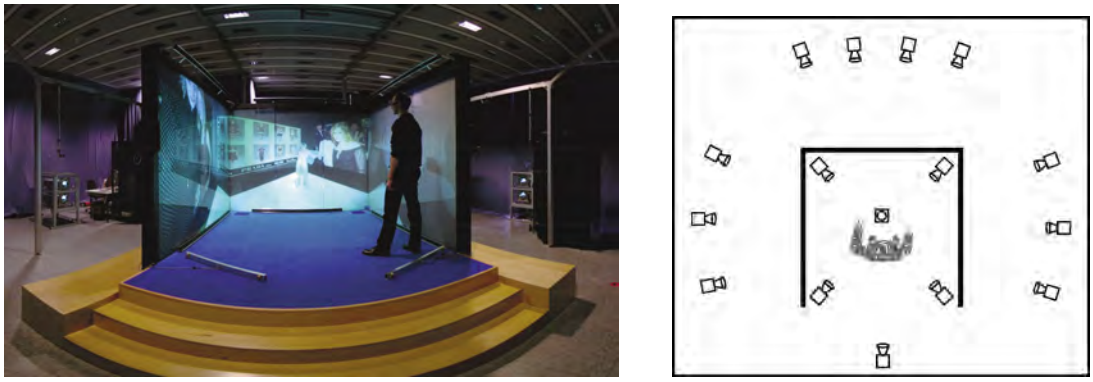
The optimisation is performed by combining SMD with particle filters to form a Smart Particle Filter (SPF) [13]. This combines the benefits of SMD, good convergence, reduced processing time and high-dimensional tracking, with the multiple hypotheses framework of particle filters. SMD was selected over other optimisation methods for its improved accuracy and faster convergence as shown in [11].

There is no specific clothing constraint placed on the subject being tracked (unlike some systems) and no special environment is required as long as a few cameras surround the tracking area. A coloured superellipsoid body model is used for its simplicity in calculation while still offering a reasonably humanoid-shaped model. This model is utilised both in optimisation and for the resultant output display.

Chapter 3

Image Capture

The image capture stage deals with the gathering of observable data. This is in the form of either still or video images. While it is possible to use a monocular (single camera) approach, these usually exhibit various limitations such as pose ambiguity and occlusion [15, 75, 88]. Therefore, following Cai and Aggarwal [16], Gavrila and Davis [30], Kehl and Van Gool [50] to list a few, a multi-camera approach is used to minimise these problems. The measurement environment is thus defined as a capture space in the centre surrounded by a number of cameras. A example of such a setup is the Blue-C portal CAVE [33] at ETH Rechenzentrum (see Figure 3.1).



(a) (b)

Figure 3.1: (a) Panoramic picture of the Blue-C portal showing the projection screens and the outer scaffold with attached cameras and (b) bird's-eye view of the camera arrangement.

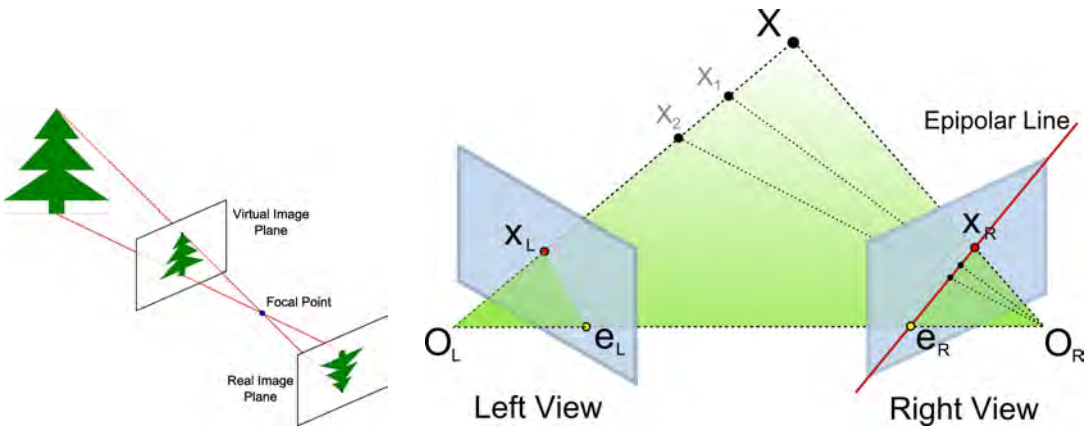
An important part of image capture is camera calibration, which is the process of determining the internal camera geometry and distortion characteristics as well as their 3-D world co-ordinate location and orientation. Multi-camera calibration, in addition, allows information from multiple views to be combined. The geometry behind camera calibration as well as the 2-D to 3-D transformation is examined in Section 3.1. Camera calibration follows in Section 3.2 before the datasets used for testing and their setups are briefly described in Section 3.3.

3.1 Epipolar Geometry

Epipolar geometry refers to the geometry of stereo vision [103].It is used to describe the geometric relationships between a real-world, 3-D scene and the scene's projections onto 2-D images from different viewpoints. These 2-D images usually come from cameras which are focused on the same scene but a little distance apart. The basic idea is that given two different views of the same scene with known camera parameters and relative position, then various properties of the 3-D scene can be calculated — including the 3-D location of various objects.

The relations between the 3-D and 2-D views are based on the assumption that the cameras can be reasonably approximated by the pinhole camera model. This model describes the camera's aperture as a point and does not include (among other things) any lens to focus the light. However, some of these distortion effects can be compensated for and others are sufficiently small to be neglected — especially if high quality cameras are used. Therefore, the simplification the use of the pinhole camera model provides is justified.

The image plane for a real camera is behind the focal point or aperture/lens and is inverted. However, to simplify the problem a virtual image plane is used instead and is placed in front of the camera's focal point such that the real and virtual image planes are equidistant from the focal point (see Figure 3.2a).



(a) (b)

Figure 3.2: Diagrams illustrating, (a) Real and imaginary image planes and (b) Epipolar geometry.

Figure 3.2b depicts a simple example of epipolar geometry which is used to explain the concept. Suppose point $\mathbf{X}$ is viewed by two cameras with $\mathbf{O}_L$ and $\mathbf{O}_R$ representing the focal points of the left and right cameras respectively. Now points $\mathbf{x}_L$ and $\mathbf{x}_R$ are the projections of $\mathbf{X}$ onto the cameras' image planes. The two focal points $\mathbf{O}_L$ and $\mathbf{O}_R$

can be projected into the other camera's image plane along a single straight line called the baseline. These image points, $\mathbf{e}_L$ and $\mathbf{e}_R$, are called epipoles or epipolar points and are common for all viewed points $\mathbf{X}$, since the cameras do not change position. If the line $\mathbf{O}_L - \mathbf{X}$ is viewed by the left camera it will see a point ($\mathbf{x}_L$) since it is in line with the focal point. However, if viewed by the right camera, a line will be seen since it is viewed 'side on'. This image line, $\mathbf{e}_R - \mathbf{x}_R$, is called the epipolar line. This can be viewed in exactly the same manner with the cameras the other way around. Line $\mathbf{O}_R - \mathbf{X}$ is seen by the right camera as point $\mathbf{x}_R$ but by the left as the epipolar line $\mathbf{e}_L - \mathbf{x}_L$. Since any point $\mathbf{X}$, when viewed by the left camera, must have a line passing through its focal point, $\mathbf{O}_L$, then the corresponding epipolar line in the right image must pass through the epipolar point $\mathbf{e}_R$ (and vice versa).

If the relative translation, rotation and position of the two cameras are known, then two important observations can be obtained from epipolar geometry.

1. **Triangulation**
   If image points $\mathbf{x}_L$ and $\mathbf{x}_R$ are known and correspond to the same point $\mathbf{X}$, then their projection lines are also known and must intersect at point $\mathbf{X}$. This means that the co-ordinates of $\mathbf{X}$ can be calculated from the co-ordinates of the image points $\mathbf{x}_L$ and $\mathbf{x}_R$ by triangulation.

2. **Epipolar constraint**
   If point $\mathbf{x}_L$ is known, then the red epipolar line $\mathbf{e}_R - \mathbf{x}_R$ is also known (since the relative parameters of the cameras mentioned above are known). This means that the image point $\mathbf{x}_R$ of point $\mathbf{X}$ lies somewhere along that line. This provides a constraint, called the epipolar constraint, which can either limit the area of the right image plane which must be searched to find $\mathbf{X}$ or which can provide a test to check that the two points ($\mathbf{x}_L$ and $\mathbf{x}_R$) really correspond to the same 3-D point. This epipolar constraint is described by the fundamental or essential matrix between the two camera views.

3.1.1 The Essential and Fundamental Matrices

The fundamental matrix, $\mathbf{F}$, is a 3×3 matrix which is the algebraic representation of epipolar geometry. If $\mathbf{x}_L$ and $\mathbf{x}_R$ are a corresponding stereo image pair (Figure 3.2b), then $F\,x_R$ describes the epipolar line along which the corresponding point $\mathbf{x}_L$, on the other image, must lie. Hence, the fundamental matrix relates corresponding stereo points or images.

The essential matrix, $\mathbf{E}$, is very similar to the fundamental matrix but describes the correspondence when using calibrated cameras where the inner intrinsic camera properties are known so that normalised world co-ordinates are used instead of the image co-ordinates.

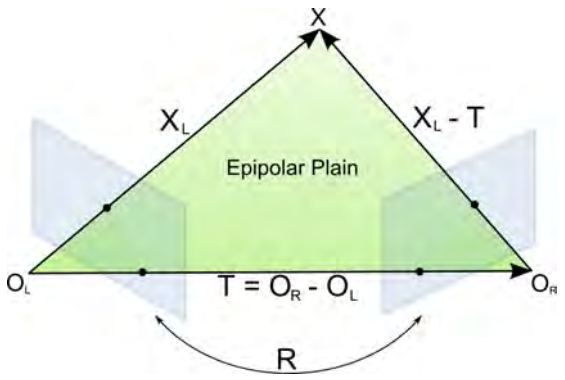


Figure 3.3: Epipolar plain vectors used to define the coplanarity condition

The epipolar plain (Figure 3.3) is the plain defined by $\mathbf{X}$, $\mathbf{O}_L$ and $\mathbf{O}_R$ where $\mathbf{X}_L$ and $\mathbf{X}_R$ are vectors giving 3-D positions relative to $\mathbf{O}_L$ and $\mathbf{O}_R$ (i.e. they are projections of $\mathbf{X}$ from the left and right cameras). Now the two camera views are related by a translation,

$$\mathbf{T} = \mathbf{O}_R - \mathbf{O}_L$$

and a rotation, $\mathbf{R}$, such that

$$\mathbf{X}_R = \mathbf{R}\,(\mathbf{X}_L - \mathbf{T}) \tag{3.1}$$

The epipolar plain equation through $\mathbf{X}$ is given by following the coplanarity condition

$$(\mathbf{X}_L - \mathbf{T})^T \cdot (\mathbf{T} \times \mathbf{X}_L) = 0 \tag{3.2}$$

The following shows this empirically. $\mathbf{X}_L - \mathbf{T}$ gives a vector in the epipolar plane. The cross product, $\mathbf{T} \times \mathbf{X}_L$, gives a vector *perpendicular* to the epipolar plane since $\mathbf{T}$ and $\mathbf{X}_L$ are both in it. Therefore, the dot product ($\mathbf{A} \cdot \mathbf{B} = |\mathbf{A}|\,|\mathbf{B}| \cos(\angle \mathbf{AB})$) returns a zero, 0, since $(\mathbf{X}_L - \mathbf{T})^T$ and $(\mathbf{T} \times \mathbf{X}_L)$ are perpendicular to one another which results in their angle being 90°, thus Equation (3.2) is satisfied.

Reorganising Equation (3.1) one gets

$$\mathbf{R}^T \mathbf{X}_R = \mathbf{X}_L - \mathbf{T} \tag{3.3}$$

Substituting this into Equation (3.2) gives

$$(\mathbf{R}^T \mathbf{X}_R)^T \mathbf{T} \times \mathbf{X}_L = 0 \tag{3.4}$$

Now the vector cross product can be expressed as matrix multiplication [60] as follows

$$\mathbf{T} \times \mathbf{X}_L = [\mathbf{T}]_\times \, \mathbf{X}_L = \mathbf{S}\,\mathbf{X}_L$$

where

$$\mathbf{S} \stackrel{def}{=} [\mathbf{T}]_\times = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$$

Therefore, from (3.4)

$$\mathbf{X}_R^T \mathbf{R}\,\mathbf{S}\,\mathbf{X}_L = 0 \tag{3.5}$$

which can be rewritten as

$$\mathbf{X}_R^T \mathbf{E}\,\mathbf{X}_L = 0 \tag{3.6}$$

where $\mathbf{E} = \mathbf{R}\,\mathbf{S}$ is the essential matrix.

Thus $\mathbf{E}$ can be used to link $\mathbf{X}_L$ and $\mathbf{X}_R$, projection vectors of the 3-D points $\mathbf{x}_L$ and $\mathbf{x}_R$. However $\mathbf{X}_L$ and $\mathbf{X}_R$ are not generally known, one only knows where $\mathbf{X}$ appears in each image (ie $\mathbf{x}_L$ and $\mathbf{x}_R$).

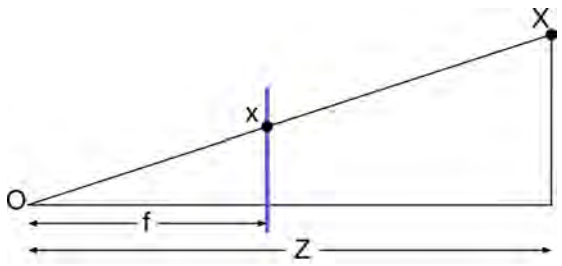


Figure 3.4: Diagram showing the perspective projection of point $\mathbf{X}$

Using perspective projection (Figure 3.4) and similar triangles it may be shown that

$$\mathbf{X} = \mathbf{x}\frac{Z}{f} \tag{3.7}$$

Therefore,

$$\mathbf{X}_L = \mathbf{x}_L\frac{Z_L}{f_L} \qquad \text{and} \qquad \mathbf{X}_R = \mathbf{x}_R\frac{Z_R}{f_R} \tag{3.8}$$

Substituting the above two equations into Equation (3.6) and multiplying through by $\dfrac{f_L\,f_R}{Z_L\,Z_R}$ gives

$$\mathbf{x}_R^T \mathbf{E}\,\mathbf{x}_L = 0 \tag{3.9}$$

which shows that the points $\mathbf{x}_L$ and $\mathbf{x}_R$ are also linked.

All these calculations have been done in 3-D world co-ordinates but the data received

from images is in the form of 2-D pixel co-ordinates. To transform from the 2-D pixel to 3-D world co-ordinates the intrinsic camera parameters are needed. These include the focal length, principal point and skew and distortion coefficients of the camera. Let $\mathbf{M}_L$ and $\mathbf{M}_R$ be the matrices of the intrinsic parameters for the left and right cameras respectively, which transform the world co-ordinates $\mathbf{x}_L$ and $\mathbf{x}_R$ to the pixel co-ordinates $\hat{\mathbf{x}}_L$ and $\hat{\mathbf{x}}_R$. Therefore

$$\hat{\mathbf{x}}_L = \mathbf{M}_L \, \mathbf{x}_L \qquad \text{and} \qquad \hat{\mathbf{x}}_R = \mathbf{M}_R \, \mathbf{x}_R \tag{3.10}$$

Changing the subject of the formula and substituting into Equation (3.9) gives

$$(\mathbf{M}_R^{-1} \, \hat{\mathbf{x}}_R)^T \, \mathbf{E} \, \mathbf{M}_L^{-1} \, \hat{\mathbf{x}}_L = 0 \tag{3.11}$$

or

$$\hat{\mathbf{x}}_R^T \, \mathbf{F} \, \hat{\mathbf{x}}_L = 0 \tag{3.12}$$

where $\mathbf{F} = \mathbf{M}_R^{-T} \, \mathbf{E} \, \mathbf{M}_L^{-1}$ is the fundamental matrix.

Therefore, the pixel co-ordinates, in the left and right camera images, of $\mathbf{X}$ are linked by Equation (3.12).

It is possible to observe that the essential matrix is defined in the world co-ordinates and only encodes information about the extrinsic camera properties (translation and rotation). On the other hand, the fundamental matrix is defined in pixel co-ordinates and encodes both the extrinsic and intrinsic camera properties. As such, it is generally more practical to use the fundamental matrix since the pixel co-ordinates of viewed objects is the data obtained.

3.2 Camera Calibration

Having seen how 2-D pixel co-ordinates can be mapped to 3-D world co-ordinates, one realises that camera and setup information is required to achieve this.

Camera calibration is the process of determining these parameters and includes the internal camera geometry and lens/optical distortion characteristics (intrinsic parameters) as well as the 3D position and orientation of the camera relative to some common world co-ordinate system (extrinsic parameters). All vision systems, and their performance, greatly depend on the accuracy of the camera calibration; hence it is a very important task.

Originally camera calibration techniques could be roughly split into two categories:

- **Photogrammetric calibration** is where calibration is performed by observing a very well known calibration object. This object usually has two or more

orthogonal planes. Alternatively, a plane undergoing some precisely known movement or translation may also be used. This method, however, requires expensive and specialised equipment and setup and is thus not practical for general vision systems.

- **Self-calibration** is where no calibration object is used but instead the camera is moved in a static scene. The rigidity of the scene provides constraints from one camera displacement to another. These, together with the image information, can be used to recover the internal and external camera parameters since the same camera (with fixed internal properties) is used in all cases. While this approach is flexible, it does not always provide reliable results because there are many parameters to estimate. It also cannot be used to obtain inter-camera relationships, which is very important for stereo vision systems.

Nowdays, however, far more flexible and versatile methods are used. One of the more common approaches is for the camera(s) to observe a known planar pattern — usually an accurate checker-pattern printed using a laser printer and attached to some flat surface. Now either the pattern or the camera can be moved. However, if using multiple cameras, as with the system described in this dissertation, it is best to fix the cameras in place and then move the planar pattern by hand. The one advantage with this method is that the movement need not be known as long as the pattern can be reasonably seen from all the camera views. This makes it a more flexible method than the traditional photogrammetric calibration while retaining more robustness compared to the self-calibration.

3.2.1 Calibration Toolbox

Since the project's main focus is not camera calibration, a third party, preferably free yet robust, calibration algorithm needed to be found. Fortunately, a number of these are available. Of the calibration methods examined, the *Camera Calibration Toolbox for Matlab* by Jean-Yves Bouguet [8] stood out. For this calibration, the intrinsic camera model is inspired by Heikkila and Silven's work [35], while its main initialisation phase has been partially based on Zhang's method mentioned in his paper, [105]. This toolbox also has a number of example tutorials which demonstrate how to use it under different circumstances. The toolbox is included in another very handy tool — the *Multi-Camera Self-Calibration Toolbox* by Svoboda [92]. This toolbox is also freely available for non-commercial use and, as its name implies, is used to calibrate multiple (three or more) cameras where the Jean-Yves Bouguet's toolbox can only calibrate up to a stereo (two) camera configuration.

Since most multi-camera setups use more than two cameras, Svoboda's toolbox is generally used. It calculates the linear camera parameters itself (internal: focal length,

principle point, skew coefficient ; external: camera translation and rotation) and employs Bouguet's toolbox to estimate the non-linear distortion parameters. Instead of using a known calibration pattern, as in Bouguet's toolbox, which is only able to be viewed by a small number of cameras, a laser pointer is used. By waving this around the working volume, a virtual calibration object is created which is viewed by most if not all the synchronised cameras.

The calibration is performed in three easy steps:

1. Capture images while waving a laser pointer around in the capture space. It is important to keep the laser pointer visible to as many cameras and to cover as much of the working volume as possible.

2. Run software from the toolbox to compute the image statistics and find and measure the laser point projected in the camera.

3. Use the self-calibration software (which runs Svoboda's toolbox) to estimate all the camera parameters.

3.2.1.1 Calibration Parameters

Without going into too much detail, the various parameters produced by the toolbox, along with their meaning and how they are used, is presented. These parameters are calculated once at the start, during the setup phase, and thus the image capture environment must remain fixed for the duration of the image capture.

The **extrinsic parameters** are calculated for each camera and determine the position and orientation of each with respect to some defined world co-ordinate origin.

These parameters are defined as:

- *Rotation* – A 3×3 rotation matrix $\mathbf{R}$.
- *Translations* – A 3×1 vector $\mathbf{T}$.

This allows the co-ordinates of some 3-D point, $\mathbf{P}$, seen in the first camera's image as $\mathbf{X_1}$ and in the second camera's image as $\mathbf{X_2}$ to be related through the rigid motion equation:

$$\mathbf{X_2} = \mathbf{R}_2 \left(\mathbf{R}_1^{-1} \left(\mathbf{X_1} - \mathbf{T}_1 \right) \right) + \mathbf{T}_2 \tag{3.13}$$

The **intrinsic parameters** produced are:

- *Focal length* – the focal length, in pixels, is stored in the 2×1 vector **fc**.
- *Principal point* – the principal point co-ordinates are stored in the 2×1 vector **cc**.

- *Skew coefficient* – the skew coefficient, defining the angle between the x and y pixel axes, is stored in the scalar **alpha_c**.
- *Distortions* – the image distortion coefficients (radial and tangential distortions) are stored in the 5×1 vector **kc**.

In order to demonstrate how these are used, one needs to start at the beginning.

The extrinsic camera transform $\mathbf{M}_c$ is constructed from a camera's rotation and translation matrices as in (5.4). It transforms a point $\mathbf{P}$ from its 3-D world reference frame point, $\mathbf{X}_p$, to its camera reference frame point, $\mathbf{X}_c$, via

$$\mathbf{X}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \mathbf{M}_c \mathbf{X}_p = \begin{bmatrix} R_c & T_c \\ 0 & 1 \end{bmatrix} \mathbf{X}_p$$

Projecting this onto the image plane one obtains the normalized image projection

$$\mathbf{X}_n = \begin{bmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{3.14}$$

Using the lens distortion information stored in the **kc** vector and defining $r^2 = x_n^2 + y_n^2$, the new normalised point co-ordinate, $\mathbf{x}_d$, is calculated as

$$\begin{aligned} \mathbf{x}_d = \begin{bmatrix} x_d \\ y_d \end{bmatrix} &= (1 + \mathbf{kc}(1)r^2 + \mathbf{kc}(2)r^4 + \mathbf{kc}(5)r^6) \begin{bmatrix} x_n \\ y_n \end{bmatrix} \\ &+ \begin{bmatrix} 2\mathbf{kc}(3)x_n y_n + \mathbf{kc}(4)(r^2 + 2x_n^2) \\ \mathbf{kc}(3)(r^2 + 2y_n^2) + 2\mathbf{kc}(4)x_n y_n \end{bmatrix} \end{aligned} \tag{3.15}$$

Finally, the normalised point co-ordinate $\mathbf{x}_d$ is mapped to the pixel co-ordinate $\mathbf{x}_p$ by the *camera matrix* $\mathbf{K}$ which contains various camera parameters. Thus the pixel co-ordinate of $\mathbf{P}$ in the image plane, with the effects of distortion, is:

$$\begin{aligned} \mathbf{x}_p = \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} &= \mathbf{K} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} fc(1) & alpha\_c \times fc(1) & cc(1) \\ 0 & fc(2) & cc(2) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \end{aligned} \tag{3.16}$$

In cases where the non-linear lens distortions are assumed negligible and ignored, the mapping between the 3-D world co-ordinate of point $\mathbf{P}$, $\mathbf{X}_p$, and its image co-ordinate,

$\mathbf{x}_p$, is simplified and can be calculated by

$$\mathbf{x}_{pn} = \mathbf{K} \begin{bmatrix} R_c & T_c \\ 0 & 1 \end{bmatrix} \mathbf{X}_p = \boldsymbol{P}\mathbf{X}_p$$
$$\mathbf{x}_p = \begin{bmatrix} \frac{x_{pn}}{z_{pn}} \\ \frac{y_{pn}}{z_{pn}} \end{bmatrix} \tag{3.17}$$

where $\boldsymbol{P} = \mathbf{K}\mathbf{M}_c$ is the *camera projection matrix*.

3.3 Dataset for testing

Traditionally human motion and pose capture methods have been evaluated mostly qualitatively. This is largely due to the fact that there was no real common or standard dataset available, with ground-truth, for testing and evaluation of the algorithms.

Recently, however, the HumanEva [83, 84] dataset has been made available in which synchronised video and 3-D ground-truth has been captured. Since its release, numerous researches have used this dataset for evaluation purposes: Bo *et al.* [7], Lee and Elgammal [55], Li *et al.* [58], Ning *et al.* [70], Rogez *et al.* [76], Urtasun and Darrell [96], Vondrak *et al.* [98], Xu and Li [104].

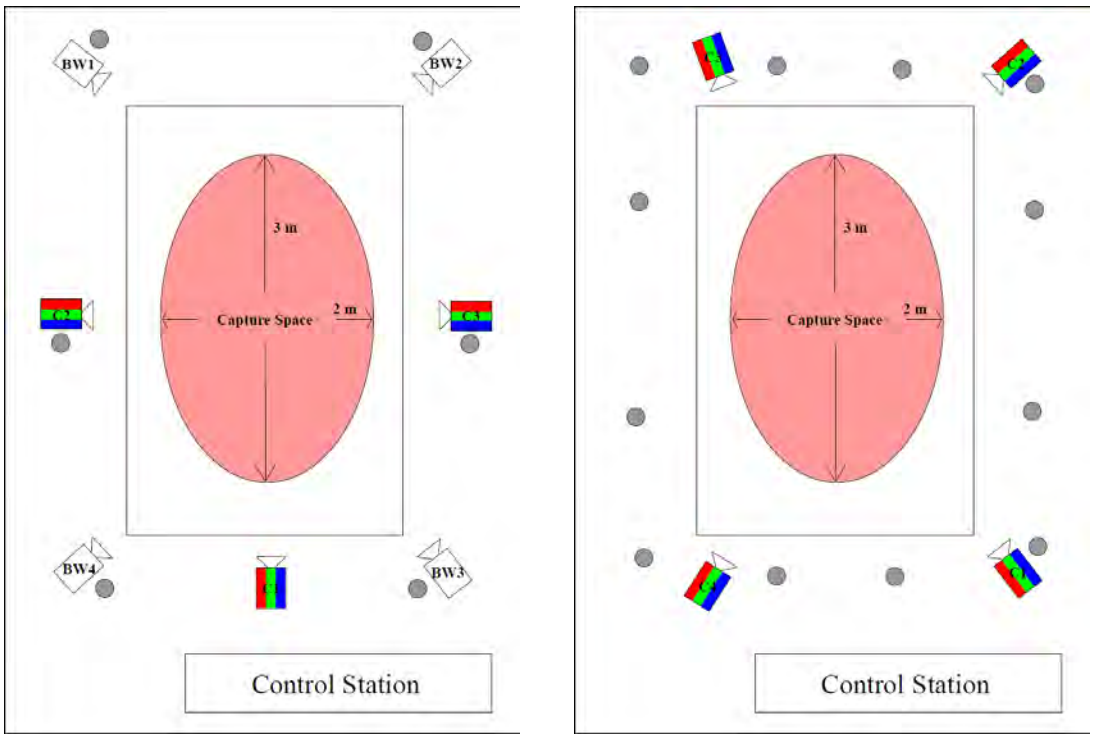
It provides a number of benefits over other available datasets which makes it very attractive to use. These include multiple colour cameras (with calibration data), a number of different subjects performing a range of motions, the natural appearance of the subjects and 3-D ground-truth data. While some of the other datasets [34, 100] have some of these features, none have all of them. That is not to say the HumanEva dataset is perfect (the HumanEvaI set has seven camera views but only three are colour, it would be preferable if there were more colour views) but currently it is the best available.

The HumanEva dataset contains two different versions, HumanEvaI and HumanEvaII. The setup of each is shown in Figure 3.5.

HumanEvaI [1] data was captured earlier using 3 colour and 4 greyscale cameras. The data captured was from four subjects performing a range of different actions — walking, jogging, gesturing, throwing and catching, boxing and combo. The cameras captured the images at a frame rate of 60Hz with a colour camera resolution of 659×494 pixels and grayscale camera resolution of 644×448.

HumanEvaII [2] data was captured using four colour cameras and contains a combo sequence of captured images from 2 subjects. In the combo sequence the subject walks in a circle then jogs in the same direction after which balancing on each leg is performed. The images were captured at a 60Hz frame rate with 656×490 pixels resolution.

[1] http://vision.cs.brown.edu/humaneva/download1.html
[2] http://vision.cs.brown.edu/humaneva/download2.html



(a) (b)

Figure 3.5: Bird's eye view sketch of the (a) HumanEvaI and (b) HumanEvaII capture configurations. The colour video cameras (C) are depicted by a RGB stripped pattern, the greyscale cameras (BW) by the empty camera icon and the motion capture cameras are shown as grey circles.

In each case the ground-truth motion capture data was acquired using a commercial marker-based motion capture system from ViconPeak[3]. This system is "an industry standard for optical marker-based motion capture and has been successfully employed in a variety of entertainment applications for over 10 years" [83]. The 3-D position of reflective markers, placed on the subjects' clothing with invisible tape, was recovered using the ViconPeak system's cameras and the 3-D body pose was estimated. The ground-truth for HumanEvaII is considered to be slightly more accurate as twelve 1.3 Mpixel cameras were used as opposed to six 1 Mpixel cameras for the HumanEvaI ground-truth.

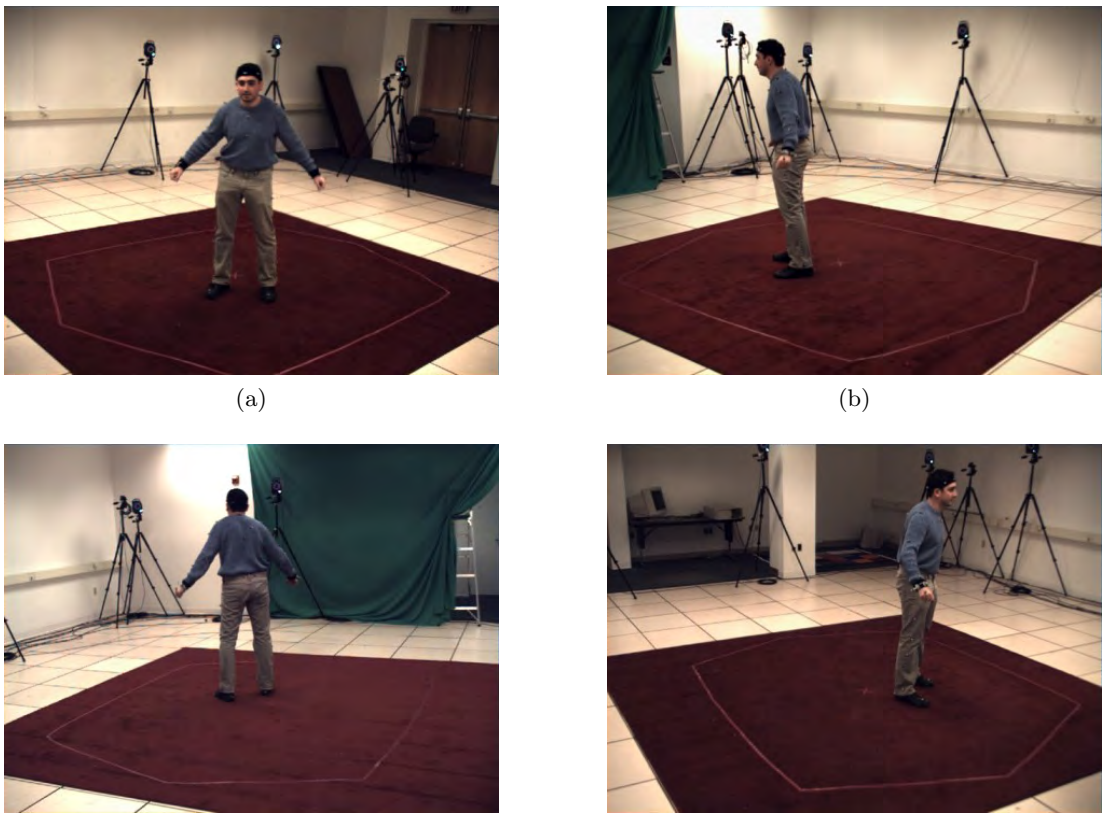
Calibration of the system was performed using Vicon software for the extrinsic properties while Bouguet's *Camera Calibration Toolbox for Matlab* [8] was used to obtain the intrinsic camera parameters.

Although the HumanEvaI dataset is larger and therefore more preferable to use than the HumanEvaII dataset, it only has three colour cameras and slightly less accurate

[3]http://www.vicon.com/

ground-truth data. It does, however, make available the ground-truth data for many of the sequences — which HumanEvaII does not. The ground-truth motion capture data is withheld for a number of test sequences in HumanEvaI and all, except the first few frames, of HumanEvaII to prevent parameter tuning. Instead, on-line evaluation of the test data is available from the project website[4] for quantitative evaluation of the test sequences.

Since colour information from the captured images is utilised, the HumanEva II dataset is used for the majority of the testing as it provides the most colour cameras. Figure 3.6 shows example images from the four camera views of the HumanEvaII capture setup. With very close examination of the images, it may be possible to see the markers on the subject which are used by the motion capture system to determine the body pose for testing and verification purposes.



(a) (b)

(c) (d)

Figure 3.6: Example image from the four camera views of the HumanEva II capture setup. Figures (a) – (d) show camera views 1 – 4.

[4] http://vision.cs.brown.edu/humaneva/index.html

3.4 Summary

The HumanEva datasets contain the calibration parameters for the camera setups. Therefore, it is not essential to know the inner workings of epipolar geometry and camera calibration. However, in order to successfully use and implement the calibration parameters this background information does help and gives one a better understanding of the calibration process. The HumanEvaI and II datasets provide a very good evaluation base for testing human pose capture algorithms and allow quantitative analysis of the results to be performed. This is achieved through the use of ground-truth motion capture data which was captured simultaneously with the video data. Due to the availability of four colour cameras the HumanEvaII dataset is used for most of the testing.

Chapter 4

Image Processing

The image processing stage is an important part of the human pose tracking system as it converts the observed data into some usable form for tracking. Segmentation is dealt with in Section 4.1, where the foreground subject of interest is separated from the background. Section 4.2 presents edge detection, an important feature in image analysis which offers 2-D information. Volumetric reconstruction, a visual hull construction from multiple intersecting silhouettes which provides 3-D information, is explored in Section 4.3. Results of all the processing is also given in Section 4.4.

4.1 Segmentation

Segmentation is the first step taken after acquiring an image. In this process the foreground subject of interest is segmented or separated from the background. This allows only the section of interest to be retained and passed on for further processing — thus reducing the processing required in later steps, such as edge detection.

While there are many different methods and variations of segmentation, this work predominantly focuses, and is based, on the illumination-invariant method proposed by Mester *et al.* [63], on which Kehl *et al.*'s [52] work is also based.

In order for segmentation to be successful, it should have a very low false alarm rate (be robust to various disturbances in the video) and yet be able to detect all the truly relevant visual content. This change detection is often achieved by using statistical decision and detection theory. In these types of methods a locally computed test statistic (for each image pixel) is compared to some global threshold to determine if the pixel is foreground or background. However, this 'non-adaptive' method often produces a large number of false positive and false negative alarms. To improve on this, prior information about the object of interest can be incorporated into the algorithm — such as typical shape and size. This is commonly achieved by using a Gibbs-Markov random field which introduces spatial context into pixel labelling, such as background-foreground segmentation.

While the above helps to some extent, real-life video data presents some very challenging situations of its own, the most important one being illumination changes. A change in illumination results in a noticeable visual and numerical change (mimicking a change from the previously known background, making it a foreground). However, in most cases this should not be regarded as a relevant change since a change in lighting does not correspond to a change in the objects in a scene. Therefore, focus is put into integrating illumination invariance into a segmentation framework based mainly on decision theory and statistical image models [63].

4.1.1 Change detection with a Bayesian framework

To build a solid starting point, change detection is initially formulated as a Bayesian estimation problem [2]. From this, various rules and schemes for practical implementation are added and illumination invariance integrated. This work is largely based on that by Aach and Kaup [2].

Consider two images Y_1 and Y_2 of an image sequence, with $y_1(k)$ and $y_2(k)$ the grey-level values at pixel k. Now, let $D = \{d(k)\}$ denote the grey-level difference image with $d(k) = y_1(k) - y_2(k)$. Q, the change mask, is a binary image with label $q(k)$ for each pixel k in the image. $q(k)$ either takes the value $q(k) = u$ (unchanged) if the grey-level difference $d(k)$ supports hypothesis H_0 that the change is due to camera noise only, or the value $q(k) = c$ (changed) if it does not support this hypothesis (instead supports alternate hypothesis H_1). The change mask Q is estimated in an attempt to maximise its *posteriori* probability $Pr(Q|D)$, maximum *a posteriori* probability (MAP) estimate.

Assume all the $q(k)$ label values are known except for one pixel element, i. Estimating Q then becomes deciding between $q(i) = u$ and $q(i) = c$. The change mask from $q(i) = u$ is represented by Q_u^i, and that from $q(i) = c$ by Q_c^i. Therefore the decision rule is:

$$\frac{Pr(Q_u^i|D)}{Pr(Q_c^i|D)} \underset{c}{\overset{u}{\gtrless}} t \tag{4.1}$$

where t is the decision threshold and $\overset{u}{\underset{c}{\gtrless}}$ means that the outcome is $q(i) = u$ if the left-hand side of (4.1) is greater-than t, or $q(i) = c$ if less-than t. Using Bayes' theorem, the decision can be written as:

$$\frac{p(D|Q_u^i)}{p(D|Q_c^i)} \underset{c}{\overset{u}{\gtrless}} t\frac{Pr(Q_c^i)}{Pr(Q_u^i)} \tag{4.2}$$

where $p(D|Q)$ denotes the conditional probability of the difference image D given the change mask Q and $Pr(Q_u^i)$ and $Pr(Q_c^i)$ are the *a priori* probabilities for Q_u^i and Q_c^i respectively.

Assume the grey-level differences $d(k)$ are conditionally independent ($p(D|Q) = \prod_k p(d(k)|q(k))$). This is justified in 'unchanged' areas where the observed differences are regarded as caused by camera noise. In 'changed' areas it is technically not justified since the differences are correlated. However, it was experimentally found in [2] that "for the purpose of change detection, these statistical dependencies can in practice be neglected without perceptible deterioration in detection performance"[2]. This leads to the following simplification:

$$\frac{p(d(i)|H_0)}{p(d(i)|H_1)} \underset{c}{\overset{u}{\gtrless}} t\frac{Pr(Q_c^i)}{Pr(Q_u^i)} \tag{4.3}$$

where $p(d(i)|H_j)$ denotes the likelihood for hypothesis H_j, $j = 0, 1$, with respect to pixel i.

The detection algorithm is more reliable if the decision is based on a number of surrounding local grey-level differences $\mathbf{d_i}$ and not just the one at pixel i, $d(i)$. To achieve this, the differences $d(k)$ inside a sliding window w_i, centred at location i, are used. Thus (4.3) can be changed slightly to:

$$\frac{p(\mathbf{d_i}|H_0)}{p(\mathbf{d_i}|H_1)} \underset{c}{\overset{u}{\gtrless}} t\frac{Pr(Q_c^i)}{Pr(Q_u^i)} \tag{4.4}$$

This means that the decision to accept the null hypothesis or not is based on the entire sample $\mathbf{d_i} = \{d(k)|k \in w_i\}$.

In order to implement the above decision rule, some assumptions need to be introduced for the conditional densities and prior probabilities to make the rule more practical.

Looking at the conditional density $p(d(k)|H_j)$, it can be assumed that the grey-level differences obey a zero-mean Gaussian distribution with variances σ_0^2 and σ_1^2 for H_0 and H_1 respectively. This leads to the decision rule being rewritten as:

$$exp\left\{-\frac{1}{2}\left(1-\frac{\sigma_0^2}{\sigma_1^2}\right)\overline{\Delta_i^2}\right\} \underset{c}{\overset{u}{\gtrless}} t\left(\frac{\sigma_0}{\sigma_1}\right)^{N_w}\frac{Pr(Q_c^i)}{Pr(Q_u^i)} \tag{4.5}$$

with

$$\overline{\Delta_i^2} = \frac{1}{\sigma_0^2}\sum_{k\in w_i} d^2(k) \tag{4.6}$$

being the normalised square sum of grey level differences inside w_i. N_w is the size of the window in pixels.

The variance caused by the noise, σ_0^2, is generally much smaller than the variance σ_1^2 due to a changed area ($\sigma_1^2 > 100\sigma_0^2$). This allows the fraction σ_0^2/σ_1^2 to be dropped and (4.5) to be further simplified (taking the logarithm of each side) to:

$$\overline{\Delta_i^2} \underset{u}{\overset{c}{\gtrless}} \underbrace{-2ln\left[t\left(\frac{\sigma_0}{\sigma_1}\right)^{N_w}\right]}_{t_s} + 2ln\frac{Pr(Q_c^i)}{Pr(Q_u^i)} \tag{4.7}$$

The left portion of the right side of (4.7) is the fixed threshold term t_s, while the right portion represents the adaptivity of the threshold due to the *a priori* probabilities.

Non-adaptive threshold

Ignoring the adaptive part of (4.7) and looking only at the global decision threshold t_s, it is seen that it is specified in terms of σ_0, σ_1 and t. It is more practical however, to relate t_s to the rate α of false alarms of the test. Since the normalised square sum $\overline{\Delta_i^2}$ is known to obey a χ^2 distribution with N_w degrees of freedom [2], the threshold value t_s can be determined from

$$Pr(\overline{\Delta_i^2} > t_s|H_0) = \alpha \tag{4.8}$$

by choosing an appropriate false alarm rate. This is a significance test with a significance of α. This non-adaptive threshold has some shortcomings in that it sometimes either fails to detect large sections of the moving object or generates many false alarms.

The choice of the false alarm rate α not only depends on mathematical considerations but is also guided by the above two types of error. False alarms result in excess data which can cause an increased data rate, whereas misses result in a reduction of image quality due to missing information. Since this acquired information is going to be used for further processing and is not transmitted anywhere, the image quality is of more concern and thus higher false alarm rates are generally preferred.

Adaptive threshold

To overcome, or at least minimise, the problems with the fixed threshold detector, properties of the various areas of the image are looked at. Generally, the detected objects tend to be compact in their occupied region with smooth edges, while the errors are small, scattered regions of irregular shape.

By exploiting these properties, *a priori* probabilities can be introduced which increase the probability of smooth regions being detected over irregular ones. This improves the change detection in moving areas while at the same time suppressing false alarms. Gibbs/Markov random fields are well suited to this kind of *a priori* probability problem and are thus used. The *a priori* probability is given by

$$Pr(Q) = \frac{1}{Z}exp\{-E(Q)\} \tag{4.9}$$

where Z is a normalising constant and $E(Q)$ the energy term which assesses the state of the change mask.

The smoothness of the region is evaluated by looking at the *border pixel pairs*. These are a pair of horizontal, vertical or diagonally adjacent pixels which are situated across the boundary from a changed region to an unchanged one. Only the local component is looked at when determining the decision rule — that is, the eight surrounding pixels of the examined pixel, i. The number of horizontal or vertical border pixel pairs is given by $v_B(q(i))$ and the number of diagonal border pixel pairs is given by $v_C(q(i))$. Since there are four horizontally/vertically and four diagonally orientated pairs these values both range from zero to four. Using them the local energy contribution is given by

$$E_L(q(i)) = v_B(q(i))B + v_C(q(i))C \tag{4.10}$$

where the constants B and C are so-called potentials.

After some manipulation of (4.10) (see [2]) and substituting it into (4.9) and then (4.7) one gets the decision rule

$$\overline{\Delta_i^2} \underset{u}{\overset{c}{\gtrless}} t_s + 8(B+C) - 4(m_B^c(i)B + m_C^c(i)C) \tag{4.11}$$

where $m_B^c(i)$ is the number of horizontal/vertical neighbouring pixels which carry the label c (changed). Similarly, $m_C^c(i)$ is the number of diagonal neighbouring pixels with the label c.

Further simplification is possible if it is chosen that $B = C$. In this case the horizontal/vertical and diagonal border pixel pairs are not discriminated between. It was shown through testing in [2] that in practice this simplification produces almost identical results to the bivariate rule (4.11). The simplified univariate rule is

$$\overline{\Delta_i^2} \underset{u}{\overset{c}{\gtrless}} t_s + 16B - 4B\, m^c(i) \tag{4.12}$$

B is chosen based on trials and visual examination of the results. In [2] it was found that the precise value used is not critical and a value of $B = 2.25$ was used.

Since B and t_s are fixed values and $m^c(i)$ only has eight possible values, the results of the right-hand side of (4.12) can be stored in a look-up table for fast processing. This cuts down the processing required at run-time as only the left-hand side needs to be computed and the comparison made.

4.1.2 Illumination Invariance – Collinearity Criterion

Up until now, the test decision has been based on the *normalised square sum of grey-level differences* $\overline{\Delta_i^2}$. However, this does not take into account the problem of illumination change. This is especially prevalent in real-life environments where lighting is not constant and fixed but continually varying. A rapid change in light can cause a noticeable signal variation in an image (both visually and numerically) which is not a relevant change of an object's position.

Aach and Kaup [2] therefore based their change detection on testing whether the signal vectors, $\mathbf{x_1}$ and $\mathbf{x_2}$, from two different images are collinear. In this case, the two different images are the background and foreground images. If the signal vectors are judged to be collinear (or parallel) then there is no change between the two image pixels and if not, then a change has occurred. Of course, in every image there is some degree of noise and so perfect collinearity is not possible. Instead an estimate of the true signal direction, $\mathbf{u}$ (see Figure 4.1), is determined. When Gaussian noise is assumed, $\mathbf{u}$ can be estimated by minimising the sum $D^2 = |\mathbf{d_1}|^2 + |\mathbf{d_2}|^2$ of the square distances $\mathbf{d_i}$ of the observed vectors $\mathbf{x_i}$ [2]. Note that $\mathbf{x_1}$ and $\mathbf{x_2}$ are collinear if the sum of their norms is zero (or at least close to zero for almost collinear vectors).

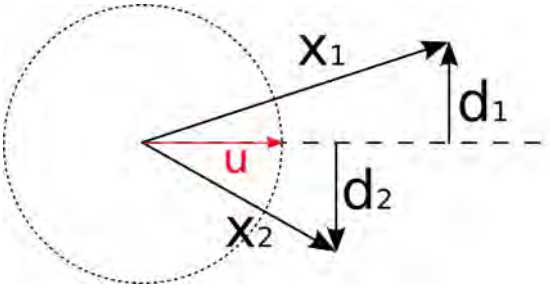


Figure 4.1: Geometric interpretation of collinearity testing of vectors $\mathbf{x_1}$, $\mathbf{x_2}$

By defining the $2 \times N$ matrix

$$\mathbf{X} \stackrel{def}{=} \begin{pmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \end{pmatrix} = \begin{pmatrix} x_1^1 & x_1^2 & \cdots & x_1^N \\ x_2^1 & x_2^2 & \cdots & x_2^N \end{pmatrix} \tag{4.13}$$

with N pixels in the neighbourhood of the considered pixel i, it can be shown (see [2]) that the test statistic D^2 is equal to the smallest non-zero eigenvalue of the 2×2 matrix $\mathbf{X}\,\mathbf{X}^T$.

Thus the decision rule is now

$$D^2 \underset{u}{\overset{c}{\gtrless}} t_s + 16B - 4B\, m^c(i) \tag{4.14}$$

4.1.3 Initial Segmentation Results

For initial testing, a test image sequence of a spinning man is used. This was obtained from [32] and is a *Blue-C demo clip*. It is ideal since the first few frames only show the background before the subject enters the viewing area, which allows a background test frame to be obtained.

The test results, using the background frame as the comparison for the current frame, are shown in Figure 4.2 (a form of background subtraction). The first image, (a), shows the square sum test statistic result. This segmentation of the subject per-

forms reasonably well, producing smooth edges and only misses a little of the right leg. However, it also segments the shadow, which is undesirable. Shadows are relatively indistinct shapes and hence do not change much from one frame to another.

Results from the application of the eigenvector method are shown in Figure 4.2b. Their *illumination–invariant* feature is clearly seen with the subject's shadow not being detected. However, the segmentation is not very smooth and 'noisy' image spots in the background are also detected. The adjustment of various parameters, such as the window size and adaptive constant B, did not help to improve this.

(a)

(b)

Figure 4.2: Background test frame segmentation of turning man, frame 70. The left hand side images show the image overlaid with the background mask to dim out the unselected parts and the right hand side images display the logical background mask. (a) using the square sum test statistic; (b) using the eigenvalue test statistic

4.1.3.1 Improvements to Eigenvector Method

Following on from the above results, further improvements were introduced into the eigenvector method's algorithm.

The first improvement is minor and only entails adding a **variance component**

to the test statistic (see Figure 4.3a). This helps to reduce the overall sensitivity of the change detection and thus almost completely eliminates the background noise seen in Figure 4.2b. However, it also is less sensitive to the foreground subject and hence only detects the most contrasting parts of the moving person. So, while this method improves some aspects of the image, it reduces others and therefore cannot be introduced just by itself.

The second addition is to introduce the use of **colour** in the collinearity testing as opposed to just using the grayscale image values for it. This is achieved by redefining the $2 \times N$ matrix, $\mathbf{X}$ (4.13), to also include the separate RGB colour components of the surrounding pixels as follows:

$$\mathbf{X} \stackrel{def}{=} \begin{pmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \end{pmatrix} = \begin{pmatrix} r_1^1 \; g_1^1 \; b_1^1 & r_1^2 \; g_1^2 \; b_1^2 & \cdots & r_1^N \; g_1^N \; b_1^N \\ r_2^1 \; g_2^1 \; b_2^1 & r_2^2 \; g_2^2 \; b_2^2 & \cdots & r_2^N \; g_2^N \; b_2^N \end{pmatrix} \tag{4.15}$$

This helps significantly to improve the detection of the complete foreground subject, which was lost when the variance was added, while still retaining the background noise reduction benefit (see Figure 4.3b). This does not completely fix the problem as there are still a number of 'gaps' in the foreground segmentation.

Looking at these missing regions throughout the short clip, it was seen that they are all linked to relatively dark regions – especially the dark trouser leg and hair. This led to the **darkness compensation** component being introduced. This problem is one drawback of the illumination-invariant method used. Although the illumination-invariance offers robustness to lightning changes and shadows, it can also affect the detection of dark (close to black) objects. This is because "black can be seen as a low intensity version of any colour" [31] and hence will not trigger segmentation.

This problem is rectified by adding an extra component to the vectors $\mathbf{x_1}$ and $\mathbf{x_2}$ in (4.15). This added component has a fixed value of $\sqrt{O_{dc}}$. The square-root is used simply for convenience as this value is multiplied by itself (almost straight away) when calculating the eigenvalues. This component helps to make the collinearity test more sensitive to differences, especially in dark areas.

The exact value of O_{dc} is chosen experimentally but is usually quite high. For the test image used, a value of 5800 was found to work best. The end result can be seen in Figure 4.3c. Here, it is seen that the segmentation is complete (no gaps) and also fairly accurate (only segments the foreground object) with no background noise.

Since all image sequences do not have the same characteristics, the parameters which define the segmentation need to be tuned for each image set used. Segmentation is controlled by three user defined parameters: the static threshold t_s; the darkness offset O_{dc}; and the importance factor B of the spatio-temporal compactness. The

(a)

(b)

(c)

Figure 4.3: Improvements to the Eigenvector method of segmentation of turning man, frame 70. The left hand side images show the image overlaid with the background mask to dim out the unselected parts and the right hand side images display the logical background mask.
(a) addition of variance term to test statistic; (b) use of colour in the collinearity testing; (c) introduction of the darkness compensation component

.

following steps are followed to tune these parameters:

1. The static threshold t_s is determined with O_{dc} and B set to zero.
2. The darkness offset O_{dc} is increased until a balance between appearing shadows and vanishing holes is reached.
3. Finally, the compactness value B is increased until the foreground regions are smooth and compact.

4.2 Edge Detection

Edge detection is a very important feature in any image analysis and as a result there are many different algorithms and methods for detecting and enhancing edges. Edges are boundaries of objects and shapes and as such aid greatly in the location, segmentation, tracking and identification of objects in a video sequence. While edge detection is a relatively low-level operation, accurate and clear edges are essential and necessary for any higher-level processing.

4.2.1 Edge Detection Methods

Canny edge detection is a very well known method and many consider it to be the standard edge detection algorithm in industry [69]. It was developed in 1983 at MIT by John Canny, and still out-performs many of the newer algorithms today. The main problem with the Canny edge detector (as mentioned in Section 2.1.2) is that it does not use colour and as such may detect a number of edges where there should not be any (shadow edges). For example, it may detect creases on a trouser leg as edges when they are not since the trouser leg is a single object.

This problem has lead to the investigation and use of colour edge detectors which can use the similarity in colour to reject some edges and thus reduce the clutter of the resulting edge map. Following Kehl and Gool's paper [52], colour edge detection in RGB space is used. The method is based on the Difference Vector Edge Detector. The advantage of using the RGB (Red, Green, Blue) colour space is that most images are produced in this space and hence do not need to be converted into some other form before being processed. While there are a number of other colour spaces, such as HSI (Hue, Saturation, Intensity) or LUV (Luminance, Chrominances U and V) which may describe colour more appropriately, RGB is the most convenient and is easy to use in terms of the HumanEva datasets used in this thesis.

There are two main distance metrics which are used to describe colour changes, *Euclidean Distance* and *Vector Angle*. Euclidean distance does not qualify colour as well in RGB space as it does in LUV space. However, in general it is observed that Euclidean distance is sensitive to variations in intensity, but not very sensitive to variations in

hue and saturation. Alternatively, vector angle is insensitive to intensity differences but quantifies hue and saturation differences well [101].

Both of these metrics can be adapted to work with the Difference Vector Edge Detector. This operator calculates the *maximum gradient* across the central pixel, of a 3×3 pixel window, in the four directions — horizontal, vertical, left diagonal and right diagonal.

The Euclidean distance version is written as

$$E_{DV} = \max_{i=1...4} \{\|\vec{v}_i\,(x,y) - \vec{v}_{4+i}\,(x,y)\|\}$$

while the vector angle version is

$$S_{VG} = \max_{i=1...4} \left[\sqrt{1 - \left(\frac{\vec{v}_i^{\,T}\,(x,y) \cdot \vec{v}_{4+i}\,(x,y)}{\|\vec{v}_i\,(x,y)\|\,\|\vec{v}_{4+i}\,(x,y)\|} \right)^2} \right]$$

where $\vec{v}_i\,(x,y)$ is the i^{th} colour vector ($\vec{v}_i = [v_R\ v_G\ v_B]$) around the centre pixel (x,y) (see Figure 4.4).

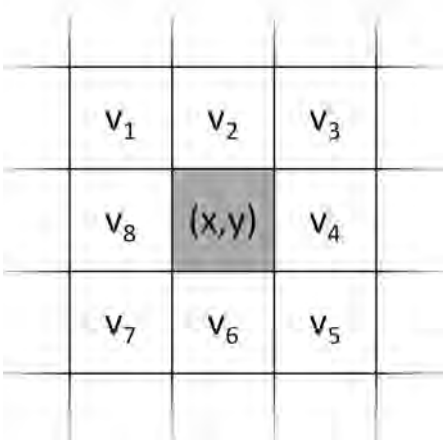


Figure 4.4: Difference Vector Edge Detector 3×3 pixel window

It is also possible to use the metrics with the Vector Gradient Edge Detector which calculates the *maximum distance* between the central pixel and the eight surrounding pixels. However, this method requires double the number of calculations to the Difference Vector Edge Detector and therefore is not used.

4.2.2 Combination Method

Since the Euclidean distance and vector angle metrics together use the intensity, hue and saturation information of a colour image, it is beneficial to combine the two metrics into one single method. This can be done in a number of ways but in this case they are combined using intensity. This is because intensity can be easily calculated by

finding the average of the three RGB components for any pixel. Saturation can also be used but requires conversion into the HSI colour space. The intensity value is used to determine which metric is given more preference. If both pixels have a high intensity it is better to use vector angle while if one of the two pixels has a low intensity it is better to use Euclidean distance.

In order to create a smooth transition between the two extremes, the following sigmoid transition function [101] is used:

$$\alpha(I) = \frac{1}{1 + e^{-slope(I - offset)}}$$

where *slope* and *offset* are application dependent and set experimentally.

Instead of using the sigmoid function for the combining of the metrics, some threshold value x, above which the vector angle metric is used and below which the Euclidean distance metric is used, could have been employed. However, a smooth function, such as the sigmoid function, aids in the transition from one metric to the other by giving different weighting to each metric based on the intensity of the pixel examined. Obviously, at very high and very low intensity values only one of the metrics is used, but in the middle regions, values from both metrics are combined to give the 'probability' of the pixel being on an edge.

The slope and offset parameters determine the 'shape' of the transition function. The offset value sets the midpoint of the intensity range – the intensity value at which one metric over the other start being used. The slope value determines the range over which a combination of the two metrics is used. A larger value means a steeper slope of the transition function and hence a much smaller intensity value range in which both metrics are used. The actual slope and offset values were chosen experimentally with testing as discussed in Section 4.2.5.2.

Since two points are considered each time, the two sigmoids are combined into a single transition function:

$$\rho(I_1, I_2) = \sqrt{\alpha(I_1) \cdot \alpha(I_2)}$$

This all results in the intensity-based combination of the Difference Vector operator being represented by:

$$C_{DV} = \max_{i=1\ldots4} \left(\begin{array}{l} \rho(I_1, I_2) \sqrt{1 - \left(\dfrac{\vec{v}_i^T(x,y) \cdot \vec{v}_{4+i}(x,y)}{\|\vec{v}_i(x,y)\| \, \|\vec{v}_{4+i}(x,y)\|} \right)^2} + \\ \\ (1 - \rho(I_1, I_2)) \cdot \|\vec{v}_i(x,y) - \vec{v}_{4+i}(x,y)\| \end{array} \right)$$

4.2.3 Initial Edge Detection Results

The above three edge detection algorithms (RGB Euclidean Distance Difference Vector, RGB Vector Angle Difference Vector and intensity-based Combined Difference Vector) were all implemented in Matlab and tested on a number of images. To start, the offset and slope values have been set at 75 and 0.1 respectively, following the values used in [101]. The effect these values have on edge detection, and hence their optimal values, is examined later in this section.

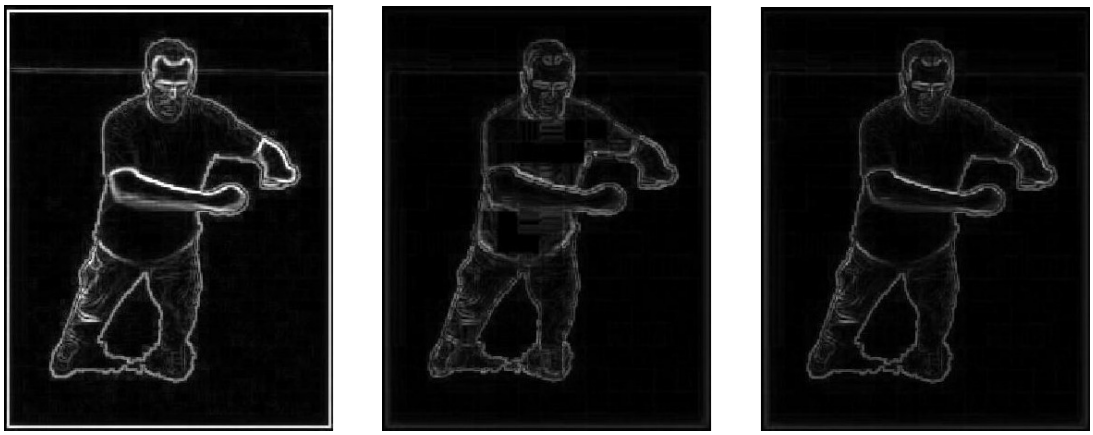
The three algorithms have been applied to the original picture (Figure 4.5) of a standing man. The results are shown in Figure 4.6. The RGB Euclidean Distance Difference Vector edge detector, Figure 4.6a, produces good distinct edge lines but a lot of unwanted clutter as well (see the right trouser leg). The RGB Vector Angle Difference Vector edge detector, Figure 4.6b, produces less clutter but some of the edge lines are indistinct and also a little 'blocky' (see hairline and right forearm regions). The intensity-based Combined Difference Vector edge detector, Figure 4.6c, shows the best results. The edges are distinct and there is only a little clutter on the right trouser leg.

Figure 4.5: Original picture of a man in some pose

4.2.4 Edge Refinement

While the edges are used in their edge intensity form in the optimisation, it can be difficult to judge the effectiveness of the edge detection algorithm when only looking at the edge intensity. Thus, edge refinement is used to get a distinct edge line instead of the rough area of the edge intensity. This edge refinement is used to evaluate the effect of changes in the edge detection algorithm. It must be emphasised again that this edge refinement is not used in the final edge detection used in the optimisation.

Initially the use of a simple threshold was looked at. However, this did not produce good results as the boundary was not always a single width line, was not a very



(a) (b) (c)

Figure 4.6: Edge detection: (a) RGB Euclidean Distance Difference Vector (b) RGB Vector Angle Difference Vector (c) Intensity-based combined Difference Vector (offset = 75 and slope = 0.1)

continuous line and either missed some important edges or included irrelevant ones.

These problems were caused by the fact that a simple threshold looks at the image as a whole and thus uses a global threshold or limiting value. Instead, it is much better to find *local maxima* and then classify them as edges or not depending on whether they fall into a *threshold region* (in between an upper and lower threshold value). Once this is achieved, the edge can be thinned to a single-line contour where necessary. Inspiration and method to achieve this was obtained by closely studying the Canny edge detection implementation used by Matlab.

Figure 4.7 shows the result of edge refinement on the initial combined edge detection result from Figure 4.6c. It can be seen that the edges are single-line width and largely continuous. There are double lines around the hand region due to the black border on the initial image (Figure 4.5).

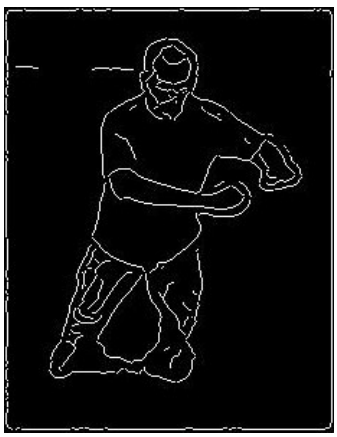


Figure 4.7: Edge detection refinement

4.2.5 Improved Edge Detection Results

As previously mentioned, one of the main advantages to using the intensity-based combined RGB Difference Vector edge detector was its resistance to clutter edges — those caused by shadows and wrinkles in clothing. But looking at Figure 4.7 this appears to have only been partially achieved. The clutter is especially still prominent on the trouser legs but is of a low intensity, as noted from Figure 4.6c. It therefore does not have much of a effect in the edge detection matching during optimisation. However, this clutter can be improved by adjusting a few of the parameters of the algorithm as well as pre-filtering.

4.2.5.1 Pre-filtering

Pre-filtering is used to smooth the image and essentially reduces the sharpness or detail of the image. However, it is mainly the fine detail which is reduced, which helps to eliminate some clutter while still retaining the overall detail needed to obtain the edges.

Three different filters were implemented for testing with the results shown in Figure 4.8. From Figure 4.8a, which shows a non-filtered result, it is apparent that pre-filtering does substantially improve the edge detection results. The custom Gaussian filter (Figure 4.8b) gives the best result, with the *fspecial* Gaussian filter (Figure 4.8d) and simple smoothing filter (Figure 4.8c) also improving the result, but not optimally.

It can be seen that the edge refinement result shown in Figure 4.7 already used pre-filtering.

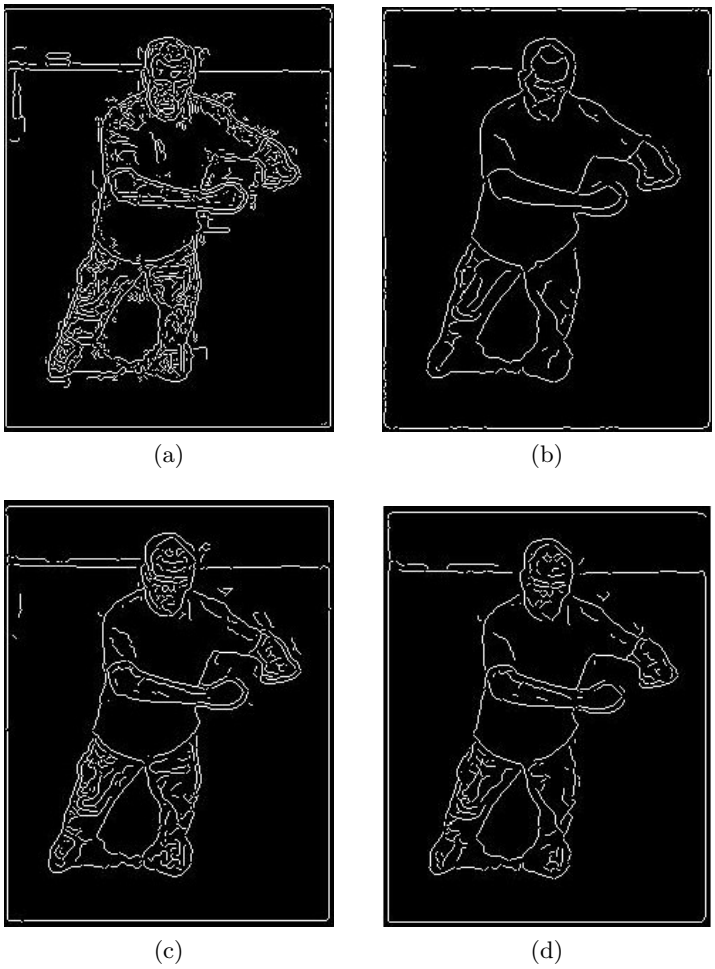
4.2.5.2 Parameter Adjustment

There are two main parameters to adjust (apart from the threshold value), the slope and offset values from the sigmoid function. The best values for these parameters were determined experimentally.

First looking at the **slope** (while keeping the offset constant), it was observed that noticeable changes only occurred when slope varied from approximately 0.05 to 0.15 (see Figure 4.9). The best result was achieved with a slope of approximately 0.1. In Figure 4.9 the images with slopes of 0.1 and 0.15 appear very similar. However, in some other test images it was observed that a slope of 0.1 performed better.

Moving on to the **offset**, noticeable changes occurred up to a value of about 150 from close to 0 (see Figure 4.10). Once again, the best result was in the middle range with an offset value of approximately 75. This provided the best compromise between reduced clutter while retaining important, unbroken edges.

Values above this generally resulted in an increase of unwanted edges while values below led to an unwanted break in some edges and an increase in clutter in other places.



(a) (b)

(c) (d)

Figure 4.8: Pre-filtering: edge detection results of original image with (a) no filtering applied (b) custom Gaussian filter (size 17) (c) small simple smoothing filter (size 7) (d) *fspecial* Matlab Gaussian filter (size 9, applied 3 times)

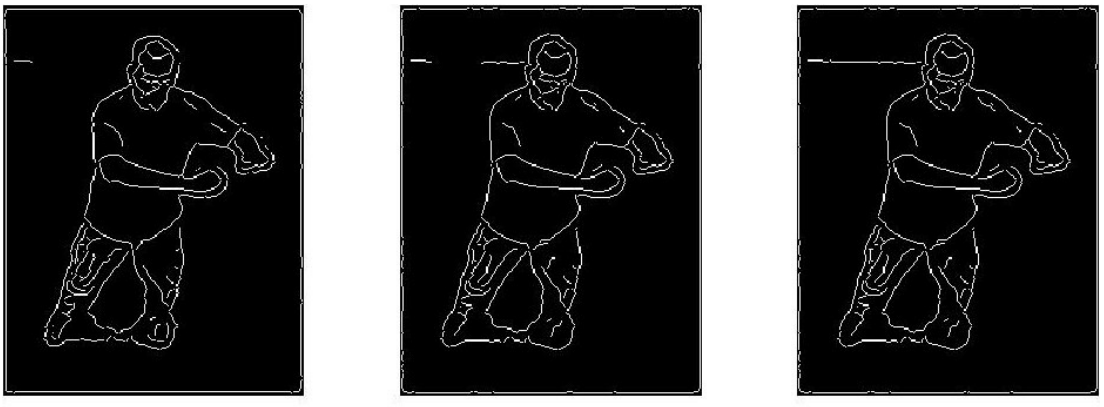


Figure 4.9: Slope parameter adjustment

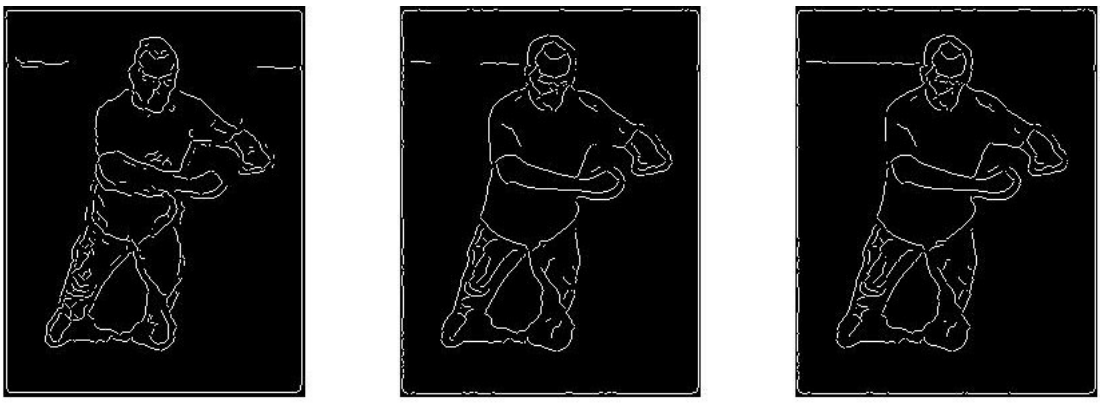


Figure 4.10: Offset parameter adjustment

4.3 Volumetric Reconstruction

So far, the only concern has been for 2-D images and videos. However, the world we live in is three-dimensional and therefore to properly model subjects in it, 3–D data is needed. This data is acquired though multiple simultaneous views of a scene, each from a different angle.

Segmentation, to identify the foreground subject mask in each image, is performed before the masks are backprojected. The intersection of these cones, from the calibrated image views, 'carves' a volume (known as the *visual hull*) in which the object exists (see Figure 4.11). This method is known as *shape-from-silhouette* and produces more accurate results if more cameras and angles of view are used. However, Van den Bergh *et al.* [97] have shown that a visual hull computed from four or five appropriately placed cameras produces reasonable and generally sufficient results for pose estimation.

This basic method is fairly uniform for most volumetric reconstruction methods. It is the different shape-from-silhouette algorithms used that distinguish the various methods. While there are many different variations, the main methods are *voxel-based*, *polyhedral visual hull* and *space carving and photo consistency.*

Voxel Reconstruction The working volume is divided into little cubes, called voxels. These voxels are then projected into the image plane of all camera views. Any voxel which does not lie inside the silhouette of every camera is discarded and considered to be outside the volume of the object. This is a very simple and popular method and has been used for body tracking by, amongst others, [67] and Theobalt *et al.* [94]. This reconstruction can, however, be computationally very expensive due to all the projection calculations.

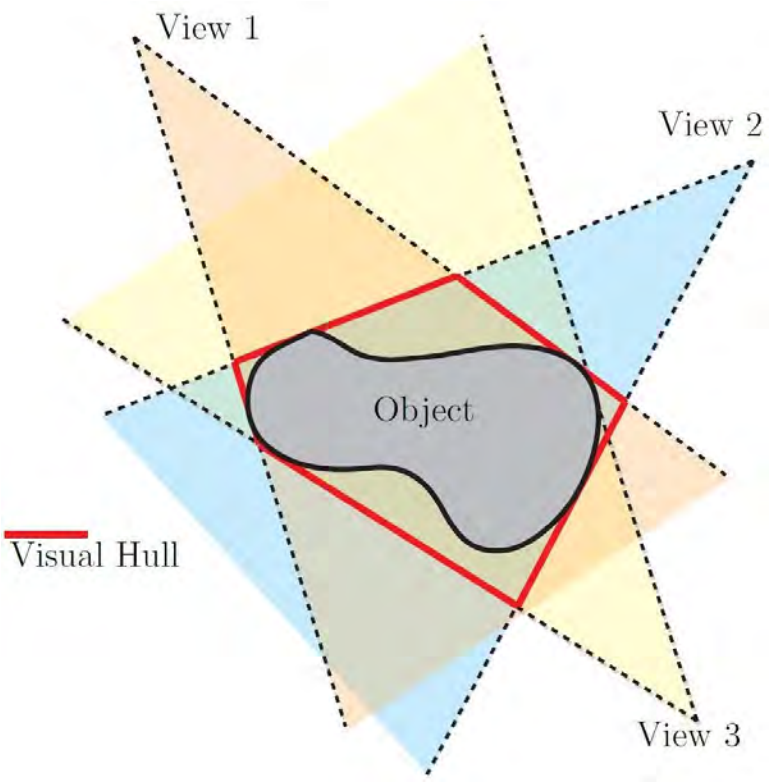


Figure 4.11: Visual hull of object constructed from intersecting silhouette projections [97]

Polyhedral Visual Hull The visual hull is computed from polygonal representations of the silhouettes and uses constructive solid geometry to calculate the intersections. This method does not use discrete volumes, so yields better accuracy than voxel-based approaches and is able to operate in real-time, [25]. However, it requires perfect silhouettes due to the complex geometric calculations. Thus, poor silhouettes can result in incomplete and incorrect surface models.

Space Carving and Photo Consistency This method uses silhouettes as well as photo consistency to create the visual hull, Seitz and Dyer [81]. Voxels which are not photo-consistent in all camera views are discarded. This approach assumes constant illumination and Lambertian reflectance. It also needs to perform multiple plane-sweep passes making it both complex and fairly slow.

For human body tracking the visual appearance of the hull is not of importance but the structure of the human body needs to be retained. Speed is also important and perfect silhouettes are generally never available. Thus, voxel-based reconstruction is used in this system.

4.3.1 Voxel-based Reconstruction

Voxel-based methods are a popular [19, 52, 67] but computationally expensive shape-from-silhouette approach.

For this technique, the capture space that contains the subject is partitioned into small cubes called *voxels*. These voxels are each projected into the image planes of all the camera views. Those voxel projections which lie inside all (or a certain pre-determined number) of the foreground masks are marked as part of the object while those that do not fit this criteria are discarded. In this way the voxels belonging to the subject are found and the basic volume constructed.

A lot of computation is needed to achieve the reconstruction and this is only one frame. To speed up the process, Kehl *et al.* [52] proposed that, since the cameras are fixed, the voxel projection can be done off-line. Here a fixed look-up-table (LUT) is stored for each pixel of each camera view, pointing to the voxels which project onto that pixel (see Figure 4.12).

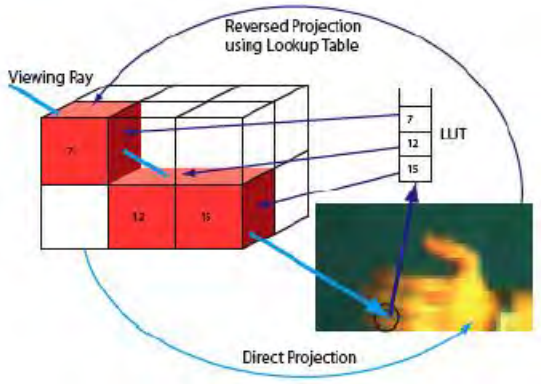


Figure 4.12: Depicting the LUT for each image pixel which points to its projected voxels [97]

If all the pixel bitmasks for a particular voxel are 1 (indicating that they lie inside the foreground mask), then the voxel must be part of the object. These comparison checks are quick to perform at runtime with most of the processor intensive calculations being done at startup or some previous initialisation time.

Another advantage is that instead of having to recalculate the value of every voxel in each frame, they can instead be updated from the previous frame. A voxel will only change its status (as either part of the object or not) if one of its pixel projections changes. Since there is a direct map from image pixel to voxel, only those voxels which correspond to a changed pixel mask need to be recalculated — further improving processing time.

4.3.2 Computing the Look-up-Table

The initial idea was to somehow project a ray from a pixel into the 3-D space and record which voxels it 'encounters' or 'sees'. However, this was complicated as a voxel is not a single point but has a volume and also the ray would diverge as it moved further away from the pixel. It was realised that it would be much easier to start at a voxel and project backwards to the image to see which pixel sees that particular voxel (and in particular the centre of the voxel). This is performed using (3.16) from epipolar geometry and camera calibration.

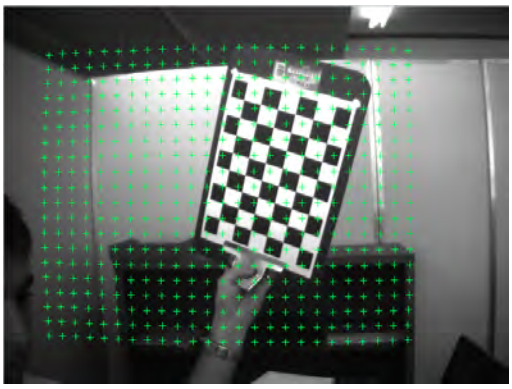
However, the problem discovered with this is that not every pixel is labelled as 'seeing' the front-facing pixels of the 3-D rectangle. This is because, as mentioned, the voxels have a volume and only their centre point is being projected back to the pixels. Voxels closer to a camera have a larger projection surface in the image plane and cover many pixels. If every pixel covered is included a larger number of look-ups would be required. To overcome this, Cheung *et al.* [19] use a Sparse Pixel Occupancy Test (SPOT) algorithm which uses 2 uniformly distributed samples out of every 10 pixels in the voxels projected area. In [49] the projected area is described in more detail by projecting the outline of the cube and diagonals joining opposite corners into the image plane. The pixels these lines correspond to are marked as belonging to the voxel. If any of these pixels do not fall inside the silhouette, the voxel is removed from the hull.

After some study, it was determined that this extra work was unnecessary and, if appropriately sized voxels are used, only the centre of each voxel need be used for projection into the image plane. This is because it is the voxel labelling that is important, not the pixel, and every voxel does correspond to some pixel in a camera. It does result in the voxel-hull edges of the detected object not being as smooth as they could be, if more pixels were used to determine which voxels are excluded, but with an appropriately small voxel size this is not too noticeable.

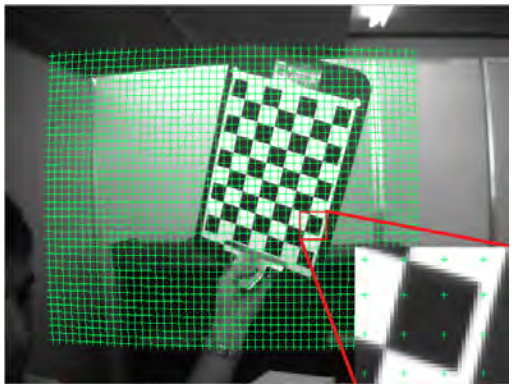
Figure 4.13 displays the above problem when only the centre of the voxel is used to determine corresponding pixels. Even with a voxel size of 4mm (Figure 4.13c) there are still about five unmarked pixels in between every labelled pixel for the given z plane, about 70cm from the camera. Figure 4.14 shows the pixels corresponding to the centre of voxels from several z-planes. This illustrates that the pixel correspondence density increases as the voxels are further away from the camera. The cameras used to capture the HumanEva data are at least 1 metre away from the moving subject which allows a voxel size of 20mm to be comfortably used for the reconstruction.

4.3.3 Initial Testing

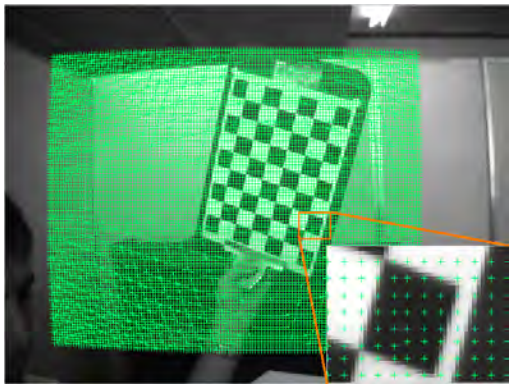
For initial voxel reconstruction verification a block-like shape was 'drawn' in each camera view as the supposedly detected object (Figure 4.15). The separate voxels hulls produced from each image view are displayed in Figure 4.15c & 4.15e, with the left



(a)



(b)



(c)

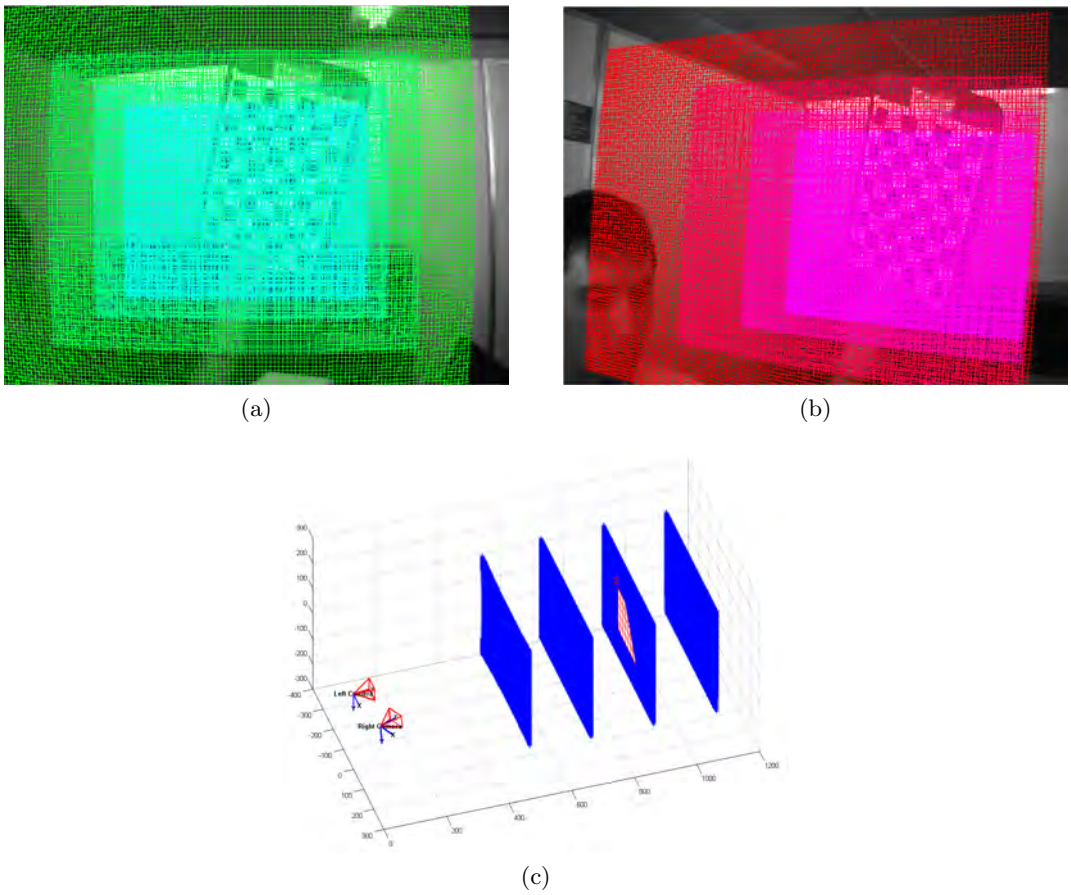
Figure 4.13: Effects of different voxel sizes on pixel correspondence using voxel centre. (a) voxel size 20mm; (b) voxel size 10mm with zoomed patch; (c) voxel size 4mm with zoomed patch.

camera view reconstructions shown in green and the right in pink. The common voxels (i.e. detected object) is displayed in blue in Figure 4.15d & 4.15f. This verifies that the projection and voxel reconstruction is working correctly.

Figure 4.15 also demonstrates the effect the voxel size has on the volumetric reconstruction quality. A larger voxel size means fewer voxels to fill the capture volume and hence faster processing and less storage memory required. But it also translates to a poorer quality resolution. Thus a compromise is found between the processing and memory requirements and the reconstruction quality. A voxel size of 20mm provides this balance with the HumanEva datasets.

4.3.4 Texturing

While texturing does improve the look of the voxel hull reconstruction, its main purpose is to help improve the tracking as described in Section 6.1.3. In the 3-D reconstruction, body parts may end up being joined. This is due to inaccurate reconstruction or simply because the body parts are actually close together and touching. In these



(a) (b)

(c)

Figure 4.14: Pixels corresponding to the centre of voxels from several z-planes as viewed in (a) the left camera and (b) the right camera with (c) 3-D plot of viewed z-planes.

cases it becomes difficult to distinguish between the parts. However, very often these parts are of a different colour. Therefore, by making use of colour information with the volumetric reconstruction, these body parts may be distinguished from one another and the ambiguity removed.

To achieve this colouring, a representative colour is assigned to each surface voxel. This colour is the average colour of all the image pixels seen by the voxel (or rather voxel centre). To efficiently accomplish this a depth-buffer algorithm is used together with a 4-component colour representation (the three RGB colour components with the fourth component keeping track of the number of contributing views). This allows a single pass algorithm to be used since, if a previously coloured voxel is detected as being occluded from some camera view by another voxel, the pixel's colour can be removed from the occluded voxel. The depth-buffer handles the occlusion detection while the 4-component colour mix allows a pixel's colour to be removed.

Intrinsically this algorithm only colours the surface voxels since all interior voxels are occluded. Thus it is able to identify these required surface voxels at the same time.

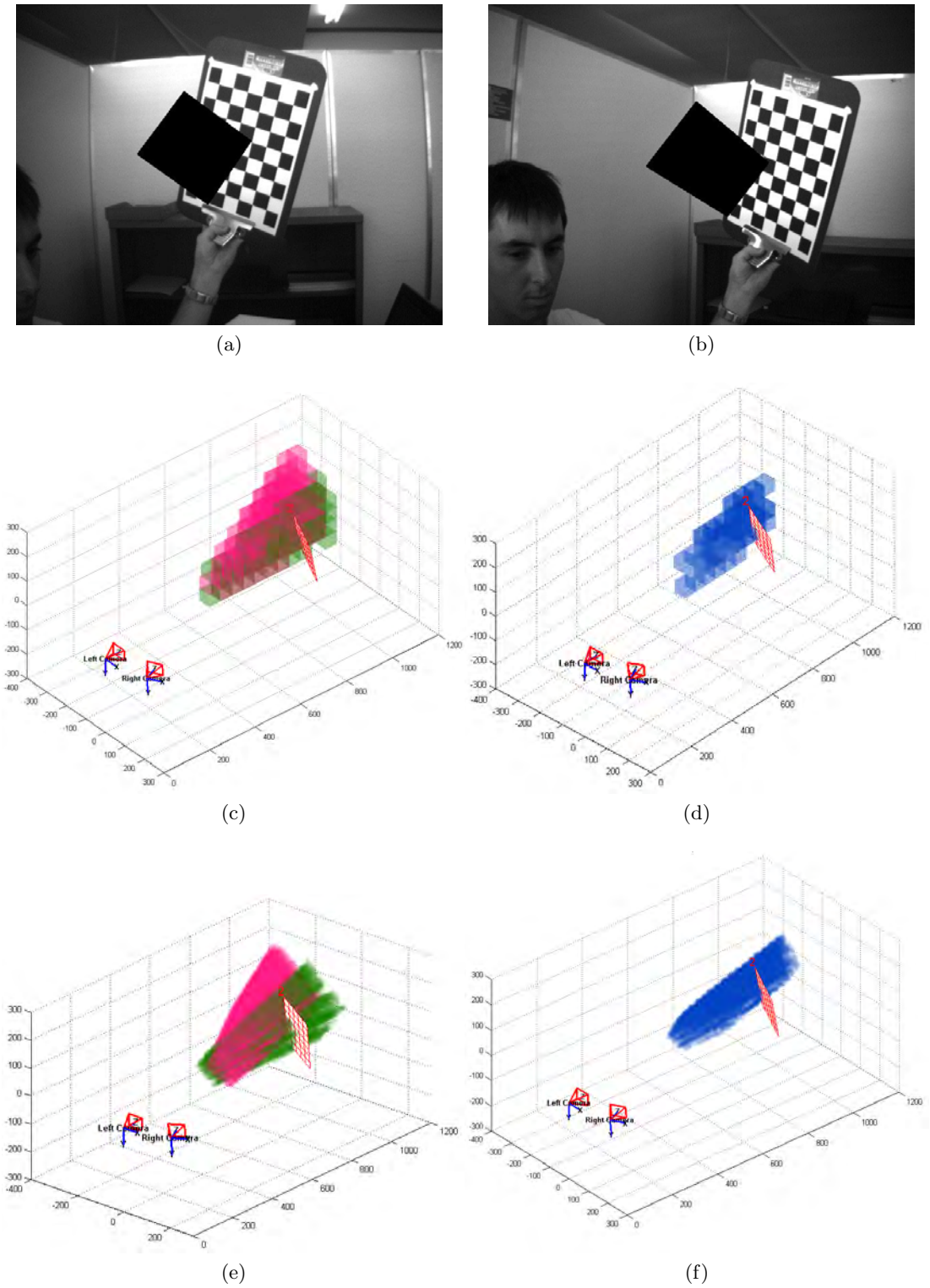


Figure 4.15: 3-D voxel hull reconstruction of fabricated block-like shapes in (a) left and (b) right camera views for:
(c) left and right camera view and (d) common voxel reconstructions for voxels of size 50mm
(e) left and right camera view and (f) common voxel reconstructions for voxels of size 10mm.

4.4 Results

As mentioned in Section 3.3 the HumanEva datasets are used for testing. Below, various processing results using these datasets are displayed. Initially HumanEva II results are considered before some HumanEva I results are looked at.

4.4.1 HumanEva II

The HumanEva II datasets are captured using 4 colour cameras and consist of two subjects in two sequences (S2 and S4). In each case a combination of actions are performed throughout the sequence — walking, jogging and balancing.

The dimensions of the LUT are 3×No. of Voxels expected per pixel×vertical pixels of image×horizontal pixels of image×No. of cameras in setup. So for the HumanEvaII setup this is $3 \times 200 \times 656 \times 490 \times 4$.

For the HumanEvaII setup, a $3m \times 2m \times 4m$ capture space is examined with 20mm cubed voxels used. This corresponds to a total of 3 000 000 voxels.

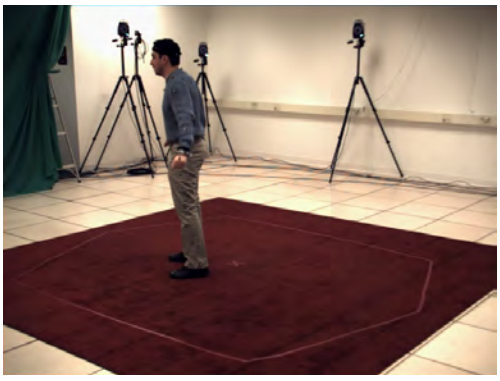
Figure 4.16 shows the input images captured for frame 1049 of the HumanEva II S2 sequence. It can be seen that the four cameras surround the subject, giving a view of him from all angles. Figure 4.17 displays the results from the three processing steps performed on the input images — segmentation (top), edge detection (middle) and volumetric reconstruction (bottom). The segmentation and edge detection results are shown from the four camera views (cropped) while three virtual camera views are selected to display the volumetric reconstruction results.

The results produced are very good with a clear silhouette of the subject extracted. The edge detection emphasises the important edges while minimising the detection of clutter/shadow edges and the volumetric reconstruction produces a good representation of the subject's shape and colour.

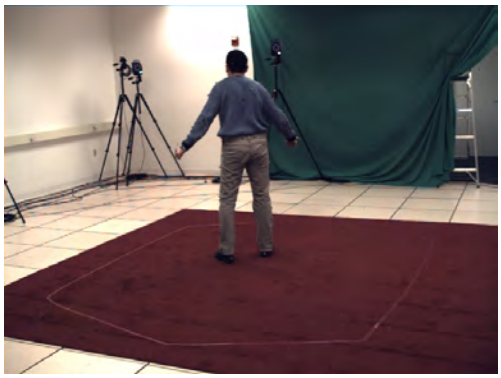
Figure 4.18 shows the input images captured for frame 820 of the HumanEva II S4 sequence. As with the S2 sequence, the subject is visible in all surrounding camera views. The results displayed in Figure 4.19 show the processing outputs from segmentation (top), edge detection (middle) and volumetric reconstruction (bottom).

Once again clear silhouettes are produced while important edges are detected and displayed with a higher intensity white than the less important clutter edges which are shown a lot dimmer. The bottom row shows different virtual views of the volumetric reconstruction. Again, reasonable results are produced with the shape and colour of the subject represented well. However, in the third- and fifth-from-left views, a 'third' leg is also seen. This is known as a ghost leg and is a result of the low number of surrounding cameras. If there were a few more cameras this ghost leg would be able to be eliminated, or carved away, by another view of the scene. Due to the position of the legs in relation to the four cameras, there is a 'blind spot' where views of the legs

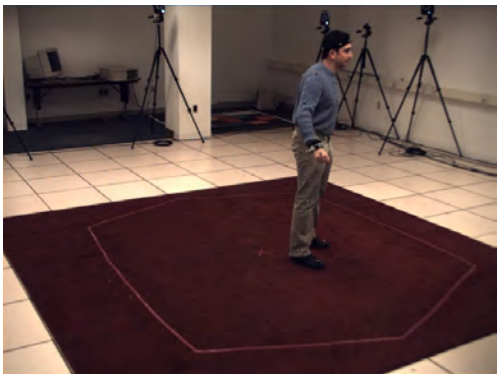
camera 1



camera 2



camera 3



camera 4

Figure 4.16: Images captured from frame 1049 of the HumanEvaII S2 dataset from the 4 camera views.

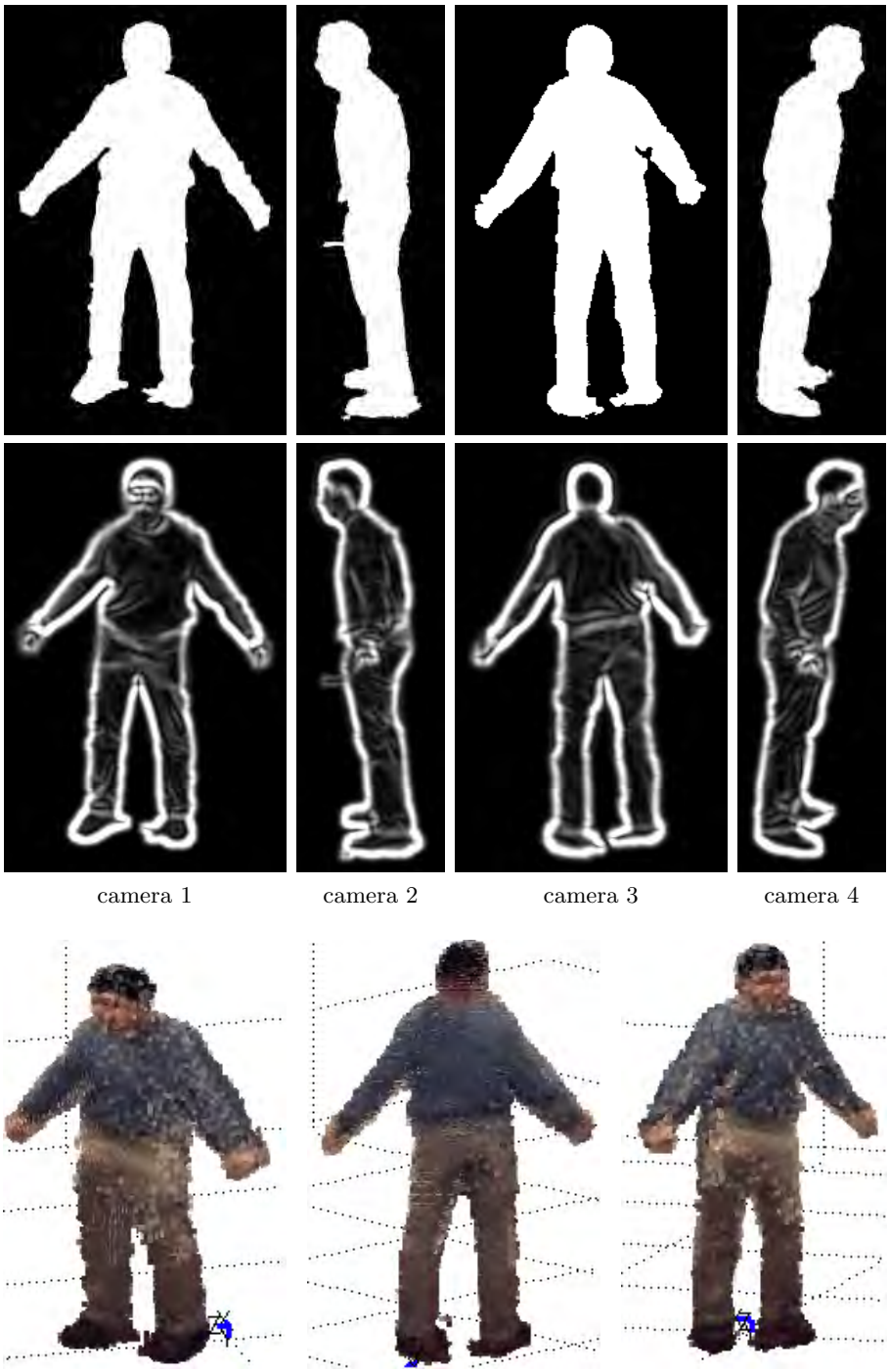
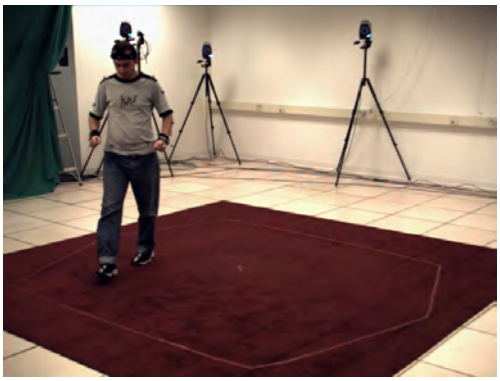


Figure 4.17: Processing results for the HumanEvaII S2 dataset frame 1049. Top: silhouettes, Middle: 2-D edges, Bottom: 3-D volumetric reconstruction.

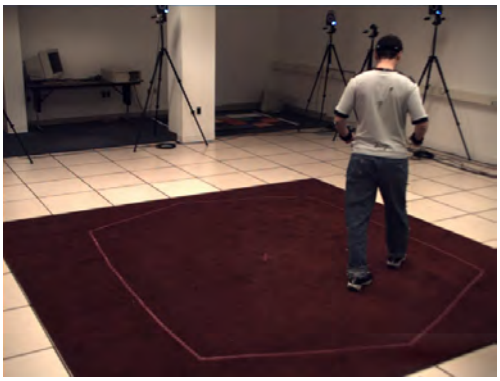
overlap in more than one place. This ambiguity results in the ghost leg being produced. This does not adversely affect the tracking and is not a permanent feature throughout the tracking sequence — it is only seen in certain frames. Ideally it would be better to have more available camera views, but 4 cameras do produce adequate results for the required purpose.

camera 1



camera 2

camera 3



camera 4

Figure 4.18: Images captured from frame 820 of the HumanEvaII S4 dataset from the 4 camera views.

4.4.2 HumanEva I

The HumanEva I datasets were captured prior to HumanEva II and use software, as opposed to hardware, synchronisation. Seven cameras are used to capture the data. However, only 3 are colour, with the remaining being greyscale cameras. Four subjects (S1 – S4) performing a variety of actions, such as walking, jogging and throwing/catching, are captured by all the cameras.

Figure 4.20 shows the captured images from synchronised frame 180 of the HumanEvaI S1 walking 1 sequence. The 3 colour camera and 4 greyscale camera views are clearly visible. This is the disadvantage of the HumanEvaI dataset over the HumanEvaII. Since colour cameras are almost exclusively used now, all the processing

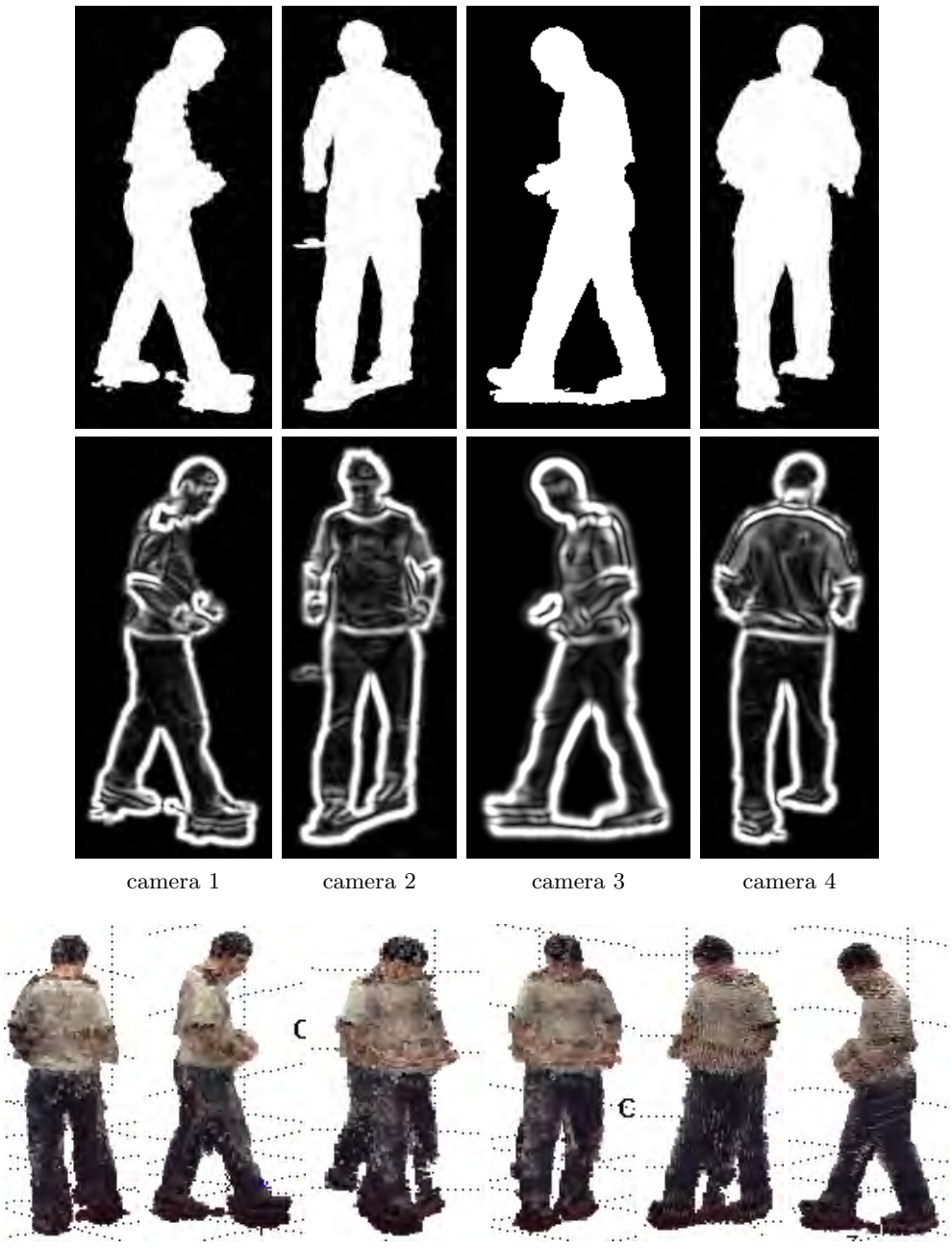


Figure 4.19: Processing results for the HumanEvaII S4 dataset frame 820.
Top: silhouettes, Middle: 2-D edges, Bottom: 3-D volumetric reconstruction.

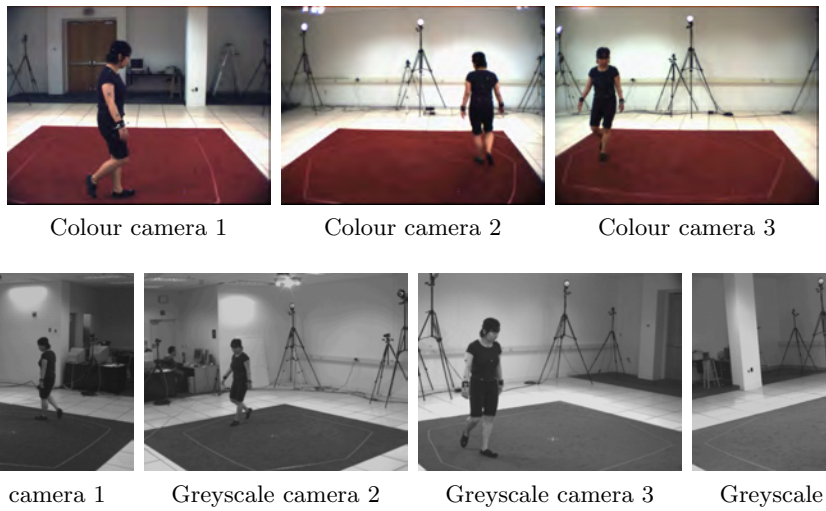
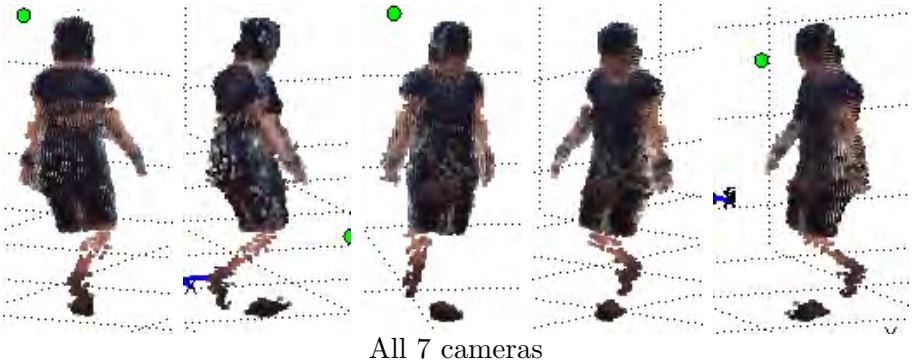
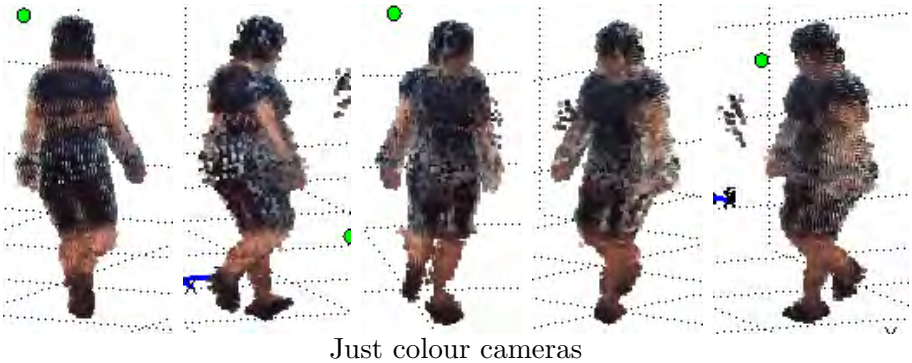


Figure 4.20: Images captured from synchronised frame 180 of the HumanEvaI S1 dataset from the 7 camera views.

algorithms used in this system make use of the colour information they provided and are optimised for it. Hence, poor results are achieved with the greyscale camera images.

Figure 4.21 displays these processing results with the first and second rows giving the segmentation and edge detection outputs. It is clearly seen that the colour images (cam 1 – 3) result in much better results than the greyscale images (cam 4 – 7), especially with the segmentation. This poor segmentation also affects the volumetric reconstruction as seen in the bottom three rows of Figure 4.21. When all 7 camera views are used for the volumetric reconstruction, parts of the body are carved away, which should not be, because they are 'missing' in some silhouette views. On the other hand, if only the three colour views are used, very little detail is provided and the resulting reconstruction contains a lot of extra bulk (compare the last view in each row) because the number of views is insufficient to produce good results. Therefore, an alternative is instead used to perform the volumetric reconstruction; information from 6 out of the 7 camera views is used. This means that if any body part is missing in one segmentation view it will not affect the result as long as the remaining views all contain it. This reconstruction is shown in the bottom row and resulted in better detail than when using only the colour cameras without carving away too much of the body, as was the case when all cameras were used. This is by no means perfect but provides better results than the other two options. Using information from 5 of the 7 cameras was also examined. However, this resulted in reduced detail and similar results to using just the colour cameras was produced.


All 7 cameras


Just colour cameras

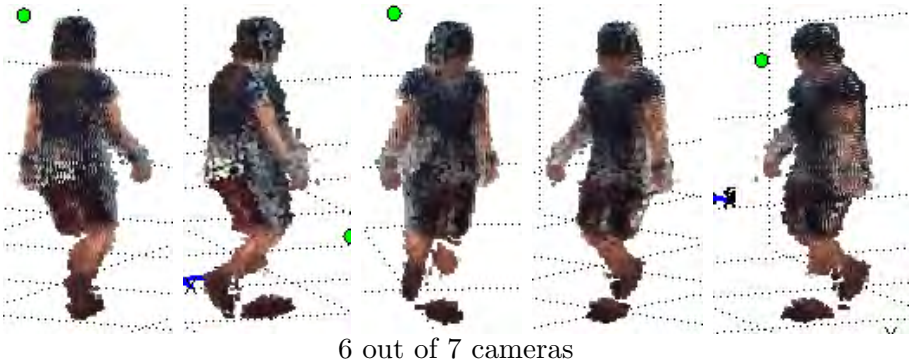

6 out of 7 cameras

Figure 4.21: Processing results for the HumanEvaI S1 dataset frame 180.
Top: silhouettes, Middle: 2-D edges, Bottom: 3-D volumetric reconstruction using: all 7 cameras, just colour cameras, 6 out of 7 cameras

4.5 Summary

This chapter provides the basic building blocks of the system with all future stages using and depending on the results from this processing.

The segmentation is based on an illumination-invariant method with *a priori* probabilities, colour collinearity testing and darkness compensation added to it to help improve the extracted silhouette results.

With the subject of interest segmented from each image, edge detection can be applied which provides 2-D information to be used for tracking. A RGB colour space Difference Vector Edge Detector is used to achieve this. It combines the Euclidean-distance and vector-angle metrics to improve results by making the detection sensitive to both intensity as well as hue and saturation changes. The use of colour helps to reduce the clutter by decreasing the detection of shadow edges, such as trouser creases, which are not true edges.

The volumetric reconstruction provides 3-D data to the system. A voxel-based method is used to carve a visual hull from the intersection of the back-projected silhouettes. A pre-computed look-up-table helps to improve the processing time of the reconstruction along with silhouette and voxel update between each frame — instead of a completely new calculation each frame. Colour is also introduced to the reconstruction to disambiguate between joined body parts and thus help improve tracking.

Various processing results performed on the HumanEva I and II datasets are displayed. All the processing algorithms are designed and optimised for colour images. As a result, the outputs from the HumanEva II datasets, which consist of 4 colour camera images, outperform those from the HumanEva I datasets, which consist of 7 input views but of which only 3 are colour images. The input images and resulting segmentation, edge detection and volumetric reconstruction results are presented for certain frames of the HumanEvaII S2 and S4 subjects and the HumanEvaI S1 subject. The segmentation and edge detection performs well, especially with the colour images, while the 3-D volumetric reconstruction performs adequately. Better volumetric results would be obtained if more colour camera views of the scene were available. This would help improve the reconstruction detail and eliminate unwanted ghost limbs.

Chapter 5

Body Model

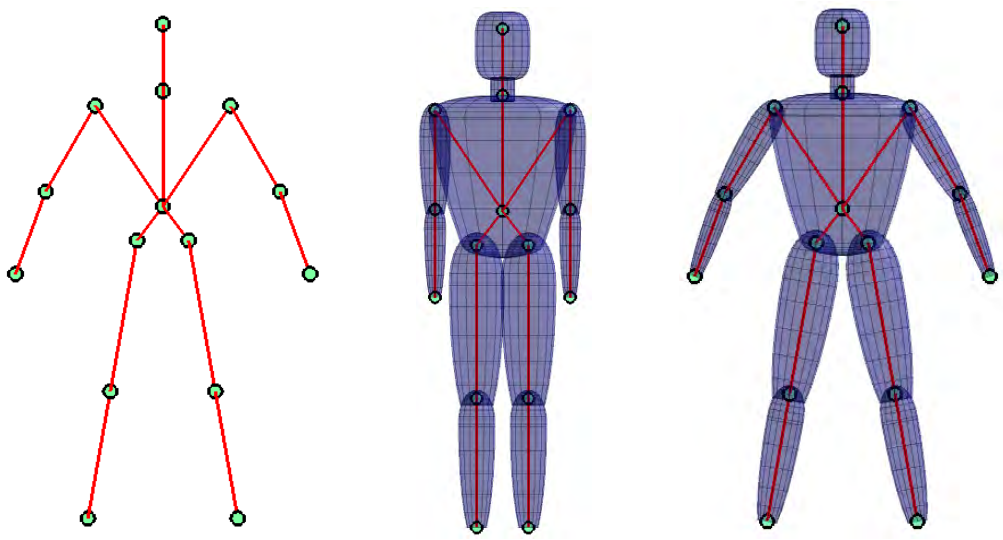
In order to match and track human pose, a human body model is needed which can be matched to the image data acquired in Chapter 4 . While there are many different shapes and combinations of shapes which can be used to represent the human body, the models built with them must be concise and simple to allow for fast matching and tracking. For this reason, many models make use of simple shapes such as cylinders, ellipsoids, cones or spheres. Even though only a coarse model is required, since many of the tracked features are noisy anyway, models created from these shapes do not always resemble the human body shape as closely as desired. For this reason, superellipsoids are used in this research to create the body model (Section 5.1).

Superellipsoids offer a very good trade-off — giving acceptable body shape approximation with a compact numeric representation. Other possible shape models were discussed in Section 2.1.1.2. The body model also serves a secondary purpose of providing a more attractive way of displaying the results. Instead of using a stick figure to show the pose tracking output a more realistic superellipsoid human model is used.

Keeping with this idea of simplicity, a simple skeletal structure is used around which the superellipsoids are fitted (as used by Kehl *et al.* [52]). This skeleton consists of 10 joints and 24 degrees-of-freedom, see Figure 5.1a. These 24 parameters consists of 3 rotations for each shoulder and hip, 2 rotations for the head, 1 rotational bend for each knee and elbow and 3 rotations and 3 translations for the torso.

Finer body detail such as hands, feet and facial features are ignored because they do not affect the pose. Kinematic constraints (Section 5.2) are applied to this skeleton to improve the detection of valid poses.

This superellipsoid body model is based on that presented by Kehl *et al.* [52].



(a) (b) (c)

Figure 5.1: Superellipsoid body model:
(a) the skeleton structure, (b) the body parts joined together, (c) the body model (with skeleton structure) in a pose.

5.1 Superellipsoid Body Model

5.1.1 Superellipsoid Calculations

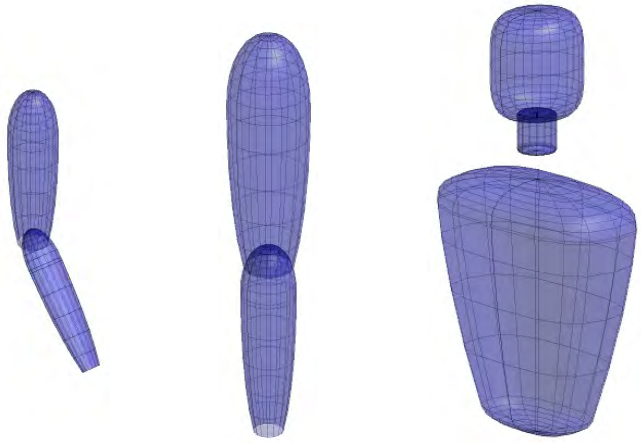
A superellipsoid is a special case of a superquadratic and is the result of crossing two orthogonal ellipses. The superellipsoid's representation is controlled by two shape parameters $0.0 < \epsilon_1, \epsilon_2 < 2.0$, one for each ellipsoid ; three scaling parameters $\alpha_1, \alpha_2, \alpha_3$ defining each axis size ; and the longitude $-\frac{\pi}{2} \leq \eta \leq \frac{\pi}{2}$ and latitude $-\pi \leq \omega \leq \pi$ angle parameters.

These parameters are used in (5.1) which gives a 3-D point $\mathbf{p}(\eta, \omega)$ on the superellipsoid surface.

$$\mathbf{p}(\eta,\omega) = \begin{bmatrix} \alpha_1 cos^{\epsilon_1}(\eta) cos^{\epsilon_2}(\omega) \\ \alpha_2 cos^{\epsilon_1}(\eta) sin^{\epsilon_2}(\omega) \\ \alpha_3 sin^{\epsilon_1}(\eta) \end{bmatrix} \tag{5.1}$$

In order to create the various body shapes from superellipsoids, the controlling parameters were initially 'played around with' to determine which kind of shapes can be created. In order to get the correct tapering shape for the arms and legs two superellipsoids are joined together with the lower one truncated for either the join or limb ending (see Figure 5.2a & b). The representation of the torso was more complicated, and required the use of three superellipsoids — one each for the top and bottom and

one for the main tapering midsection (see Figure 5.2c). The head and neck were the easiest parts to shape and only required a single superellipsoid each. So the end result is a complete human body model created from twenty-one superellipsoids, or only 147 parameters (Figure 5.1c).



(a) (b) (c)

Figure 5.2: Superellipsoid body parts:
(a) upper and bent lower arm, (b) upper and lower leg and (c) torso with neck and head above.

The size and dimensions of the body model are based on an 'average' human's body dimensions obtained from [79]. Three scaling parameters are also used to adjust the width, breadth and length of all the superellipsoids to better match the tracked subject's size. While this does not provide an exactly matching body model, it does provide a sufficiently similar model for the required tracking purposes.

Once created, the body parts needed to be correctly positioned with respect to one another. Initially, when created, they are centred around the origin $(0, 0, 0)$ and therefore their joint positions and hence offset amount need to be calculated in some manner. Again, this was done on a trial-and-error basis by placing a part in roughly the correct area and then moving it slightly until the desired position was achieved. The offset and joint position to achieve this was then worked out and recorded. The final result is shown in Figure 5.1b. It should be stressed again that this model is not an exact or perfect match with any tracked subject. However, the matching image data is noisy anyway and thus the model is sufficient for the required purposes.

5.1.2 Superellipsoid Transformations

The setting–up of the body model is, however, still not complete. The slightly more difficult task of adding movement to the body still remains. Two different types of movement are required — rotation and translation.

Translation is by far the easier of the two to achieve as all that is needed is for a scalar value to be added to all x-, y- and z-vector co-ordinates to offset everything by a certain amount.

Rotation is more tricky as certain limbs have dependencies which are linked to them, such as the upper-arm which has the lower-arm linked to it. If these parts are rotated, their dependencies need to be moved accordingly so that, for example, the arm remains attached as it should be. There are two options for this movement:

1. the dependent limb is rotated by the same amount and then translated so that it stays attached — thus remaining in the same position *relative* to the moved upper limb

2. the dependent limb is only translated so that it remains attached — thus keeping the same orientation it had before the move

Both methods have their advantages and disadvantages and each can achieve the same pose positions, just sometimes in more steps depending on the specific pose. The first option is used here since it gives the more natural movement of the body (at least in my opinion). If one moves an upper limb, one expects the lower limb to move with it and remain in the same relative position — which is what this option achieves.

Each superellipsoid is described by its set of surface points. The rotations and translations are performed on these surface points to move the superellipsoids as needed. The rotation matrix $\mathbf{R}$ is split into three rotations about the x, y and z axes such that

$$\mathbf{R} = \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z \tag{5.2}$$

with

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) \\ 0 & sin(\theta_x) & cos(\theta_x) \end{bmatrix}$$

$$\mathbf{R}_y = \begin{bmatrix} cos(\theta_y) & 0 & sin(\theta_y) \\ 0 & 1 & -0 \\ -sin(\theta_y) & 0 & cos(\theta_y) \end{bmatrix}$$

$$\mathbf{R}_z = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\theta_x, \theta_y, \theta_z$ are the angles of rotation about the x-,y- and z-axes respectively.

The translation matrix $\mathbf{T}$ is simply the translation in the three directions x, y and z.

$$\mathbf{T} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{5.3}$$

Putting these together the local transformation matrix is obtained

$$\mathbf{M}_{loc} = \begin{bmatrix} \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z & \mathbf{T} \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{5.4}$$

These local transformations contain the model parameter values for the joint angles (ie. body part rotations) and torso translations where applicable.

Multiple local transformations can be chained together to translate a child body part in relation to all of its parent parts via

$$\mathbf{M} = \mathbf{M}_{locA}\mathbf{M}_{locB}\mathbf{M}_{locC}$$

So, for example, if the transformation matrix of the lower left arm is required, it would depend on all of its parent transformations (upper left arm and torso), its own transformation and also its initial rotation-point offset transformation . Thus

$$\mathbf{M}_{LAl} = \mathbf{M}_{locT}\mathbf{M}_{locUAl}\mathbf{M}_{locLAl}\mathbf{M}_{initLAl}$$

These calculations are all performed within a function. Thus, whenever a body part needs to be rotated, numerous parameters including the part name, angle of rotation, axis of rotation and rotation origin are passed to the *rotateSE_M* function. This function takes care of the rotation and translation of the specified body part as well as all the other required dependency movements.

5.1.3 Superellipsoid Contour

To perform the tracking, the human body model needs to be aligned with the image edges and 3D voxel reconstruction. To achieve this, two key pieces of information are needed from the superellipsoid surface:

1. Selection of random, *evenly-spaced surface points* for surface matching
2. Identification of *non-occluded contour points*, from different camera views, for edge matching

Operation (1) is fairly easily achieved. The initial construction of the superellipsoids produces a trigonometric sampling of surface points which creates a concentration of points on more curved regions and would thus result in more points being chosen in

those regions. To correct this problem and allow a more uniform sampling of points, the angle step size in the middle superellipsoid region (the flatter part with less points) is decreased to increase the number of sample points. While this is not a perfect solution as the points are not all exactly evenly spaced, it does solve the problem adequately in the easiest manner.

Operation (2) is more complicated, but can be split up into smaller parts. The first set of these parts deals with identifying the contour points, while the second set handles the occlusion detection of the contour points, both of which are detailed below.

5.1.3.1 Contour point identification

In order to identify the contour points (the rim or edge of a superellipsoid) the surface points are split into those that **can** be seen from a specific view and those that **cannot**. This is achieved by obtaining the surface normal directions for each superellipsoid surface point

$$\mathbf{n}_d(\eta,\omega) = \begin{bmatrix} \frac{1}{\alpha_1} cos^{2-\epsilon_1}(\eta) cos^{2-\epsilon_2}(\omega) \\ \frac{1}{\alpha_2} cos^{2-\epsilon_1}(\eta) sin^{2-\epsilon_2}(\omega) \\ \frac{1}{\alpha_3} sin^{2-\epsilon_1}(\eta) \end{bmatrix} \tag{5.5}$$

(with the same parameters as Equation (5.1))

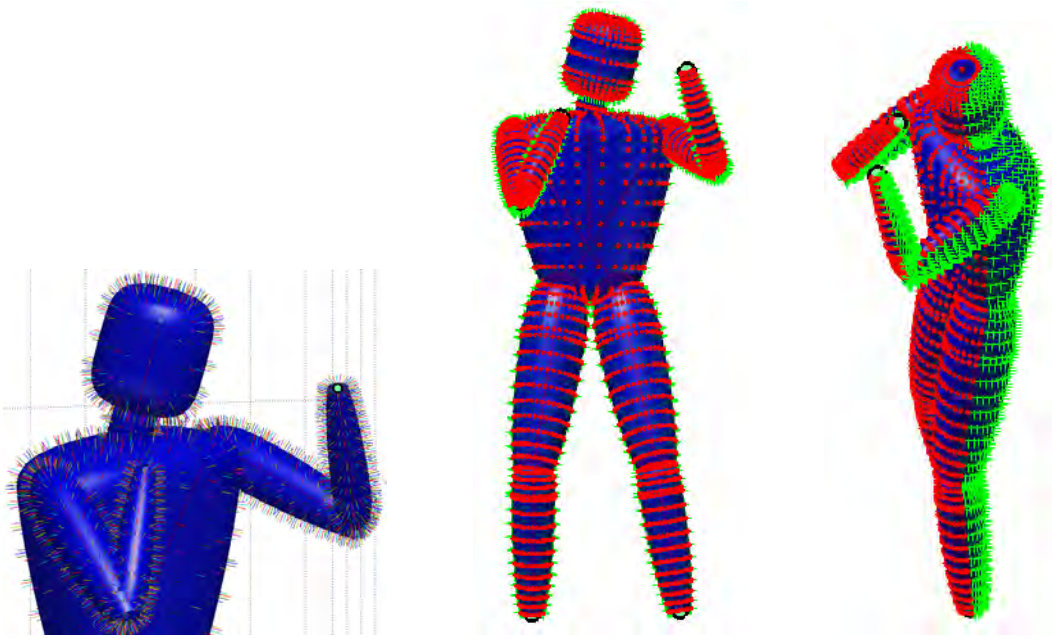
and then solving the equation

$$\mathbf{n}_d(\eta,\omega) \cdot \mathbf{v}(\eta,\omega) = 0 \tag{5.6}$$

for the dot product of the surface normal direction and the viewing ray $\mathbf{v}(\eta,\omega)$, in the initial *superellipsoid co-ordinate system*. However, Equation (5.6) can also be calculated with both values in the *world co-ordinate system* and since the superellipsoid surface data is already being translated into the world co-ordinate system it is simple enough to translate the normal direction data at the same time.

Once both sets of points are in the same co-ordinate system, the closed form solution for Equation (5.6) gives the points on the rim of the superellipsoid, assuming orthographic projection. However, the cameras produce too much perspective distortion for this method to produce accurate results and solving the equation for specific edge points is also challenging. Instead, the result of the dot-product for all the regularly sampled points (from operation (1)) is used. This is achieved by determining the visible, Equation (5.6) ≤ 0, and invisible, Equation (5.6) > 0, points — see Figure 5.3.

The rim/outline points of the superellipsoids are then identified by finding the points between which the sign changes, along all vertical and horizontal meridian lines, and



(a) (b) (c)

Figure 5.3: (a) A close up of the normal vectors from superellipsoid surface points. All visible surface points shaded red and points hidden from view shaded green – (b) from camera orientated front-on view and (c) from a side view.

interpolating between these two points to find the exact contour points. This can be seen in Figure 5.4. There are two things to note in this figure. Firstly, looking at the shoulder region of Figure 5.4a, both horizontal and vertical extrapolated contour points can be seen. Secondly, all contour points are shown whether they are "inside" the body or not. This can be seen in Figure 5.4b and 5.4c at the top of the legs, on the arms by the shoulders and even by the knees where pink dots can be seen inside the body parts.

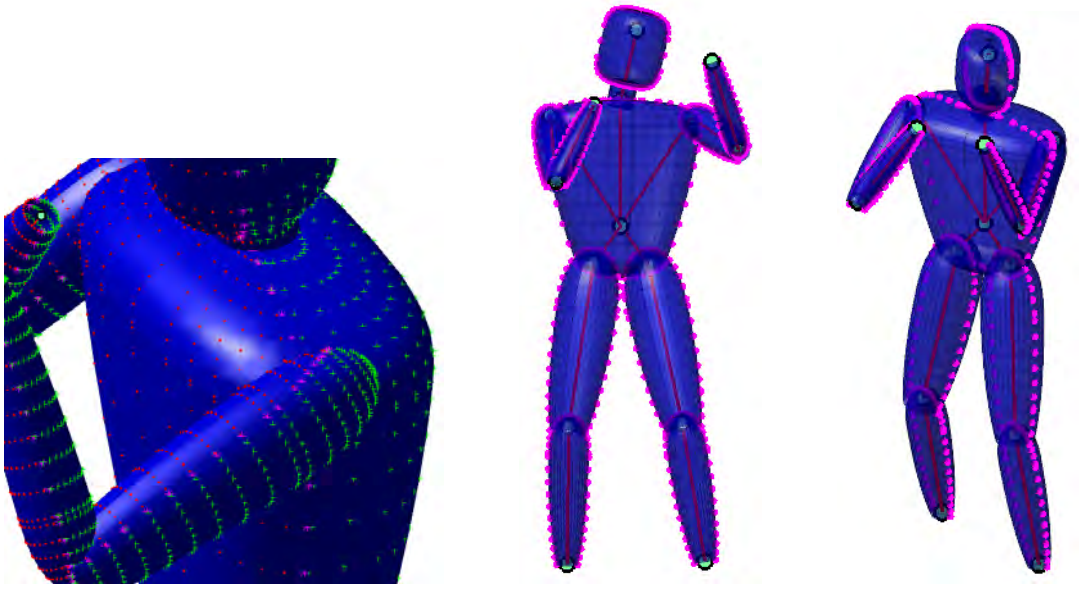
5.1.3.2 Occlusion handling

Once the contour points are identified, it needs to be determined which of these points are directly visible and which are blocked from view (occluded) by some body part closer to the viewing camera. This can be achieved by computing the intersection point between the viewing ray from a point $\mathbf{r}_{\varepsilon_1}(\eta, \omega)$ on superellipsoid ε_1

$$\mathbf{I}_{\varepsilon_1}(\eta, \omega, \lambda) = \mathbf{r}_{\varepsilon_1}(\eta, \omega) + \lambda \mathbf{v}_{\varepsilon_1}(\eta, \omega) \tag{5.7}$$

and the surface of superellipsoid ε_2. However, these intersection points cannot be directly calculated so instead the superellipsoid *inside-outside* function $F(x, y, z)$ is used:

$$F(x, y, z) = \left[\left(\frac{x}{\alpha_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{\alpha_2} \right)^{\frac{2}{\epsilon_2}} \right]^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{\alpha_3} \right)^{\frac{2}{\epsilon_1}} \tag{5.8}$$



(a) (b) (c)

Figure 5.4: (a) A close up of the interpolated contour points (pink $*$) between the visible (red $\bullet$) and hidden (green $+$) surface points.
The contour outline points (pink) from (b) a camera orientated front-on view and (c) a side view.

This function returns a value that is

< 1 for points inside the superellipsoid,

$= 1$ for points on the superellipsoid surface and

> 1 for points outside the superellipsoid.

Since the *inside-outside* function operates by testing points in the initial superellipsoid reference frame of ε_2, the potential intersection vector $\mathbf{I}_{\varepsilon_1}(\eta, \omega, \lambda)$ needs to be transformed (rotated and translated) into that reference frame. To easily accomplish this, whenever a superellipsoid body part is transformed in any way, its rotation and translation matrices are updated accordingly and stored. These transformation matrices are then used to inversely rotate and translate the intersection vector back into the initial reference frame before testing points along it.

It would be impossible to test every point along the vector — it is infinite — so a bounding-box region around the superellipsoid is created. The bounding-box is used since intersection points with the bounding-box can be calculated. It is just intersections with a superellipsoid which cannot be determined. If the vector intersects the box then only the points inside the box need to be tested by the *inside-outside* function. In addition, a half-point testing method is used to reduce the maximum possible number of tests required since it has the greatest chance of being within the superellipsoid near the centre. Thereafter, the result is used to update either the top

or bottom bounding value accordingly.

There are a few other special cases which can be used to speed up the performance of the algorithm. If the vector does not intersect the bounding box, it cannot intersect the superellipsoid, so no inside-outside test needs to be performed. Also, if the contour point of ε_1 already lies within ε_2, which is easily tested, no further computation need be performed.

The final step in detecting the non-occluded contour points, is to determine whether the intersection point of the superellipsoid along the ray vector actually lies in between the camera and the contour point or not. If it does lie in between, then it obviously occludes the point; but if not, then it cannot occlude the point since it must be behind it. This test is performed by calculating the *Euclidean distance* from the camera origin to each of the points of interest — the contour point and the superellipsoid intersection point. If the intersection point is in front of the contour point (i.e. its Euclidean distance is less), then the contour point is marked as occluded.

The test procedure steps are summarised as follows:

```
Input: point of interest x

  if point not inside superellipsoid then
      Test if ray-vector through x intersects bounding box
      if intersect with bounding box then
          Test if ray-vector through x intersects with superellipsoid (using half-point
  method)
          if intersect with superellipsoid then
              Test if intersection occurs in front of x
              if in front then
                  Mark as occluded
              end if
          end if
      end if
  end if
```

Figure 5.5: Pseudocode summarising the occlusion test procedure steps

This is performed for each superellipsoid body part and all contour points not on that superellipsoid. The above pseudo code shows the nested nature of the testing which ensures that as few tests as possible are performed each time.

The results of this non-occluded contour point detection is shown in Figure 5.6. The first detail to note is that there are no contour points 'inside' the body such as at the top of the legs, on the arms by the shoulders or by the knees as previously seen in Figure 5.4. The second detail to note is that the contour is correctly shown in both

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

Figure 5.6: Non-occluded contour points shown from a front-facing ((a) & (c)) and slightly offset ((b) & (d)) camera view.
(a) & (b) have a fairly forward-facing orientated camera while (c) & (d) have a more top-view orientated camera.

Figure 5.6a and 5.6c — the outline is seen as the outline from the intended camera view.

Looking closely at Figure 5.6b, it is possible to observe 'missing' contour points on the upper arms. This is because they are occluded by the lower arms in the given camera view. Similarly, the missing contour points on the upper legs and lower right leg of 5.6d are due to their occlusion by various body parts, but especially the left arm from the given camera view.

5.2 Kinematic Constraints

Kinematic constraints are one of the main constraints used in model-based tracking systems. These are limitations imposed on the range of motion at specific joint locations. This helps to eliminate anatomically invalid postures which allows the pose estimation stage to more reliably converge to the correct pose; thus increasing the reliability of pose tracking. The downside, however, is that they can be very complex and therefore difficult to implement.

Various kinematic constraint approaches were examined in Section 2.1.1.1. In this system, very simple hard limits are imposed on the individual angles/parameter used to represent the joint rotations or bends of the body model. This gives a minimum and maximum value in which the joint parameter can fall. While this does not account for

the interdependency of joints or overlapping of limbs, it does help to restrict movement to more acceptable poses.

Table 5.1: Constraints for all the body parts in a specific direction of movement along with their parameter index.

Body Part	Axis of Rotation & Description	Parameter Index	Movement Direction	Constraints (◦)	
				Min	Max
Upper Leg left (ULl)	X - Yaw	1	+ right	-45	15
	Y - Pitch	2	+ forward	-60	110
	Z - Roll	3	+ anti-clockwise	-50	50
Upper Leg right (ULr)	X - Yaw	4	+ right	-15	45
	Y - Pitch	5	+ forward	-60	110
	Z - Roll	6	+ anti-clockwise	-50	50
Lower Leg left (LLl)	Y - Flexion	7	- back	-150	0
Lower Leg right (LLr)	Y - Flexion	8	- back	-150	0
Upper Arm left (UAl)	X - Pitch	9	+ down	-135	0
	Y - Yaw	10	+ forward	-50	150
	Z - Roll	11	+ anti-clockwise	-50	50
Upper Arm right (UAr)	X - Pitch	12	+ up	0	135
	Y - Yaw	13	+ forward	-50	150
	Z - Roll	14	+ anti-clockwise	-50	50
Lower Arm left (LAl)	Y - Flexion	15	+ forward	0	150
Lower Arm right (LAr)	Y - Flexion	16	+ forward	0	150
Head (H)	X - Yaw	17	+ left	-45	45
	Y - Pitch	18	+ back	-40	60
Torso (T)	Rotation X	19	+ top moves left	mod(*angle* ,360)	
	Rotation Y	20	+ top moves back	mod(*angle* ,360)	
	Rotation Z	21	+ anti-clockwise	mod(*angle* ,360)	
	Translation X	22	+ back	-	
	Translation Y	23	+ right	-	
	Translation Z	24	+ up	-	

The constraints used in the system are shown in Table 5.1 and were implemented using the SMD Equations (6.31) and (6.32).

5.3 Summary

The body model serves two purposes. Its primary purpose is its use in matching 3-D surface and 2-D edge points for tracking alignment. However, it can also be used for display purposes to improve the visual output by using a more realistic-appearing human model than a plain stick figure. Superellipsoids are used to create this model and provide good body shape approximation with simple calculations. An underlying 24 parameter skeleton controls the body movement and pose. These actions are performed by local transformation matrices which include both translation and rota-

tion movements. Matrix ‘chaining’ handles the subsequent movement of all dependent body parts. Very basic kinematic constraints are included in the model to improve the detection of valid poses by imposing a hard limit on all joint angle motions.

Chapter 6

Optimisation

Section 4.2 dealt with 2-D edges which were obtained from silhouettes, while a 3-D voxel hull carving was explained in Section 4.3. These are the observations, or *observed data*, all obtained from processed visual input. Chapter 5 provides the superellipsoid body model which gives the *model data*.

Optimisation involves finding the most likely pose, given the available information, such that the error between the observed and model data is minimised. This is done by adjusting the model's pose parameters, $\mathbf{p}$, so that the body model is brought into agreement (or as close as possible given the errors and noise in the system) with the observations. This is achieved by first defining an objective function (Section 6.1) and then secondly using this and some optimisation scheme to search for the best configuration.

Stochastic Meta Descent (Section 6.2) forms the basis of the optimisation but a few additional approaches and frameworks are also tested. These are SMD with a Hierarchical approach (Section 6.3), SMD with Pose Prediction (Section 6.4) and SMD inside a Particle filter — Smart Particle Filter (Section 6.5).

6.1 Objective Function

The objective function is a measure of how well the model data agrees with the observed data. Since two types of input data are used the objective function can be split into two parts. The 3-D section, $f_s(\mathbf{p})$, deals with the *surface* alignment between the points on the model surface and the surface voxels. Similarly, the 2-D section, $f_e(\mathbf{p})$, handles the *edge* alignment between the edges from the silhouettes and the contour surfaces from the model.

The objective function, $f(\mathbf{p})$, is defined as the sum of errors between the model

features $\mathbf{x}$ and the observed features $\bar{\mathbf{x}}$.

$$\begin{aligned} f(\mathbf{p}) &= w_s f_s(\mathbf{p}) + w_e f_e(\mathbf{p}) \\ &= w_s \sum_{N_s} \delta_s(\mathbf{x}, \bar{\mathbf{x}}) + w_e \sum_{N_e} \delta_e(\mathbf{x}, \bar{\mathbf{x}}) \end{aligned} \tag{6.1}$$

The scaling factors, w_s and w_e, are used to balance the influence of the edge and surface points.

The error term $\delta(\mathbf{x}, \bar{\mathbf{x}})$, in essence, calculates the squared difference error between the model and observed features. However, to minimise the effect of outliers the Leclerc error potential is applied to the distance measure $e(\mathbf{x}, \bar{\mathbf{x}})$. The Leclerc error potential is defined as:

$$\rho(z) = 1 - e^{\frac{-z}{v}} \tag{6.2}$$

where v is a constant parameter that controls the interval for the squared distance that is considered 'normal' or the average value of z. This error potential is computed for each feature individually

$$\delta(\mathbf{x}, \bar{\mathbf{x}}) = \rho(e(\mathbf{x}, \bar{\mathbf{x}})) \tag{6.3}$$

6.1.1 Surface Alignment

The first half of the objective function (6.1) deals with the matching and alignment of the surface points. In Section 5.1 the body model was created with a set of surface points $\mathcal{T}$, while the set of observed surface voxels $\mathcal{V}$ was computed in Section 4.3. The set $\mathcal{T}$ is generally smaller than the set $\mathcal{V}$.

The obvious measure to use is the summed square distances between the model surface points and their closest voxels. This requires a nearest-neighbourhood search of all the voxels (or at least the nearby voxels) for each surface point to find its closest voxel. This search can be computationally expensive. However, stochastic optimisation (discussed in Section 6.2) allows a small subset of points to be used for the tracking, instead of all the points, while still retaining the optimisation's robustness.

Given that $\mathbf{x} \in \mathcal{T}$ and its corresponding closest voxel $\bar{\mathbf{x}} \in \mathcal{V}$ is found, the squared distance error is calculated as

$$e_s(\mathbf{x}, \bar{\mathbf{x}}) = \frac{1}{2} \left(\frac{\|\mathbf{x} - \bar{\mathbf{x}}\|}{\sigma} \right)^2 \tag{6.4}$$

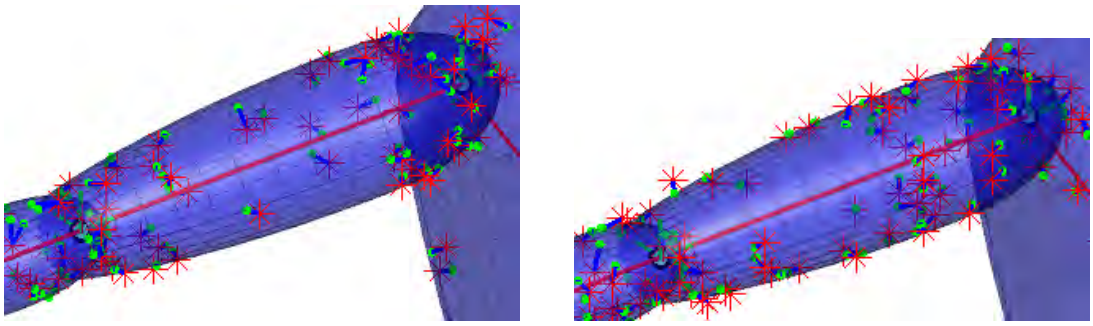
where σ is a scaling factor which corresponds to the limb diameter at point $\mathbf{x}$ on the model. The limb diameter 'normalises' the distance measure and makes values more comparable between different cues and limbs of varying sizes.

6.1.1.1 Surface Alignment Process

This surface alignment process is illustrated in Figures 6.1 – 6.2.

Figure 6.1: Superellipsoid body model inside coloured surface voxels.

Figure 6.1 shows a pose with the body model inside the coloured voxels. It is with these voxels that selected surface model points are matched.



(a) (b)

Figure 6.2: Views of the upper left arm at two different runs showing a different random set of selected model surface points (green dot) and matched voxels (red asterisk).

Figure 6.2 shows close up views of the upper right arm with selected model surface points (green dot) and matched voxels (red asterisk). A blue line joins the corresponding matched points. Note that a voxel may be matched to more than one surface point if it is the closest for both points. This is observed a couple of times when studying the figure closely.

Figure 6.2 also demonstrates that a different random set of surface points is selected for each run. This is seen by comparing the two views and their different selected points

(green dots). The reason for this stochastic sampling is explained in Section 6.2.

6.1.2 Edge Alignment

A set of contour points $\mathcal{C}$, for a specific camera view and current pose, is returned in Section 5.1.3 along with their normal directions $\mathbf{x}_n$. This set excludes all occluded points so only non-occluded contour edges are considered. The observed edge points $\bar{\mathbf{x}}$ are identified in Section 4.2 which returns the edge points set $\mathcal{E}$.

Once again a distance metric is used to measure the alignment between the observed and predicted edge points. Instead of finding the closest edge point along a contour point's normal (which has a fairly large risk of fitting the incorrect edge), edge points with high image gradient magnitude and/or gradient direction parallel to contour normal are found. This is computed, for edge pixel $\bar{\mathbf{x}}$ and its gradient vector $\bar{\mathbf{x}}_n$, via the gradient score

$$s(\mathbf{x}_n, \bar{\mathbf{x}}_n) = \|\bar{\mathbf{x}}_n\| \left[\frac{\bar{\mathbf{x}}_n}{\|\bar{\mathbf{x}}_n\|} \cdot \frac{\mathbf{x}_n}{\|\mathbf{x}_n\|} \right] \tag{6.5}$$

This returns a high value if the gradient magnitude is high and/or the gradient direction and contour normal are in roughly the same direction.

The symmetry of the limbs can be taken advantage of to improve the contour–edge selection though the use of matching symmetric pairs. This was shown by Sminchisescu [86] where model predicted and matched symmetries were used to strengthen a symmetry-sensitive objective function by increasing the chance of finding correct edges.

Symmetric contour point pairs $\mathbf{x}^a$,$\mathbf{x}^b \in \mathcal{C}$ (symmetrically opposite each other on opposing sides of the limb) were also identified in Section 5.1.3. A search is performed along the Bresenham[1] line drawn through points $\mathbf{x}^a$ and $\mathbf{x}^b$ for edge points $\mathcal{E}$. From these edge points a pair $\bar{\mathbf{x}}^a$ and $\bar{\mathbf{x}}^b$ is chosen so that the length difference error, scaled by the gradient scores, is minimised

$$\bar{\mathbf{x}}^a, \bar{\mathbf{x}}^b = \min_{\bar{\mathbf{x}}^a, \bar{\mathbf{x}}^b \in \mathcal{E}} \frac{\left| \left\| \bar{\mathbf{x}}^a - \bar{\mathbf{x}}^b \right\| - \left\| \mathbf{x}^a - \mathbf{x}^b \right\| \right|}{s(\mathbf{x}_n^a, \bar{\mathbf{x}}_n^a) s(\mathbf{x}_n^b, \bar{\mathbf{x}}_n^b)} \tag{6.6}$$

In other words, it identifies two edge points with relatively high gradient scores and roughly the correct expected distance apart.

After finding the two edge points, the error distance is calculated as the squared distance between the pairs' midpoints:

$$e_e(\mathbf{x}^a, \mathbf{x}^b, \bar{\mathbf{x}}^a, \bar{\mathbf{x}}^b) = \frac{1}{2} \left(\frac{\left\| \frac{\mathbf{x}^a + \mathbf{x}^b}{2} - \frac{\bar{\mathbf{x}}^a + \bar{\mathbf{x}}^b}{2} \right\|}{\sigma} \right)^2 \tag{6.7}$$

[1] a straight line between two given points, using only integer addition, subtraction and bit shifting [1]

where σ is the limb diameter $\left\|\mathbf{x}^a - \mathbf{x}^b\right\|$ which makes the objective function scale invariant for different body parts.

For contour points with no symmetric pair, a similar approach is used to find its distance error. Each contour point is assigned the edge point which has the highest gradient score along its normal search line:

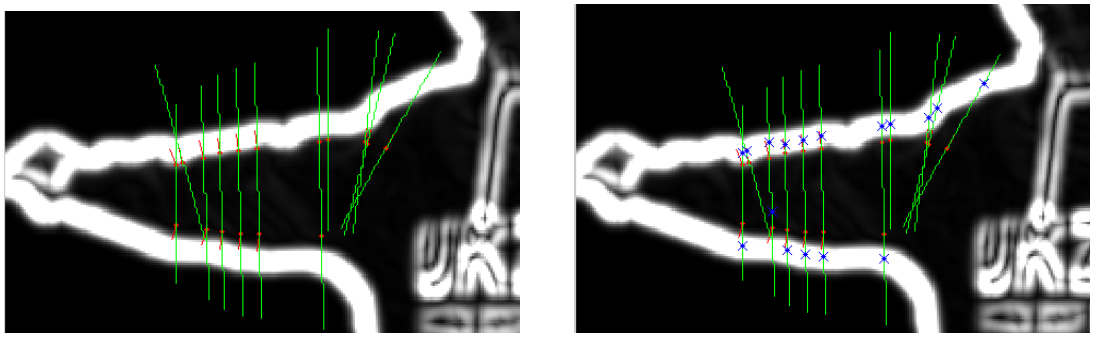
$$\bar{\mathbf{x}} = \max_{\bar{\mathbf{x}} \in \mathcal{E}} s(\mathbf{x}_n, \bar{\mathbf{x}}_n) \tag{6.8}$$

Once the contour point $\mathbf{x}$ and its corresponding edge point $\bar{\mathbf{x}}$ are obtained, the distance error measure is calculated, scaled by the limb diameter σ, as:

$$e_e(\mathbf{x}, \bar{\mathbf{x}}) = \frac{1}{2}\left(\frac{\|\mathbf{x} - \bar{\mathbf{x}}\|}{\sigma}\right)^2 \tag{6.9}$$

6.1.2.1 Edge Alignment Process

The edge matching process is illustrated in Figures 6.3 – 6.6.



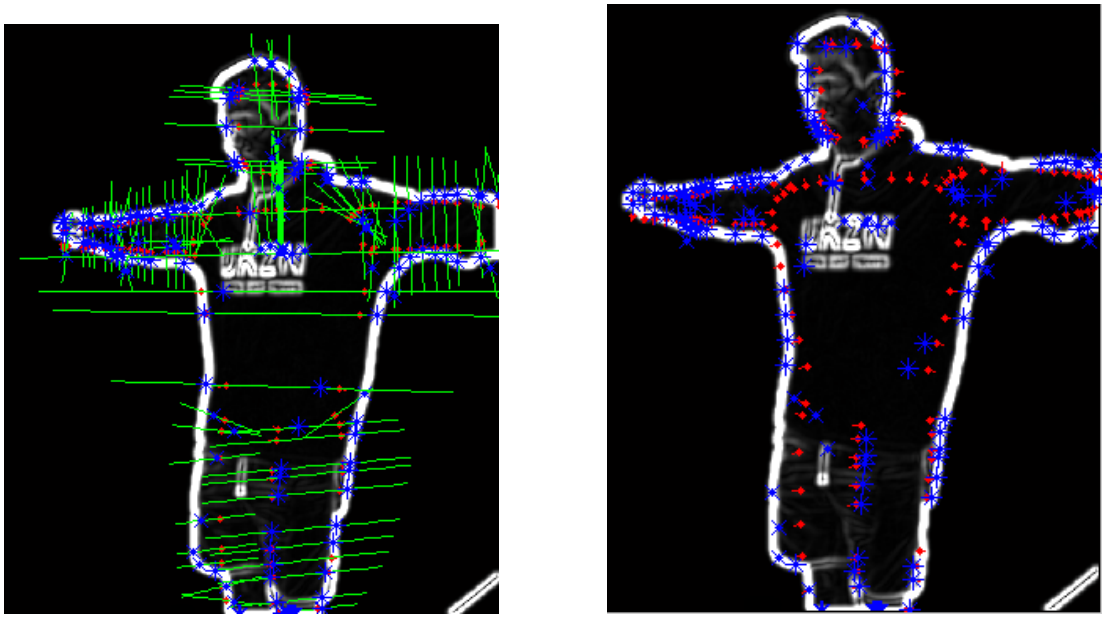
(a) (b)

Figure 6.3: View of the upper right arm during various stages of edge matching from camera view three. (a) with selected contour points and contour normal (red dot and line) and Bresenham line (green line), (b) with matched edge points (blue cross) added.

Figure 6.3 displays various stages during the edge matching. After randomly selecting a set of contour points from the body model (red dot) and projecting them into the selected camera view, Bresenham lines are drawn to determine which edge points are searched for a best match to the contour points. This is performed, for symmetric contour point pairs, with the line drawn through the two points on either side of the arm. For non-symmetric contour points, the line is constructed through an individual point in its normal direction.

Searching all the possible edge points (points along the Bresenham lines) a best edge point match is found and displayed (blue cross). Most of these edge points are on the displayed 'high intensity' edge (white lines) as expected and required. The one point which is not on the edge in Figure 6.3b, may have matched to a crease (which is

too dim to clearly see) and because the arm diameter distance at that point is a better match, that point was selected as the best match instead. However, this is only one point and the majority matched correctly.



(a) (b)

Figure 6.4: Full view from camera 3 with contour points (red dots) and matched edge points (blue asterisk) (a) with Bresenham lines (green lines) and (b) without.

The inclusion of the Bresenham lines when showing the matching makes the picture very cluttered. Therefore, all further results will not display them. However, Figure 6.4a gives one a good visualisation of how they are used for every section of the body. It shows how some parts of the body have a large portion of symmetric pair contour points (such as the arms) while other parts, depending on the camera view, largely consist of individual non-symmetric points (such as the right leg in this view). Figure 6.4b shows the same view but without the Bresenham lines which allows the selected model contour points and matching edge points to be much more clearly seen.

Figure 6.5 demonstrates good edge matching on the legs, even when there is no distinct edge line for the inside of the leg. The outside contour points find edge matches on the actual edge, while the inside contour points (of the left leg) match with edges in a fairly straight line. These inside points do not have any high intensity edge gradient to match with. However, they are able to make use of the symmetric pairs and expected limb diameter to make the matches at a roughly correct distance from their symmetric pair.

Even in cases when the model is very poorly aligned with the observed subject,

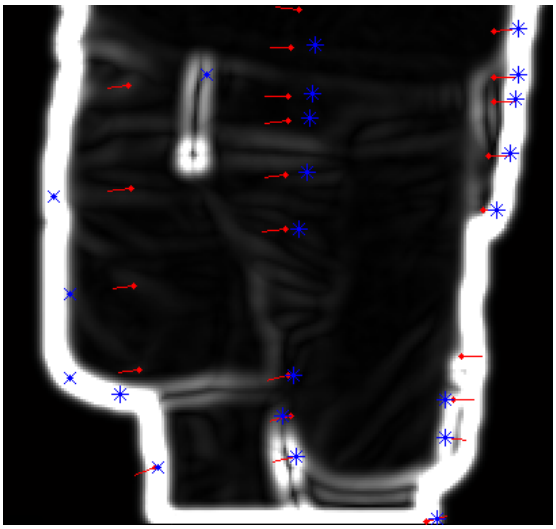


Figure 6.5: View of the edge matching for the legs from camera view 3.

Figure 6.6: Overall full-body view of the edge matching as seen by camera 4.

edge matching can generally still perform well. This is seen in Figure 6.6 where, even with the misaligned model, the correct edges are generally matched with the selected contour points. This is especially evident with the torso and leg edge point matches. When used with an optimisation method, this correct matching will allow the model to be brought into better alignment with the observations.

6.1.3 3-D Model Colour

Colour information is already used to perform segmentation in the illumination-invariant background subtraction method (Section 4.1) as well as in RGB colour-based edge detection (Section 4.2). It is even used for display of the voxel-hull reconstruction by adding colour to the surface voxels. Here, this voxel colour is used indirectly in the optimisation by restricting the voxels used in the nearest-neighbourhood search for a given model point. Only voxels with a similar matching colour to a given body model point are included in the search, with all the other voxels ignored.

The objective of this use of colour is two-fold. Firstly, the speed of the nearest-neighbourhood search (and hence optimisation) is increased as fewer voxels are required to be searched for a matched. Secondly, the surface matching is made more robust against limbs getting stuck to incorrect body parts. This sometimes occurs when a limb comes close to the body or another limb and then remains stuck to that observed part, even after departure of the actual limb from it. This use of colour in the matching will help to differentiate between the body parts, assuming they are of different colours. If there is no colour difference then the matching is not adversely affected by the use of colour.

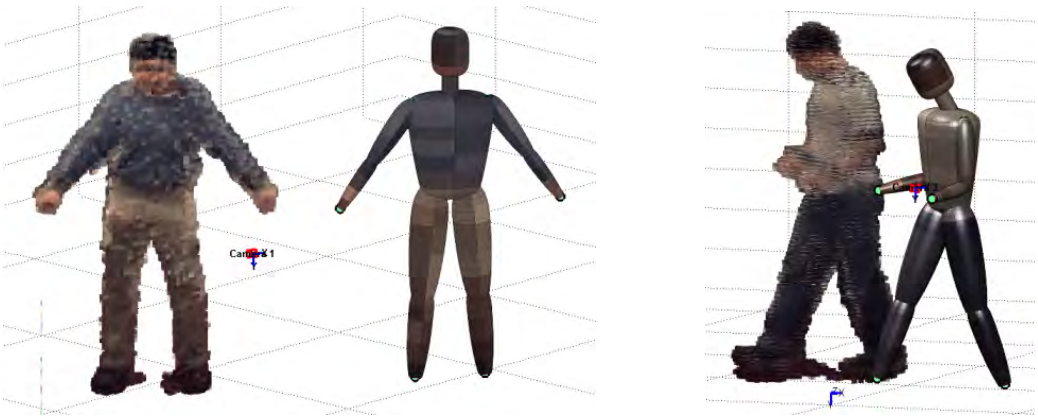
6.1.3.1 Implementation

Before any colour matching can be done, the colour models for the points on the body model need to be defined. Since these colour models are updated at the end of each frame and only a subset of points is used in the matching and optimisation due to stochastic sampling, only the colour models of the subset of points used in the last iteration is updated. This results in only an occasional and somewhat haphazard colour model update. To enable more consistent colour updating the body-model points are grouped into patches with each patch having its own colour model. Each patch is then updated by the subset of points used in the last iteration which fall into the given patch. This enables the colour models of the patches to be consistently updated even when all their individual points may not have been.

Once the body-model has been initialised and is in place (aligned fairly well with the observed data) the colour model for each patch is set. This is performed by assigning to each model point the colour of its closest voxel. All the colours of points belonging

to a specific patch are then mixed together to define the patch's colour model.

Each colour model is represented by a Gaussian distribution in RGB colour space with each channel assumed to be independent and modelled by a single Gaussian to improve speed. The speed is further increased by fixing the variance to avoid expensive computations when updating. Thus each patch model consists of 6 parameters defining it — a mean (μ_i) and standard deviation (σ_i) for each of the 3 colour channels $i \in [1, 3]$.



(a) (b)

Figure 6.7: The body model with coloured patches next to the coloured 3-D voxel-hull reconstruction for the HumanEvaII (a) S2 and (b) S4 datasets.

Figure 6.7 shows the results of the colour patch setup on the body models for the HumanEvaII S2 and S4 datasets. It is easy, especially in 6.7a, to see the slightly different colour patches made up of a number of body model points. It can also be seen that the patch colours match their corresponding voxel colours, as expected.

6.1.3.2 Colour segmentation

3-D colour segmentation is performed before optimisation starts for each frame. Each surface voxel is tested if it matches any of the patches' colour models.

Let $p_i(n, c_i) \backsim \mathcal{N}(\mu_i, \sigma_i)$, $i \in [1, 3]$ be the probability density function of the ith colour channel of patch n. The likelihood that some voxel with colour (c_1, c_2, c_3) matches patch n, using Bayes theorem with independent colour channels, is

$$P(n, c_1, c_2, c_3|\text{patch}) = p_1(n, c_1)p_2(n, c_2)p_3(n, c_3) \tag{6.10}$$

Following this, a decision of whether the voxel matches the patch or not is made based on the decision threshold value, T_d

$$P(n, c_1, c_2, c_3|\text{patch}) \overset{\text{match}}{>} T_d \tag{6.11}$$

If the test is true then the voxel is found to match the patch.

Now, during the nearest-neighbour search for the closest voxel in the surface optimisation, only the voxels which are found to match the colour model of the point's patch are used, with the others being discarded. This is illustrated in Figures 6.8 and 6.9 where only the voxels which match the colour of the selected point's patch are displayed. The decision threshold value, T_d, was set, through experimentation, as 5×10^{-6}. This value was chosen so as to ensure some voxels are found to match a given patch colour model but not so many that incorrect colour voxels are matched.

Figure 6.8 displays the matching voxels for a test performed on the HumanEvaII S2 dataset. As an example, patch 9 which is known to be on the upper left leg is selected to test the voxel matching accuracy. The voxels whos colour were found to match to this patch's colour are displayed in the top row of Figure 6.8. These images show the matching voxels in various frames, from frame 809 to frame 1049. Similarly, the voxels whos colour were found to match that of the select patch 89 (known to be on the upper left leg) are displayed in the middle row, while those found to match the colour of selected patch 190 (known to be on the head) are presented in the bottom row. From the figure it can be seen that the matching voxels are fairly consistent across the displayed frames for the different patches. Obviously the matching voxels are not only on, for example, the head but also other parts of the body which have a similar colour. However, voxels on the head *are* found to match. Thus, using the nearest neighbourhood search, one of these voxels is selected as the closest matching one to the patch on the head.

Similar results were found during the testing of the HumanEvaII S4 dataset and are shown in Figure 6.9. Here the top row displays the matching voxels for selected patch 11 (on the upper left leg) and the bottom row shows those matching selected patch 111 (on the lower right arm).

6.1.3.3 Colour model update

At the end of the frame's optimisation, the mean, μ_i of the colour model for each patch is updated as

$$\mu_i^{new} = (1 - \alpha)\mu_i + \alpha c_i \tag{6.12}$$

where c_i is the new colour value for channel i and where $\alpha \in [0, 1]$ is the learning rate which controls the adaptation speed of the patch colour.

A selected value of $\alpha = 0.3$ is used which allows updating but not too quickly. The results in figure 6.8 and 6.9 indicate that this value is fine since the matching voxels, over the progression of frames, remains relatively even — as is expected.

It should also be noted that since the model's surface colour will not change during optimisation and will only vary slightly from frame to frame, it is sufficient to only

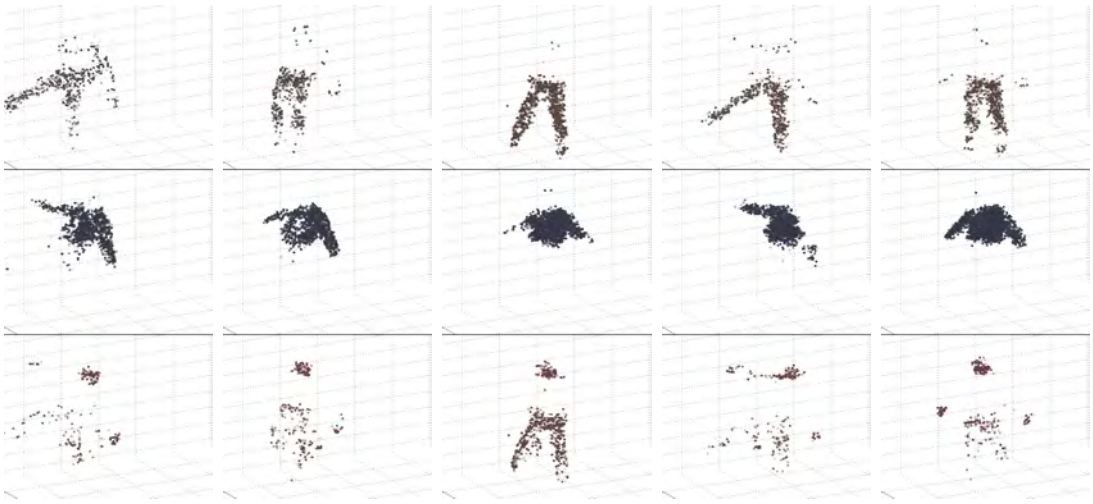


Figure 6.8: Matching voxels of HumanEvaII S2 dataset for patch 9 (top), patch 89 (middle) and patch 190 (bottom) in indicated frames

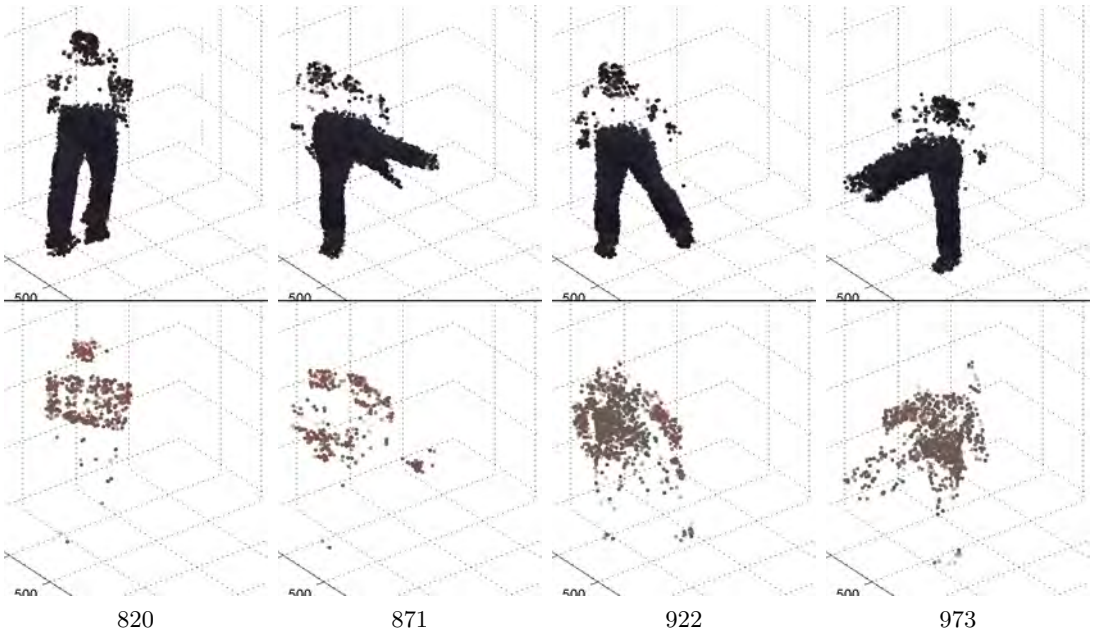


Figure 6.9: Matching voxels of HumanEvaII S4 dataset for patch 11 (top), and patch 111 (bottom) in indicated frames

perform colour model updating at the end of each frame optimisation.

A summary of the final algorithm with 3-D colour segmentation and update added (in bold) is shown in Figure 6.10.

1. Initialise body model
2. **Set patch colour models**
3. **Perform 3-D colour segmentation** (6.10 & 6.11)
4. Iterate optimisation until convergence
   - Compute gradient
   - Update step sizes
   - Update model parameters
5. **Update patch colour models** (6.12)
6. Repeat from point 3 for next frame

Figure 6.10: Final algorithm summary with 3-D colour segmentation parts added

6.2 Stochastic Meta Descent

In order to adjust the predicted model to match the observed data, the objective function must be optimised. Stochastic meta descent (SMD) is one approach examined to perform this as it provides 'gradient descent with local step size adaptation that combines rapid convergence with excellent scalability' [50]. Various other techniques were discussed in Section 2.1.3 but, as shown by Kehl and Van Gool [50], SMD optimisation performs well for full body optimisation.

Stochastic meta descent is built on three concepts [50]:

1. *Stochastic sampling*: This is a strong feature of SMD and offers better convergence than other optimisation methods. This is because instead of using all the available points to evaluate the objective function it only uses
   (a) a small subset of points — resulting in less computation and hence faster convergence
   (b) a randomly shuffled subset of points at each iteration — giving less chance of being trapped in a local minimum

   These points are randomly sampled from, in the case of this research, the 10 individual body parts (head, torso, left & right upper arm, left & right lower

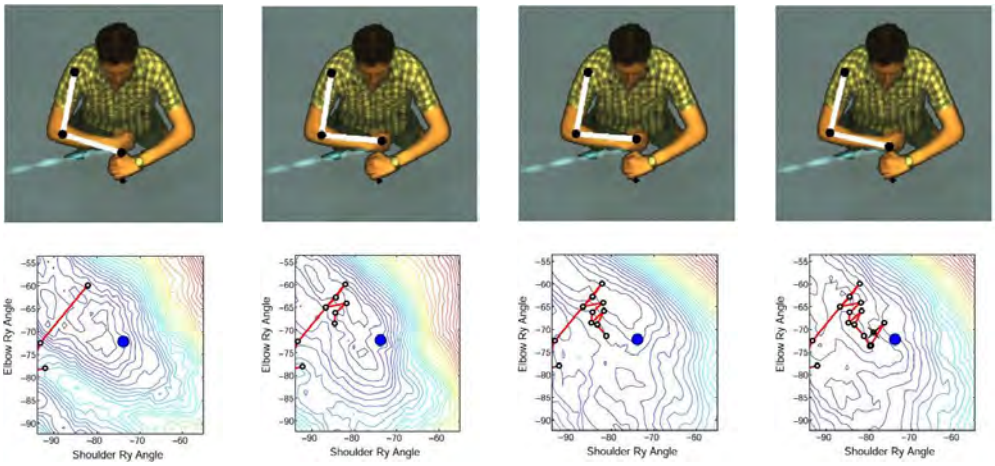


Figure 6.11: Stochastic sampling overcomes the local minima by using a randomly shuffled subset of points at each iteration. The graphs in the bottom row show the shifting local minimum with the resulting solution after 15 iterations (at black cross) being close to the optimum (blue dot).[50]

arm, left & right upper leg, left & right lower leg) with 40% of the points sampled used in the SMD optimisation.

2. *Levels of optimisation*: SMD performs two levels of optimisation. First it optimises, for all parameters, the individual step sizes $\mathbf{a}$ (using a meta step size μ) and then the state parameters $\mathbf{p}$, at each iteration, in a gradient descent method.

3. *Past history*: SMD takes into account past history when updating the parameters. This allows long-term effects to be captured which dampens erratic fluctuations in the results and increases the efficiency.

6.2.1 Update Theory

The following explains how various intermediate variables and ultimately the parameter vector $\mathbf{p}$ is updated using SMD optimisation.

Given $\mathbf{g_i}$, the gradient of $f(\mathbf{p}_i)$ at iteration step i, the parameter vector $\mathbf{p}_i$ is updated via

$$\mathbf{p}_{i+1} = \mathbf{p}_i - \mathbf{a}_i \odot \mathbf{g}_i \tag{6.13}$$

where $\odot$ represents the Hadamard[2] product and $\mathbf{a}$ is the local step size.

The local step size vector is updated via

$$\mathbf{a}_i = \mathbf{a}_{i-1} \odot \max\left(\frac{1}{2}\ ,\ 1 + \mu \odot \mathbf{v}_i \odot \mathbf{g}_i\right) \tag{6.14}$$

[2]also known as pointwise or component-wise product

where μ is the meta step size vector (a vector of fixed values which controls the rate of update of $\mathbf{a}$) and $\mathbf{v}$ is average of all past step sizes.

Vector $\mathbf{v}$ is updated via

$$\mathbf{v}_{i+1} = \lambda \mathbf{v}_i + \mathbf{a}_i \odot (\mathbf{g}_i - \lambda H_i \mathbf{v}_i) \tag{6.15}$$

where H is the Hessian matrix and λ determines the length of time over which the past step sizes are taken into account ($0 \leq \lambda \leq 1$).

As with all optimisation techniques, a condition to stop the algorithm at convergence is needed. With SMD this can be a little tricky because gradient and objective function values tend to oscillate around their optima. This is caused by the stochastic sampling of points where a different point set is used for each evaluation. Thus, the normal methods of using the gradient value or objective function result is not possible. Instead, to handle such oscillations, a tolerance on the average change of parameter values $\mathbf{p}_i$, including the new $\mathbf{p}_{i+1}$ values, over the last n steps is used. An upper limit on the number of iterations is also given in case there is no (or insufficient) convergence.

$$\left\| \frac{1}{n} \sum_{k=i-n+1}^{i} \mathbf{p}_k - \frac{1}{n} \sum_{k=i-n+2}^{i+1} \mathbf{p}_k \right\| = \frac{1}{n} \left\| \mathbf{p}_{i-n+1} - \mathbf{p}_{i+1} \right\| \leq \epsilon \tag{6.16}$$

and

$$i \leq i_{max} \tag{6.17}$$

i_{max} is set to around $20-40$, n to about 3 and the tolerance ϵ depends on the parameter values.

6.2.2 Gradient and Hessian Calculation

While most of the variable update calculations are straight forward, there are two which are not as obvious. These are the gradient ($\mathbf{g}_i$) and Hessian ($\mathbf{H}_i$) calculations. The calculation of these two are linked since the gradient is the first partial derivative, $\mathbf{g}_i = \frac{\partial f(\mathbf{p}_i)}{\partial \mathbf{p}_i}$, while the Hessian is the second partial derivative, $\mathbf{H}_i = \frac{\partial^2}{(\partial \mathbf{p}_i)^2} f(\mathbf{p}_i)$.

A lot of work went into the computation of these for update and optimisation of the model parameters. The ideal solution is to explicitly differentiate the objective function $f(\mathbf{p}_i)$ with respect to the parameter vector $\mathbf{p}_i$ for some iteration run i (for the sake of simplicity, the i will be dropped for the rest of this section as all the calculations occur during the same iteration). Since the objective function does not directly contain the parameter variables $\mathbf{p}$ but instead uses the distance variables $\mathbf{x}$ in its measure, the chain rule needs to be used such that:

$$\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}} = \frac{\partial f(\mathbf{p})}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial \mathbf{p}} \tag{6.18}$$

The first partial derivative is able to be calculated but the second is where the problem lies. Calculations from Kehl [49] aided in determining that this second derivative can be computed from the model and camera transformation matrices and equations. Before this is looked at, the calculation of $\frac{\partial f(\mathbf{p})}{\partial \mathbf{x}}$ is presented.

For this calculation it is assumed that the matrix $\mathbf{J}_x = \frac{\partial \mathbf{x}}{\partial \mathbf{p}}$ has been found and contains the derivative with respect to the model parameters for the 2-D contour points or the 3-D surface points.

The gradient (or Jacobian) of each individual error term is computed, using the chain rule, from Equation (6.3).

$$\begin{aligned}
\mathbf{J}_{\delta(\mathbf{x},\bar{\mathbf{x}})} &= \frac{\partial}{\partial \mathbf{p}} \delta(\mathbf{x}, \bar{\mathbf{x}}) \\
&= \frac{\partial}{\partial \mathbf{p}} \rho(e(\mathbf{x}, \bar{\mathbf{x}})) \\
&= \frac{\partial \rho(e)}{\partial e} \frac{\partial e(\mathbf{x}, \bar{\mathbf{x}})}{\partial \mathbf{p}} \\
&= \frac{1}{v} e^{\frac{-e(\mathbf{x},\bar{\mathbf{x}})}{v}} \frac{\partial e(\mathbf{x}, \bar{\mathbf{x}})}{\partial \mathbf{p}}
\end{aligned} \tag{6.19}$$

Now, the gradient of the squared distances varies depending on which type of data values are being looked at and used. For a model surface point $\mathbf{x}$ and its corresponding voxel $\bar{\mathbf{x}}$ or for a single contour point $\mathbf{x}$ and its corresponding edge pixel $\bar{\mathbf{x}}$, Equations (6.4) and (6.9) lead to

$$\begin{aligned}
\frac{\partial e_{s/e}(\mathbf{x}, \bar{\mathbf{x}})}{\partial \mathbf{p}} &= \frac{\partial}{\partial \mathbf{p}} \frac{1}{2} \left(\frac{\|\mathbf{x} - \bar{\mathbf{x}}\|}{\sigma} \right)^2 \\
&= \frac{\partial}{\partial \mathbf{x}} \frac{1}{2} \left(\frac{\|\mathbf{x} - \bar{\mathbf{x}}\|}{\sigma} \right)^2 \frac{\partial \mathbf{x}}{\partial \mathbf{p}} \\
&= \frac{1}{\sigma^2} (\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{J}_x
\end{aligned} \tag{6.20}$$

When symmetric contour point pairs are used instead, the gradient of the squared distances using Equation (6.7) leads to

$$\begin{aligned}\frac{\partial e_e(\mathbf{x}^a,\mathbf{x}^b,\bar{\mathbf{x}}^a,\bar{\mathbf{x}}^b)}{\partial \mathbf{p}} &= \frac{\partial}{\partial \mathbf{p}}\frac{1}{2}\left(\frac{\left\|\frac{\mathbf{x}^a+\mathbf{x}^b}{2}-\frac{\bar{\mathbf{x}}^a+\bar{\mathbf{x}}^b}{2}\right\|}{\sigma}\right)^2 \\ &= \frac{\partial}{\partial \mathbf{x}}\frac{1}{2}\left(\frac{\left\|\frac{\mathbf{x}^a+\mathbf{x}^b}{2}-\frac{\bar{\mathbf{x}}^a+\bar{\mathbf{x}}^b}{2}\right\|}{\sigma}\right)^2\frac{\partial \mathbf{x}}{\partial \mathbf{p}} \\ &= \frac{1}{\sigma^2}\left(\frac{\mathbf{x}^a+\mathbf{x}^b}{2}-\frac{\bar{\mathbf{x}}^a+\bar{\mathbf{x}}^b}{2}\right)^T\left(\frac{1}{2}\mathbf{J}_{x^a}+\frac{1}{2}\mathbf{J}_{x^b}\right) \end{aligned} \tag{6.21}$$

The above differentiations are possible because it is assumed that for a given model point $\mathbf{x}$ the pose changes by a sufficiently small amount that the same observed feature $\bar{\mathbf{x}}$ is still found — thus making it a constant in the equations and allowing local modelling of the objective function.

Therefore, the gradient of the objective function is obtained by computing

$$\mathbf{J} = \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}} = w_s \sum_{N_s} \mathbf{J}_{\delta_s} + w_e \sum_{N_e} \mathbf{J}_{\delta_e} \tag{6.22}$$

Kehl [49] approximates the Hessian matrix (matrix of second order derivatives) by the Fisher information matrix $\mathbf{F} = \mathbf{J}^T\mathbf{J}$. Therefore the Hessian is computed as

$$\mathbf{H} \approx w_s \sum_{N_s} \mathbf{J}_{\delta_s}^T \mathbf{J}_{\delta_s} + w_e \sum_{N_e} \mathbf{J}_{\delta_e}^T \mathbf{J}_{\delta_e} \tag{6.23}$$

6.2.2.1 Calculating $\mathbf{J}_x$

Now all that remains is to find $\mathbf{J}_x$ which can then be substituted in to (6.20) and (6.21). $\mathbf{J}_x$ is the 'link' between the parameter vector $\mathbf{p}$ and all the $\mathbf{x}$ points on the model surface or edge. As mentioned earlier, this can be computed from the model and camera transformation matrices and equations which were examined in Sections 3.2 & 5.1.2.

For the surface points, their values are obtained by transforming (using model rotation and translation) the original superellipsoids centred at the origin, to their new locations which make up the superellipsoid body model in the desired 3-D world co-ordinate location and pose.

The contour points' transformation starts out in the same manner but becomes more complicated as the camera's extrinsic and intrinsic parameters also have to be taken into account. The lens distortion (radial and tangential) part of the extrinsic parameters is especially problematic as it is not linear like all the other transformations and makes the equations a lot more complex.

Using the model and camera transforms, the relationships between the $\mathbf{x}$ points (surface or edge) and the model parameters are found.

A point $\mathbf{X}$ in the superellipsoid's local co-ordinate frame (centred at the origin) can be expressed, through the transformations of the articulated body model's chain, in world co-ordinates as $\mathbf{x}_s$ where

$$\mathbf{x}_s = \mathbf{M}_{loc1}\mathbf{M}_{loc2}\ldots\mathbf{M}_{loci}\mathbf{X} \tag{6.24}$$

Similarly, the same point $\mathbf{X}$ in the superellipsoid's local co-ordinate frame can be expressed, through the transformations of the articulated body model's chain and the intrinsic and extrinsic camera parameters, as $\mathbf{x}_e$ in image co-ordinates

$$\begin{aligned}\mathbf{x}_{en} &= \mathbf{K} \times \texttt{lensDistortionFunc}(\mathbf{M}_c\ \mathbf{x}_s) \\ &= \mathbf{K} \times \texttt{lensDistortionFunc}(\mathbf{M}_c\ \mathbf{M}_{loc1}\mathbf{M}_{loc2}\ldots\mathbf{M}_{loci}\ \mathbf{X}) \qquad (6.25)\end{aligned}$$

$$\mathbf{x}_e = \begin{bmatrix} \frac{x_{en}}{z_{en}} \\ \frac{y_{en}}{z_{en}} \end{bmatrix} \tag{6.26}$$

where `lensDistortionFunc` is a function which performs the tangential and radial lens distortion transform of (3.15).

The lens distortion function is not linear like the other camera transformations and as a result causes difficulties when differentiating to get the gradient. While this lens distortion function makes enough of a difference to include it when transforming 3-D camera reference co-ordinates to 2-D pixel points, its influence on the gradient is not considered to be significantly large and as such can be ignored when calculating the gradient. Therefore, for gradient calculations, (6.25) and (6.26) are simplified to

$$\begin{aligned}\mathbf{x}_{en} &= \mathbf{K}\ \mathbf{M}_c\ \mathbf{x}_s = \boldsymbol{P}\ \mathbf{x}_s \\ &= \boldsymbol{P}\ \mathbf{M}_{loc1}\mathbf{M}_{loc2}\ldots\mathbf{M}_{loci}\ \mathbf{X} \qquad (6.27)\end{aligned}$$

$$\mathbf{x}_e = \begin{bmatrix} \frac{x_{en}}{z_{en}} \\ \frac{y_{en}}{z_{en}} \end{bmatrix} \tag{6.28}$$

From Equations (6.24), (6.27) and (6.28) the gradient $\mathbf{J}_x$ may be calculated

$$\mathbf{J}_{x(surface)} = \frac{\partial \mathbf{x}_s}{\partial \mathbf{p}} = \frac{\partial(\mathbf{M}_{loc1}\mathbf{M}_{loc2}\ldots\mathbf{M}_{loci}\mathbf{X})}{\partial \mathbf{p}} \tag{6.29}$$

and

$$\mathbf{J}_{x(edge)} = \frac{\partial \mathbf{x}_e}{\partial \mathbf{p}} = \frac{\partial \mathbf{x}_e}{\partial \mathbf{x}_{en}} \cdot \frac{\partial \mathbf{x}_{en}}{\partial \mathbf{x}_s} \cdot \frac{\partial \mathbf{x}_s}{\partial \mathbf{p}}$$
$$= \begin{bmatrix} \frac{1}{z_{en}} & 0 & \frac{-x_{en}}{z_{en}^2} \\ 0 & \frac{1}{z_{en}} & \frac{-y_{en}}{z_{en}^2} \end{bmatrix} \cdot \boldsymbol{P} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \frac{\partial(\mathbf{M}_{loc1}\mathbf{M}_{loc2}\ldots\mathbf{M}_{loci}\mathbf{X})}{\partial \mathbf{p}} \quad (6.30)$$

where all the parameters of $\mathbf{p}$ are in the $\mathbf{M}_{loc}$ matrices as the rotation angles or translation distances. This differentiation may be found by applying a series of chain and product derivative rules to the last part of the above equations, with only the dependent angles in the $\mathbf{M}_{loc}$ matrices playing a role for a given parameter p of the 24-parameter vector $\mathbf{p}$.

6.2.3 Constraints

As mentioned in Section 5.2, the implementation of joint constraints helps to improve pose predictions and matches.

SMD provides a very simple and elegant method to implement these constraints. The limits are enforced by a function that maps all update parameter values back to a feasible region at the end of each iteration:

$$\mathbf{p}_{i+1}^c = f_{\text{constrain}}(\mathbf{p}_{i+1}) \quad (6.31)$$

The step sizes $\mathbf{a}$ and $\mathbf{v}$ are updated, together with the parameter vector $\mathbf{p}$, via the gradient. If $\mathbf{p}$ is constrained, the gradient must also be constrained which then effects the update of $\mathbf{a}$ and $\mathbf{v}$. In order to update these values correctly, the constrained gradient, $\mathbf{g}^c$, is calculated before their update via

$$\mathbf{p}_{i+1}^c = \mathbf{p}_i^c - \mathbf{a}_i \odot \mathbf{g}_i \quad \Rightarrow \quad \mathbf{g}_i^c = \frac{\mathbf{p}_i^c - \mathbf{p}_{i+1}^c}{\mathbf{a}_i} \quad (6.32)$$

Thus the step sizes can be updated accurately using this value in (6.14) and (6.15).

A nice summary of the implementation order of this constrained SMD optimisation is provided by Bray *et al.* [12] and shown in Figure 6.12.

6.3 SMD with a Hierarchical Approach

Optimisation was initially performed by attempting to optimise all 24 body model parameters (joint angles and torso position displacements) simultaneously. However, because the human body is a collection of kinematic chains, various body part positions are affected by the position of parent parts. Therefore, trying to optimise a whole kinematic chain simultaneously can lead to poor convergence as well as oscillation

- LOOP $t = 1$ TO last frame
  - $\mathbf{v}_0 = 0$; $\mathbf{a}_0 = \mathbf{a}*$; $\mathbf{p}_0 = IntialState$
  - LOOP $i = 1$ TO convergence
    1. pick sample points S_i;
    2. calculate $\mathbf{g}_i$ (6.22) and $\mathbf{a}_i$ (6.14);
       — IF $i = 1$ THEN $\mathbf{a}* = \mathbf{a}_1$;
    3. calculate $\mathbf{p}_{i+1}$ (6.13);
    4. calculate $\mathbf{p}^c_{i+1}$ (6.31);
    5. calculate $\mathbf{g}^c_i$ (6.32), ; $\mathbf{H}_i$ (6.23), $\mathbf{v}_{i+1}$ (6.15).

Figure 6.12: Algorithm of constrained SMD optimisation.

around the optimal point.

It, therefore, makes sense to use a hierarchical approach to optimise the various parts of the kinematic chains independently — where the root is optimised first and then optimisation moves outward to the upper and then lower limbs. This allows the dimensionality of the problem to be reduced somewhat, as instead of dealing with 24 parameters only a few parameters are optimised and dealt with each step. Parallel processing is also possible since each limb chain is independent and can be processed on a separate machine if desired — as performed by Kehl and Van Gool [50].

A number of similar hierarchical schemes have been used previously. John *et al.* [45] use a fairly long 12-step hierarchical optimisation scheme. Initially the position and then orientation of the whole body is estimated by considering the entire body as a rigid structure. It then branches to the 5 separate chains — head and neck, left arm, right arm, left leg and right leg. In each of these, the upper limb is optimised first using the lower as a guide (but keeping it rigid with respect to the upper part) and then the lower limb is optimised. No parallel processing is used so each optimisation step is performed sequentially leading to the 12 steps. These 12 steps can make the optimisation process very long.

Sundaresan and Chellappa [91] also use a hierarchical optimisation scheme but implement it slightly differently. First the root of the kinematic chain (the torso) is estimated. In the second step, the first segment of all the limbs is included — so their parameters together with the torso's are estimated. In the final step, all body part pose parameters, except the torso's, are estimated. This is illustrated in Figure 6.13. The scheme is not too long and implements a kind of parallel processing by optimisation all limb roots at one time, just not on separate machines. However, in this approach the lower limbs are never optimised by themselves and optimising the torso by itself can be problematic as it is difficult to get good torso matches due the limbs disrupting the

shape of the torso and its outlines and edges.

Figure 6.13: Hierarchical optimisation scheme implemented by Sundaresan [90]. The optimisation begins at the base of the kinematic chain (torso) and moves outwards along the chains with each step. The segments for which the pose is computed at a given step are coloured in dark grey, while the remaining segments are in light grey.

Kehl and Van Gool [50] updated their previous hierarchical tracking scheme in [52] because 'the articulated chains are not truly independent any more [due to occlusions of contours] and must be optimized simultaneously'. To fix this, they used a 2-stage scheme where, in the first stage, all 24 pose parameters are optimised simultaneously using a large tolerance ϵ_1. The second stage refines this initial pose estimate using a smaller tolerance ϵ_2 while keeping the torso fixed. Each of the articulated chains (head, arms and legs) are thus refined independently and in parallel on multiple machines.

6.3.1 Scheme Employed

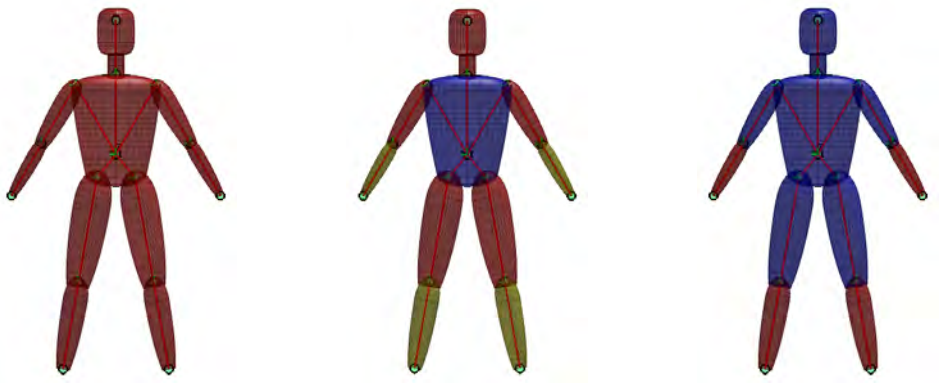
At first an approach like Sundaresan [90] was used. This is a single tolerance scheme and optimises various parts of the kinematic chain in sequence. However, the potential torso and lower limb problems mentioned above were encountered. The torso optimisation was very poor in the first stage which then affected the subsequent stages. To improve this, the first stage was eliminated, as the torso is also optimised in the second stage using the upper limbs to help, and another stage was added on at the end which just optimises the lower limbs. This helped to some extent but a fair amount of oscillation was experienced — especially noticeable with the model legs when they should have remained relatively stationary according to the observations.

Improving on the above method, a hierarchical approached very similar to that used by Kehl and Van Gool [50] is instead used while taking some ideas from Sundaresan [90] and John *et al.* [45]. What John *et al.* [45] do well is use child parts as a guide to optimise the parent pose parameters in a kinematic chain; while Sundaresan [90]'s use of parallel optimisation on a single machine is useful. The two-tolerance scheme used by Kehl and Van Gool [50] is highly beneficial in that it allows the torso to be optimised fairly well together with the simultaneous optimisation of the 'not truly in-

dependent' articulated chains. However, the kinematic chains are then separated which allows faster parallel refinement in the second tolerance stage with increased accuracy due to the lower tolerance.

Thus, these benefits are combined into a single refinement scheme with a 2-stage, 2-tolerance optimisation and 2 steps in the second stage. The first stage optimises all pose parameters simultaneously using a large tolerance ϵ_1. The second stage refines this using a smaller tolerance ϵ_2 and keeps the torso fixed. In the first step, the upper limbs and head are refined while using the lower (child) limbs to guide this optimisation but keeping them rigid with respect to the upper (parent) limbs. Since these upper limb parts and head are independent they are refined together, in parallel, on a single machine. In the second step of stage 2, the lower limbs are refined, again together in parallel. This scheme is illustrated in Figure 6.14. With these improvements there is the trade-off of longer processing time. This is inevitable as the optimisation is being rerun and refined with each step. There is an approximate increase of 2.5 – 3 times the processing time compared to the original single stage optimisation. If multiple machines were used for the parallel processing part of the optimisation it might help offset this extra processing time a little, though not completely.

Results of tracking performed using a hierarchical approach with SMD optimisation are presented in Chapter 7.



Stage 1 Stage 2 : step 1 Stage 2 : step 2

Figure 6.14: Illustration of the hierarchical optimisation approach implemented. Initial simultaneous optimisation of all parameters using a large tolerance; followed by parallel optimisation refinement of the kinematic chains — moving outward while keeping the parent parts fixed in each step. The body segments for which the pose is optimised are coloured in red, the parts which guide the optimisation while being kept rigid themselves are coloured yellow and the fixed body parts are shown in blue.

6.4 SMD with Pose Prediction

One of the actions performed in the HumanEva II dataset is that of the subject walking around in a circle. Here problems were observed in the tracking of the legs and arms when they cross over the body and each other — an arm may become attached to the body and/or a leg to the other leg. On examining when this error occurs, it is noticed that a model leg stops halfway through its motion when crossing over the other leg instead of continuing with its forward motion. This is especially the case when the legs are close together — which inevitably means the 3-D voxel data is closer together and therefore more difficult to differentiate between the two legs.

However, if some prediction scheme is used which tries to estimate the current pose position based on past movements, this can help the forward motion of the leg to continue and thus not get stuck to the other leg.

The use of some temporal model for pose optimisation can also assist the optimisation by moving the body model closer to the optimal pose, from its previous position, based on the parameter history. This is especially beneficial with fast motion sequences where parameter values can change by large amounts in each step. In all cases, both fast and slow, it reduces the number of iterations required for convergence which means faster optimisation.

In [91] the use of both motion and shape cues for tracking is mentioned. They state that only using motion-based cues for tracking can lead to drift problems as the errors of estimating the pose from one frame to another accumulate over time. Similarly, using only shape-based cues can also lead to tracking problems. This is because the system relies solely on absolute cues which are not always able to be reliably extracted from the images in every frame. Thus, the use of poor or incorrect shape information can lead to incorrect optimisation and minima convergence. By using both cues, it is possible to help improve correct convergence without suffering from these individual problems.

A number of papers have made use of motion cues [6, 20, 88, 91]. However, these often use complicated methods which regularly use data obtained from the collected images. Methods such as optical flow, pixel displacement, motion estimation from silhouettes and motion residues are a few approaches implemented. None of these methods make use of body model data. Since a body model is being fitted and optimised with the observed data, model information such as torso displacement and joint angles is known from frame-to-frame. It is possible to take advantage of this model information and past pose parameters to estimate or predict future pose parameters. This is based on the motion of body parts which leads to changing joint angles and pose parameters.

Kehl *et al.* [52] actually implement pose prediction, of a very simple kind, in their tracking. A single line in [52] mentions that a 'first order motion model is used during

the two step optimization'.

Pose prediction is the first step in the optimisation stage. First the pose is predicted, based on past parameters, and then the optimisation scheme is run to further optimise and refine the body model pose. There are a number of different methods that can be used to predict the pose based on past pose parameters, ranging from the very simple to more complicated. The ones examined are:

1. Simple linear prediction
2. Spline extrapolation
3. Autoregression

The linear and spline prediction methods are examined in Sections 6.4.1 & 6.4.2 while the autoregression is presented in Section 6.5.3.2 due to its use in the Smart Particle Filter.

6.4.1 Linear Prediction

Simple linear prediction estimates the new pose based on the movement or displacement between the last two frames and can be calculated as

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \eta(\mathbf{p}_t - \mathbf{p}_{t-1}) \tag{6.33}$$

where η is the factor that controls the effect of past change on the current estimate. This is usually set as a value in the range of [0,1], with 0.5 giving and averaging effect.

Some positive results were observed with this simple prediction. Testing was performed using the HumanEvaII S2 and S4 datasets with particular emphasis placed on the legs as they had been experiencing tracking difficulty when crossing over in the walking sequence.

6.4.2 Spline Extrapolation

For the spline extrapolation, built in Matlab functions are used.

The first function fits a curve to the selected points using

```
func = spaps(x,y,tol)
```

which 'returns the B-form of the smoothest function, `func`, that lies within the given tolerance, `tol`, of the given data points' with `x` and `y` the coordinates of the data points.

While the second extrapolates the curve using

```
g = fnxtr(f,order)
```

which 'returns the spline (in ppform) that agrees with the spline in `f` on the latter's basic interval but is a polynomial of the given order outside it'.

The number of points to which the curve is fitted can be varied. If too many are selected, the current movement is not accurately predicted as too much past information is taken into account. On the other hand, if too few points are used an accurate trend of the current movement is difficult to obtain as local pose errors can detrimentally affect the pose prediction. The use of 6 and 10 past points were tested, both without much success. The prediction seems to be incorrect and jumps too much, causing the optimisation to diverge after only a few frames. This is largely due to its cubic nature which attempts to fit too many curves to the data, which is generally more linear or parabolic.

Due to the poor performance of the spline extrapolation, the SMD approach implemented with pose prediction uses the simple linear prediction. Tracking results using this approach are included in Chapter 7.

6.5 Smart Particle Filter

6.5.1 Particle Filtering

Many real world problems, including human pose tracking, require an estimate of the state of the system, based on given observations, for each time-step or frame. This is a classic filtering problem and there exist many methods to solve these sort of problems. Particle filters have become popular for solving human motion tracking problems. The reason for this is that they are easy to implement and do not assume that signal or observation probabilities follow a linear or Gaussian model — like many other filters including the popular Kalman filter [48].

Particle filters [42] are model estimation techniques and offer a probabilistic framework for dynamic state estimation. Gall *et al.* [27] describe them as 'recursive Bayesian filters'. They can be used to compute the posterior density, $p(\mathbf{s}_t|\mathbf{z}_{1:t})$, of the current object state, $\mathbf{s}_t$, based on all observations, $\mathbf{z}_{1:t}$, up to time t. The process density, $p(\mathbf{s}_t|\mathbf{s}_{t-1})$, and the observation density, $p(\mathbf{z}_t|\mathbf{s}_t)$, are not required to be Gaussian, as mentioned, making them especially attractive.

The tracking framework is made robust in cases of clutter and occlusion by modelling and incorporating uncertainty in it. Particle filtering is an approach based on sampling, where the posterior density function is approximated by a weighted particle set $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}_{n=1}^{N}$. Each particle, $\mathbf{s}_t^{(n)}$, represents a hypothetical state of the model with its corresponding discrete sampling probability, $\pi_t^{(n)}$. This means that multiple hypotheses can be tested at once. [13]

The generic particle filter consists of three basic steps, resampling, propagation and updating. *Resampling* involves selecting N new particles, not necessarily unique, from the current N using some sampling scheme, usually based on the particles' weights. *Propagation* pertains to randomly spreading or diffusing the selected particles so as to better represent the search space. This step may also include some 'next state' prediction method, prior to diffusion, to improve the particles' movement towards the optimum. The last step *updates* the particles' weights according to a likelihood function determined by the observations. This scheme is illustrated in Fig. 6.15.

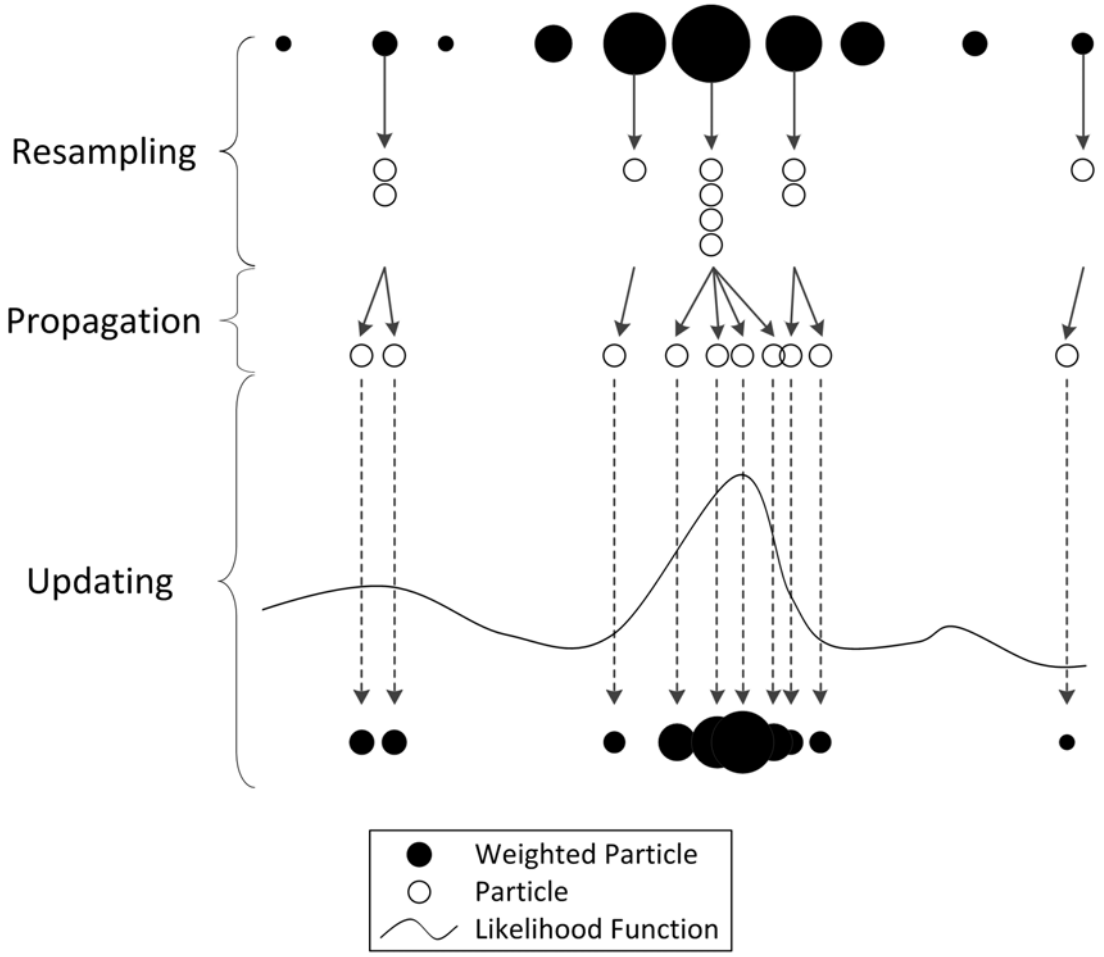


Figure 6.15: Diagram illustrating the operation of the generic particle filter.

6.5.2 Smart Particle Filtering

Tracking an articulated structure such as a full human body, without excessive computation or time, is challenging because of the high state space dimensionality.

Optimisation, as opposed to filtering, approaches to tracking perform well in that they find the best fitting local model. In particular, Stochastic Meta-Descent optimi-

sation performs competently in high-dimensional spaces [11] as it is able to use fewer sample points than many other optimisation schemes due to its stochastic sampling. However, in non-linear spaces with many local optima, optimisation (including SMD) does not guarantee that the global optimum will be reached. This is compounded when clutter and occlusion are experienced.

The chance of reaching the global optimum can be improved by examining multiple hypotheses, which improves robustness to clutter and occlusion. Particle filtering was specially designed for this but is computationally expensive with a high number of dimensions. This is because the number of particles used must drastically increase with the dimensionality to sufficiently cover the entire search space. To overcome this, either the dimensionality has to be reduced or a scheme which uses fewer samples must be employed.

Therefore, the advantages of both SMD and Particle filtering can be combined to form a multiple hypotheses framework, in which SMD is wrapped, to improve the likelihood of reaching the global minimum. This is known as a Smart Particle Filter (SPF) and was employed by Bray *et al.* [13] to perform articulated hand tracking. The algorithm implemented is largely that used by Bray *et al.* [13] but also includes some features taken from Gall *et al.*'s [29] interacting simulated annealing filter.

6.5.3 Implementation

SMD and particle filtering are combined by performing the SMD optimisation after the propagation and prediction step for each of the N samples. This results in N new states $\{(\mathbf{s}_t^{*(n)}, \pi_t^{*(n)})\}_{n=1}^{N}$, each being a 'smart particle'. In this case, the state or smart particle is the body pose, denoted by the pose parameter vector $\mathbf{p}$, while its discrete sampling probability or weight is linked to the corresponding objective function value $f(\mathbf{p})$, which gives an indication of the fit of the body model.

In order to handle multiple hypotheses well, the new optimised particles $\mathbf{s}_t^{*(n)}$ are combined with the 'older' propagation particles $\mathbf{s}_t^{''(n)}$. This is done in a manner which preserves the Bayesian distribution and helps to prevent the set of states from collapsing into only one or a few different states (in other words getting trapped in local optima), as may occur if only the new optimised states are propagated. Importance sampling [43] is used to achieve this combining. Based on the observations, a likelihood function then weights all the particles according to importance with the more important considered to represent a better fitting state than the less important. These weights are normalised in preparation for the next time instant where a new particle set is randomly resampled from the current set based on their weights. This process is repeated for all time frames of tracking. Figure 6.16 provides a flow diagram illustrating this smart particle filter in which the SMD and particle filter is combined.

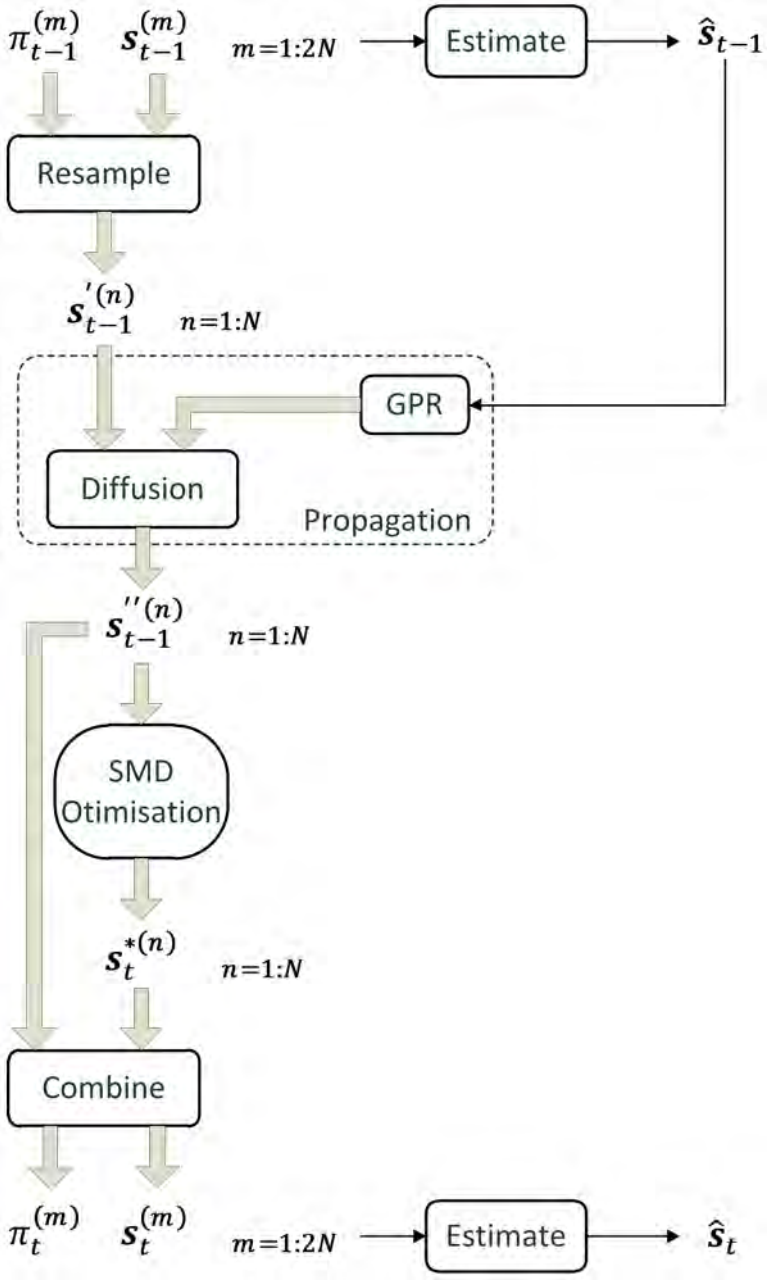


Figure 6.16: Flow diagram of the Smart Particle Filter (SPF) — combining a particle filter and SMD optimisation.

6.5.3.1 Resampling

A particle set consisting of $2N$ samples exists from the previous time frame, $\{(\mathbf{s}_{t-1}^{(m)}, \pi_{t-1}^{(m)})\}_{m=1}^{2N}$. These are resampled to a set of N particles based on their weights, $\{(\mathbf{s}_{t-1}^{'(n)}, \frac{1}{N})\}_{n=1}^{N}$. This is achieved by the selection algorithm, shown in Figure 6.17, which assumes that normalised weights, $\pi^{(m)}$, for each particle are available.

$i = 1$

1. For m from 1 to $2N$
   - Sample κ from $U[0, 1]$
   - If $\kappa \leq \epsilon_t \pi^{(m)}$ then
     * Set $\mathbf{s}_{t-1}^{'(i)} \leftarrow \mathbf{s}_{t-1}^{(m)}$
     * $i = i + 1$
2. If $i < N$
   - For m from 1 to $N - i$
     * Set $\mathbf{s}_{t-1}^{'(i+m)} \leftarrow \mathbf{s}_{t-1}^{(j)}$ with probability $\frac{\pi^{(j)}}{\sum_{k=1}^{2N} \pi^{(k)}}$
       (using binary subdivision and cumulative weights as in [43])

   Else If $i > N$
   - select N particles from set $\{\mathbf{s}_{t-1}^{'(i)}\}$ with greatest $\pi^{(i)}$

Figure 6.17: Resampling selection algorithm for Smart Particle Filter

The resampling parameter, ϵ_t, can be defined in a number of different ways (following Gall *et al.* [28]) so as to slightly alter the resampling algorithm.

1. $\epsilon_t := 0$
   (the selection is made by multinomial resampling)
2. ϵ_t is defined so that $\epsilon_t \pi^{(m)} = \frac{\pi^{(i)}}{\sum_{k=1}^{2N} \pi^{(k)}}$
   (similar to 1, but with all samples tested and particle selection based on cumulative weight)
3. ϵ_t is defined so that $\epsilon_t \pi^{(m)} = \frac{\pi^{(i)}}{\max \pi}$
   (the particle with largest weight $\pi^{(i)}$ will definitely be resampled)

The different resampling parameters where all tried and tested. It was found that option 3 performed best as the particle with the largest weight was then guaranteed to be selected each time.

6.5.3.2 Propagation

Propagation consists of two parts, pose prediction and particle diffusion. Here a new set of N particles, $\{(\mathbf{s}_t^{''(n)}, \frac{1}{N})\}_{n=1}^{N}$, is generated from the set selected in the resampling

stage according to the process density, $\mathbf{s}_t^{''(n)} \approx p(\mathbf{s}_t|\mathbf{s}_{t-1} = \mathbf{s}_{t-1}^{'(n)})$. This process density represents the dynamic model and added noise of the prediction and diffusion propagation components.

Different **pose prediction** models were tested with varying degrees of success as outlined below.

First a simple linear first order model was tried which estimates the new pose based on the movement or displacement between the previous two frames and can be calculated as

$$\mathbf{s}_t = \mathbf{s}_{t-1} + \eta(\mathbf{s}_{t-1} - \mathbf{s}_{t-2}) \tag{6.34}$$

where η is the factor that controls the effect of past change on the current estimate, usually set as a value in the range of [0,1]. This performed reasonable well, giving a very rough estimate of the predicted pose. It, however, did not adapt well to changes in movement direction nor motion speed.

A spline extrapolation prediction was tested but without much success. The number of points, to which the curve was fitted, was varied but did not help to improve the results. The cubic nature of the spline curve fitting seemed to cause problems with the prediction and, in many cases, had too much of a 'bend' at the end.

The last pose prediction test was performed using a 4th order autoregressive model such that $\mathbf{s}_t^{pred} = f(\hat{\mathbf{s}}_{t-1:4})$ where $\hat{\mathbf{s}}_{t-1:4} = (\hat{\mathbf{s}}_{t-1}, \hat{\mathbf{s}}_{t-2}, \hat{\mathbf{s}}_{t-3}, \hat{\mathbf{s}}_{t-4})$ are the previous four estimates. The function f follows the usual autoregressive model

$$x_t = \sum_{i=1}^{p} b_i x_{t-i} + \epsilon_t \tag{6.35}$$

with the autoregressive parameters, b_i ($i \in [1, p]$), being learned online, during tracking, from the history of previous poses estimates.

The autoregression is implemented by Gaussian processes [102] with the prediction determined by a multivariate Gaussian distribution of mean $\mathbf{s}_t^{pred}$ and covariance Σ_t^{pred} — which is the confidence of the prediction.

With the prediction model being learned online, it is able to adapt to the current motion. However, it may also be corrupted by tracking error in previous frames. Therefore, only 50% of the particles are propagated according to $\mathbf{s}_t^{pred}$ with the remaining 50% being passed-through as is.

Once this pose prediction or pass-through has occurred, the particles need to be spread out and are **diffused** by a zero-mean multivariate Gaussian distribution with covariance matrix Σ_t^{pred}, from the pose prediction. The use of the Σ_t^{pred} from the prediction allows the particles to be reasonably spread in the search space without

manually setting the value or bounds for each pose parameter. It also allows the variance to automatically change with frame rate and motion changes to provide the relevant information for particle diffusion.

6.5.3.3 Combine

Importance sampling combines samples from different distributions according to another distribution, $g(\mathbf{s})$, and indicates promising areas in the state space. A correction factor is applied to the sample weights in order to maintain the Bayesian distribution of the sample set during this union. It is calculated as the quotient between the original distribution and the new distribution.

The original distribution $f_t(\mathbf{s}_t^{(n)}) = p(\mathbf{s}_t^{(n)}|\mathbf{z}_{1:t-1})$ is unavailable in a closed form but is approximated as a kernel density estimator [85] with window size h in a d-dimension space

$$f(\mathbf{s}) = \frac{1}{Nh^d} \sum_{n=1}^{N} K\left(\frac{1}{h}(\mathbf{s} - \mathbf{s}^{(n)})\right) \tag{6.36}$$

where

$$K(x) = (2\pi)^{\frac{-d}{2}} e^{-\frac{1}{2}x^T x} \tag{6.37}$$

Kernel density estimation is selected instead of a Gaussian estimation as it only requires the calculation of an easily computed window size. The window size or bandwidth is an important factor in density estimation. If too large a window is used the density will be overly smoothed thus hiding the underlying data. On the other hand, too small a window can lead to a very spiky and difficult to interpret density estimate. However, Silverman Bernard [85] made this selection easy by proposing that the optimal window size can be calculated from the used data via

$$h_{opt} = 1.06\sigma N^{-\frac{1}{5}} \tag{6.38}$$

where the sample variance σ defined as

$$\sigma^2 = \frac{1}{d} \sum_i c_{ii} \tag{6.39}$$

with c_{ii} the diagonal elements obtained from the covariance matrix of the data.

Thus, using the above, the new distribution containing the old and optimised sam-

ples is

$$g(\mathbf{s}) = \frac{1}{2Nh^d}\left(\sum_{n=1}^{N} K\left(\frac{1}{h}(\mathbf{s}-\mathbf{s}^{(n)})\right) + \sum_{n=1}^{N} K\left(\frac{1}{h}(\mathbf{s}-\mathbf{s}^{*(n)})\right)\right) \tag{6.40}$$
$$= \frac{1}{2}(f(\mathbf{s}) + f^*(\mathbf{s}))$$

This leads to the correction factor

$$\frac{f(\mathbf{s})}{g(\mathbf{s})} = \frac{f(\mathbf{s})}{\frac{1}{2}(f(\mathbf{s}) + f^*(\mathbf{s}))} = \frac{2f(\mathbf{s})}{f(\mathbf{s}) + f^*(\mathbf{s})} \tag{6.41}$$

6.5.3.4 Weighting

The correction factor is applied to the individual weights such that the corrected weight is

$$\pi_t^{(m)} = \frac{f_t(\mathbf{s}_t^{(m)})}{g_t(\mathbf{s}_t^{(m)})} p(\mathbf{z}_t|\mathbf{s}_t^{(m)}) \tag{6.42}$$

with the observation density defined as a zero-mean Gaussian, with variance σ^2, evaluated at the objective function value over all sampling points

$$p(\mathbf{z}_t|\mathbf{s}_t^{(m)}) \;\equiv\; p(\mathbf{x}|\mathbf{p}) = G_{0,\sigma}\left(\sum_{S_i} \delta_s(\mathbf{x}, \bar{\mathbf{x}})\right) \tag{6.43}$$

This is calculated at the same time as the objective function in the optimisation phase. The variance, σ^2, is chosen according to the values of the objective function such that poor values receive low weights while favourable values get a much higher weighting. For tracking in this project a value of $\sigma = 0.4$ is used.

6.5.3.5 Pose Estimation

Lastly, the current pose at time t is calculated from the new weighted combined particle set as

$$\hat{\mathbf{s}}_t = \sum_{m=1}^{2N} \pi_t^{(m)} s_t^{(m)} \tag{6.44}$$

Smoothing

The overall pose estimate does not always display continuous movement from one frame to another since it is calculated from a number of weighted particles. The output is thus slightly 'jittery' and noisy. To improve the output, the pose estimation is smoothed with a filter. A number of different filters were examined to achieve this, including moving average, moving weighted window, Gaussian and low-pass Butterworth filters.

However the filter chosen for use is the Savitzky-Golay filter.

The Savitzky-Golay smoothing filter is used on equally spaced data points (which the pose estimate is) and performs similarly to the moving average filter. The difference is that instead of using a linear smooth in the averaging window, the underlying function in the window is approximated by a polynomial using least squares fit. It is thus a moving polynomial fit with the order of the polynomial and window size specified for the filter.

A big advantage of the Savitzky-Golay filter is that it preserves the features of the underlying function (maxima and width) which are flattened by most other methods.

Through testing it was found that two runs of the filter are needed to produce the best results. An order 2, frame width 15, Savitzky-Golay filter smoothes the data before a second filter of order 1, fame width 5, smoothes the smaller 'kinks' left from the first smoothing. This allows the full range of movement (maxima/minima) to be retained while still smoothing the motion. This filtering performed better than the other filters tested. Even though some of them provided more smoothing they clipped off the full range of movement.

6.5.4 Summary of SPF

This smart particle filter requires far fewer samples than an ordinary particle filter with the minimum being represented well by the combined particle set.

Figure 6.18 gives an algorithmic description of the SPF as summarised by Bray *et al.* [13]. The $2N$ sample set is randomly resampled into a N sample set using the particles' weight as described in Section 6.5.3.1. These N samples are then propagated (Section 6.5.3.2) to a new N sample set based on a 4th order autoregressive model with a noise component for diffusion. The propagated samples are optimised with SMD to produce a new set of N samples, giving a total of $2N$ samples (set before and after optimisation). These two sets of samples are joined together using importance sampling in Section 6.5.3.3 whereby a correction factor is applied to the sample weights (Section 6.5.3.4) to maintain the Bayesian distribution. A final state is estimated (Section 6.5.3.5) based on the $2N$ sample set, before the process is repeated for the next time frame.

6.6 Summary

The objective function is a key aspect of optimisation which allows the model and observed data alignment to be evaluated. This consists of both edge and surface alignment and makes use of additional colour information to help with the matching. SMD, an adaptive gradient descent optimisation method, generally provides good tracking

1. for $t > 0$
   (a) **resample** $\{(\mathbf{s}_{t-1}^{(m)}, \pi_{t-1}^{(m)})\}_{m=1}^{2N}$ into $\{(\mathbf{s}_{t-1}^{'(n)}, \frac{1}{N})\}_{n=1}^{N}$
   (b) **propagate**, generate $\mathbf{s}_t^{''(n)} \approx p(\mathbf{s}_t|\mathbf{s}_{t-1} = \mathbf{s}_{t-1}^{'(n)})$ to give $\{(\mathbf{s}_t^{''(n)}, \frac{1}{N})\}_{n=1}^{N}$
   (c) **optimise** $\{(\mathbf{s}_t^{''(n)}, \frac{1}{N})\}_{n=1}^{N}$ with SMD
   for $n \leq N$
   - $\mathbf{v}_0 = 0$; $\mathbf{a}_0 = \mathbf{a}*$; $\mathbf{p}_0 = \mathbf{s}_t^{''(n)}$
   - LOOP $i = 1$ TO convergence
     1) pick sample points S_i;
     2) calculate $\mathbf{g}_i$ (6.22) and $\mathbf{a}_i$ (6.14);
     $\rightarrow$ if $i = 1$ then $\mathbf{a}* = \mathbf{a}_1$;
     3) calculate $\mathbf{p}_{i+1}$ (6.13);
     4) calculate $\mathbf{p}_{i+1}^c$ (6.31);
     5) calculate $\mathbf{g}_i^c$ (6.32), ; $\mathbf{H}_i\mathbf{v}_i$ (6.23), $\mathbf{v}_{i+1}$ (6.15).
   - $\mathbf{s}_t^{*(n)} = \mathbf{p}_{\text{converged}}^c$
   (d) **combine** $\{(\mathbf{s}_t^{(m)}, \frac{1}{2N})\}_{m=1}^{2N} = \{(\mathbf{s}_t^{''(n)}, \frac{1}{2N})\}_{n=1}^{N} \cup \{(\mathbf{s}_t^{*(n)}, \frac{1}{2N})\}_{n=1}^{N}$
   compute correction factor
   $k_t^{(m)} = f_t(\mathbf{s}_t^{(m)})/g_t(\mathbf{s}_t^{(m)})$, $1 \leq m \leq 2N$
   (e) **weight**, set $\pi_t^{(m)} \propto k_t^{(m)} p(\mathbf{z}_t|\mathbf{s}_t = \mathbf{s}_t^{(m)})$
   to give $\{(\mathbf{s}_t^{(m)}, \pi_t^{(m)})\}_{m=1}^{2N}$ normalised so that $\sum_{m=1}^{2N} \pi_t^{(m)} = 1$
   (f) **estimate** $\hat{\mathbf{s}}_t = \sum_{m=1}^{2N} \pi_t^{(m)} s_t^{(m)}$

Figure 6.18: Algorithmic description of the Smart Particle Filter [13]

for high dimensional state space problems. It makes use of the gradient measure which is able to be calculated explicitly.

The simultaneous optimisation of all parameters has some draw-backs due to the nature of the human body. All parts are not independent but instead consist of linked kinematic chains. A hierarchical approach helps improve optimisation by using this fact and optimising in an outward fashion, starting first at the base (torso) and then moving out along the kinematic chains (limbs).

Another improvement is to include pose prediction, in the optimisation process, to increase the pose optimisation accuracy (and possibly speed). It achieves this by calculating a future pose estimate based on past poses.

However the two additions (hierarchical approach and pose prediction) did not provide the expected improvement. The Smart Particle Filter did provide this improvement by allowing multiple hypotheses to be tested each frame. It does this through the use of a particle filter framework, in which SMD optimisation is wrapped to handle the high dimensionality of the state space.

Chapter 7

Results and Discussion

Selected pose tracking results are shown in Figures 7.2 – 7.7 and 7.9 – 7.14 . These tests were performed on the HumanEvaII [83] datasets. The datasets consist of captured images of two people (known as S2 and S4) performing a few common actions (walking, running and balancing) in a defined central space. Four surrounding colour cameras were used to capture these movements, together with a marker-based ViconPeak motion capture system which obtained the ground-truth data.

The error results displayed in this chapter are a calculation of the average Euclidean distance between the individual virtual markers of the estimated pose and the marker-based ground truth pose. These virtual markers are the various joint and limb endpoint locations of the human body.

Tests of the pose tracking system were performed using the different optimisation approaches discussed in Chapter 6. The SMD with Hierarchical and SMD with Pose Prediction approaches were combined and examined together as SMD with Hierarchical and Prediction (SMD H&P). This is because the best improvement was observed with them combined. In addition, tests were also performed with just a Particle Filter approach. The various SMD and SPF approaches were coded from the ground up, but the PF results were obtained from code provided by Sigal *et al.* [84]. 200 particles were used for this PF compared to only 12 particles used by the SPF. The results below demonstrate how each approach performs with the two subjects (S2, Section 7.1, and S4, Section 7.2) on the different action sequences — walking, jogging and balancing.

Multiple runs of each approach were carried out on the action sequences. For each approach, mean and minimum pose error values of the various tracking results were calculated and statistical analyses of these mean and minimum error results were performed. This was undertaken to ensure that a quantitative evaluation of the results could be presented. Paired t-tests, for two related samples, and Friedman's test, a non-parametric test for three or more related samples, were used for this analysis. A 5% level of confidence ($p = 0.05$) is used throughout. The results from these statistical

tests are presented in their respective action sequence sections.

7.1 Subject 'S2' Results

For subject 'S2', the walking, jogging and balancing actions follow on from one another in a continuous sequence. This is clearly seen in the graph of Figure 7.1 where the error results of the individual actions are labelled and separated by grey bars.

As the names imply, the walking and jogging actions consist of the subject walking and then jogging in a circle within the capture space. The balancing action consists of the subject moving from balancing on one leg to the other.

The walking and jogging actions are a lot more difficult for pose tracking than the balancing, since a lot more occlusion occurs with the subject moving in a circle. This occlusion is between both the arms and torso as well as the separate limbs themselves. The problem is further compounded with the use of only four sub-optimally placed cameras. This can result in increased clutter, especially with the 3-D volumetric reconstruction. A better camera setup could have been used with cameras not diagonally opposite each other and possibly with a camera view from the top. Most results from other markerless pose tracking systems use more camera views, with some even having 16 simultaneous views [51].

While it is possible to start at frame 1 (or near the beginning) and work forward, a lot of initial testing of the pose tracking algorithms, on the HumanEvaII datasets, was performed in the balancing section — with the exception of the PF algorithm. This was done because this action is the easiest of the three to track as the legs and arms are generally spread out and separate from the torso and from each other.

For subject 'S2', testing started at frame 1049, where the subject is in a clear, 'spread eagle' pose, and moved backward. As a consequence, results from all four approaches are observed in most of the balancing section and part of the jogging section. SMD H&P results are also shown in the walking section but no SMD optimisation results are displayed here. This is because the SMD optimisation performed very poorly in these walking and jogging sequences with tracking results diverging after only a few frames.

Table 7.1: Table of average mean errors and their standard deviations (mm) for the Walking, Jogging and Balancing sections of S2

Approach	Walking - S2		Jogging - S2		Balancing - S2	
	Avg Mean	Std Dev	Avg Mean	Std Dev	Avg Mean	Std Dev
SMD	-	-	182.97	33.14	119.58	15.07
SMD H&P	192.57	59.91	165.72	22.72	117.87	13.75
PF	135.80	50.62	242.56	27.62	396.32	63.72
SPF	136.17	15.26	146.19	13.11	105.06	12.80

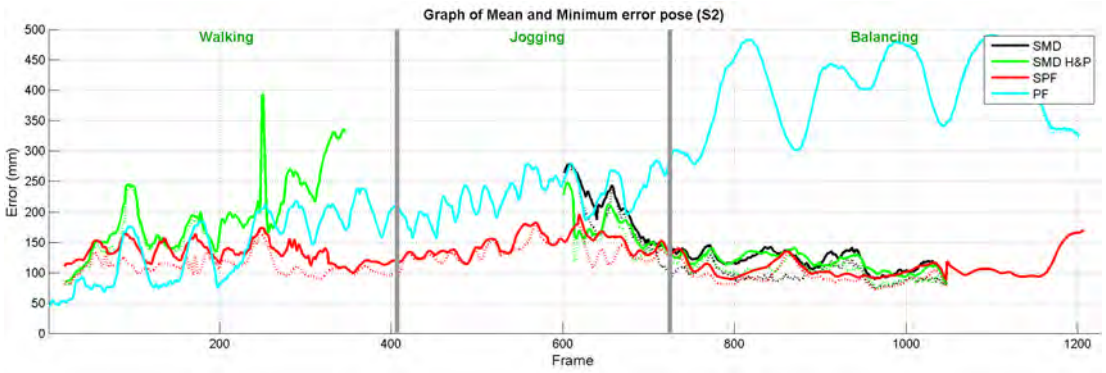


Figure 7.1: Graph of mean (solid line) and minimum (dotted line) pose errors for the walking, jogging and balancing sections of S2.

Table 7.2: Table of average minimum errors and their standard deviations (mm) for the Walking, Jogging and Balancing sections of S2

Approach	Walking - S2		Jogging - S2		Balancing - S2	
	Avg Min	Std Dev	Avg Min	Std Dev	Avg Min	Std Dev
SMD	-	-	161.94	35.63	96.06	12.15
SMD H&P	187.59	62.51	151.85	23.96	103.17	14.21
PF	135.80	50.62	242.56	27.62	398.02	64.31
SPF	111.43	15.27	126.68	9.35	92.91	13.79

7.1.1 Walking and Jogging

The walking and jogging actions are included together since they are similar actions with one just happening a little quicker than the other.

The particle filter based approaches perform much better with these actions than the SMD optimisation approaches. The main reason is that they are able to handle multiple hypotheses and recover from errors while SMD is not.

The pose error results for the walking and jogging sequences of subject S2 are displayed in Figure 7.1. The graph gives the mean (solid line) and minimum (dotted line) pose errors, over several runs, for the different approaches. Table 7.1 and Table 7.2 display the average mean and minimum error results, together with their standard deviations, for the walking (frame 20 – 346) and jogging (frame 601 – 725) sections in which results from all appropriate approaches are available. The error results over the whole jogging action sequence for the SPF and PF approaches are also examined with these result values given in the text.

From the graph it is clear that the SPF performed better than the PF throughout the jogging section. In the small section where results for all four approaches are available, the SPF still shows the best performance, followed by SMD H&P and then SMD with the PF approach showing the worst performance. In the walking sequence,

SMD H&P is clearly the worst of the three approaches, however the error results for SPF and PF are a lot closer. The SPF appears to perform better on average while the PF contains some frames of smaller pose error, but fluctuates more, making it difficult to judge qualitatively.

7.1.1.1 Statistical Results

Statistical analysis of the results show a significant ($p < 0.0005$) difference between the SMD H&P approach and the SPF and PF approaches when looking at both the mean and minimum error results for walking. The SMD H&P approach has the largest average errors and thus performed the worst in this sequence. There is no significant ($p = 0.889$) difference between the mean error results for the SPF and PF approaches in the examined section. However, there is a significant ($p < 0.0005$) difference with the minimum error results. In this case the SPF performed better then the PF approach with a lower average error value.

In the small jogging section, where results from all approaches are available, a significant ($p < 0.0005$) difference exists between all the approaches. The SPF performed best with the lowest average mean and minimum values. This is followed by SMD H&P, SMD and lastly PF which has the worst average results.

When examining the whole jogging sequence, there is once again a significant ($p < 0.0005$) difference between the SPF and PF error results from frame 407 – 725. SPF performed better with lower average error values of mean 146.13mm (18.00) and minimum 134.22mm (17.14) compared the PF values of 227.78mm (31.55) and 228.21mm (31.97) respectively.

7.1.1.2 Discussion

Graphical output of the walking and jogging action pose results for S2 can be seen in Figures 7.2 and 7.3 respectively. These figures show comparative results between the SMD H&P (top row), PF (middle row) and SPF (bottom row) approaches for walking and SMD, SMD H&P, PF and SPF (top to bottom rows) for jogging. All results displayed show the original captured image from some camera view, overlaid with the body model in the recovered pose. This body model shows the underlying skeletal structure (red lines linking the green circle joints) with the outline of the superellipsoid shapes (dotted blue lines). For good pose tracking this overlaid body model should coincide with the person in the underlying image. As mentioned, the PF tracking results were obtained using code provided by Sigal *et al.* [84]. This code uses a different body model and hence the graphical output is also displayed using this body model.

Four principle failures were identified in these two action sequences. They are: poor recovery from error (with the SMD and SMD H&P approaches), lack of lower torso

rotation (in the PF approach), limb tracking errors with fast, occluded motion (in the PF approach), and torso misalignment (with SMD, PF and SPF).

The **poor recovery from error** for the two SMD approaches was observed in both the walking and jogging sequences. The SMD H&P tracking started off reasonably well in the walking sequence. However, the moment occlusion and clutter problems were encountered, errors occurred from which there was never any recovery. This is seen in frame 116, where the model legs became joined, and frame 200, where the error was further increased with additional misalignment of the torso. Soon thereafter, tracking failed completely and was terminated.

Similar observations were made in the jogging sequence of Figure 7.3. As mentioned, tracking was performed backward from frame 1049 for these SMD based approaches. After the balancing, the legs are reasonably well aligned, as seen in frame 717 for both the SMD and SMD H&P approaches. However, shortly thereafter, in frame 661, poor torso, arm and leg alignment was observed, which was followed by complete tracking failure. This again demonstrates that even though the SMD H&P approach performs better than the SMD approach, both are unable to recover when any tracking errors are encountered.

The PF's **lower torso rotation failure** was observed indirectly through leg tracking mismatch. In the first frame of the walking sequence (frame 20, Figure 7.2), it can be seen that the left leg and arm of the body model are displayed in blue and the right in yellow. However, in frame 116 there was a leg mismatch with the left leg displayed in yellow and the right in blue. In other words, there was a leg tracking failure. Similar observations were made in later frames with the legs back to the correct tracking in frame 200 but mismatched once again in frame 279. The underlying reason for this is not due to leg tracking errors but rather due to a lack of lower torso rotation. This error was in a sense 'corrected' because for the rest of the walking and jogging actions, from frame 279 on, the leg tracking remained consistent — left leg yellow and right leg blue. However, the incorrect leg match resulted in an increased error measurement. As the legs moved closer to each other, during the cross-over, the error decreased, resulting in the fluctuating error graph of Figure 7.1.

The PF approach also experienced **limb tracking problems** in fast, occluded motion sequences. This was observed with the arm tracking in the jogging sequence, Figure 7.3. In this sequence, poor arm tracking was detected as the arms moved quickly in front of, and behind, the torso.

In both the SPF and PF approaches, minor **torso alignment errors** were experienced. This was noticed by observing the head which is seen to be bent when it should not be, as displayed in frame 200, for SPF results and the lower part of the torso which

is sticking out, frame 200, for PF results. The main reason for this is the faster motion combined with the indistinct shape and outline of the torso which makes it difficult to accurately track in every frame.

7.1.2 Balancing

The pose error results for the balancing sequence of subject S2 are displayed in Figure 7.1 with the mean (solid line) and minimum (dotted line) pose errors, over several runs, for the different approaches. The average mean and minimum error results for the balancing section (frame 726 – 1049) are included in Table 7.1 and Table 7.2.

Apart from the PF approach, all the other approaches show good performance in the balancing section. It is difficult to judge qualitatively, from the graph, which approach performed the best, but the SPF approach appears to display slightly lower error results than the rest.

7.1.2.1 Statistical Results

Statistical analysis of the results show a significant ($p \leq 0.001$) difference between all approaches for both the mean and minimum error results. For both the minimum and mean results, the average error is lowest for SPF and highest for PF. This indicates that the SPF performed best in the balancing sequence and the PF worst. The SMD and SMD H&P approaches occupy the middle two spots, with the SMD approach performing better (lower average result) than SMD H&P when examining the mean results but worse when examining the minimum results. This indicates that these two approaches did not show much difference in performance in this section.

7.1.2.2 Discussion

Graphical output of the balancing action pose results for S2 can be seen in Figure 7.4. This shows a comparison of results between the SMD (top row), SMD H&P (second from top row), PF (second from bottom row) and SPF (bottom row) approaches.

The tracking problems observed with the PF approach in the walking and jogging action sequences continued through to the balancing section. Even though there was not a complete tracking failure, as was the case with the SMD and SMD H&P approaches in the walking and jogging sequences, the model legs continued to track the incorrect leg and the arms experienced problems due to a lack of torso rotation. The arms and torso did recover slightly, if one observes the difference in model arm positions (especially upper shoulder area) between frame 763 and 995, but never completely.

Good results are shown for the other three approaches in this section. SPF occasionally displays marginally better results than the SMD and SMD H&P approaches,

as in the torso alignment in frame 903, but in general they are very close to one another.

The SPF approach was observed to fare much better in all action sequences for subject 'S2' than the other optimisation approaches, confirming the presented error results. That is, generally good tracking with occasional minor loss of tracking followed by tracking recovery.

Figures 7.5 – 7.7 show the output of only the SPF results from two camera views along with the resultant coloured body model for the walking, jogging and balancing sequences. The coloured patches of the model can clearly be seen here.

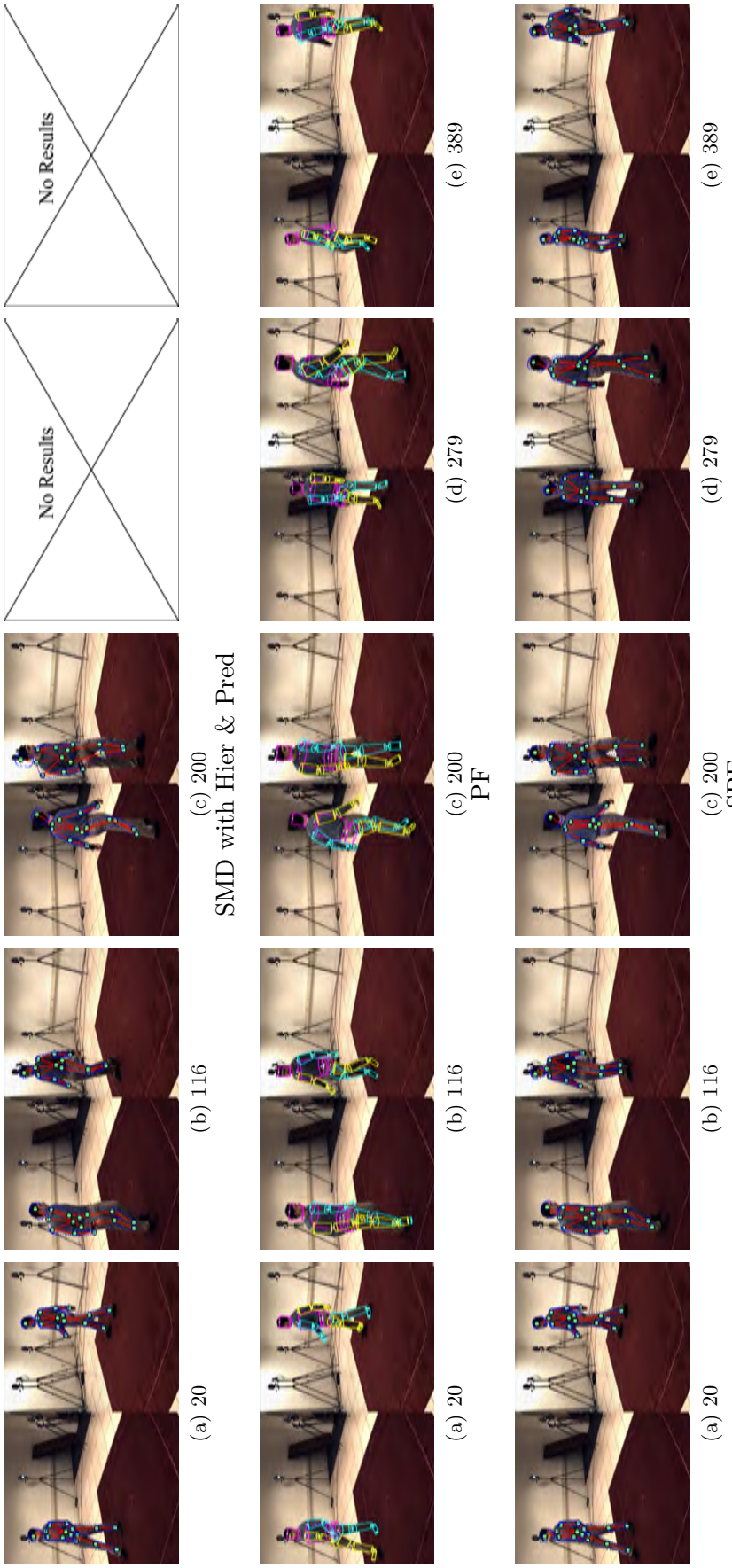


Figure 7.2: Graphical comparison of tracking results for selected frames of walking sequence for S2 using SMD with Hierarchical & Prediction, PF and SPF approaches.

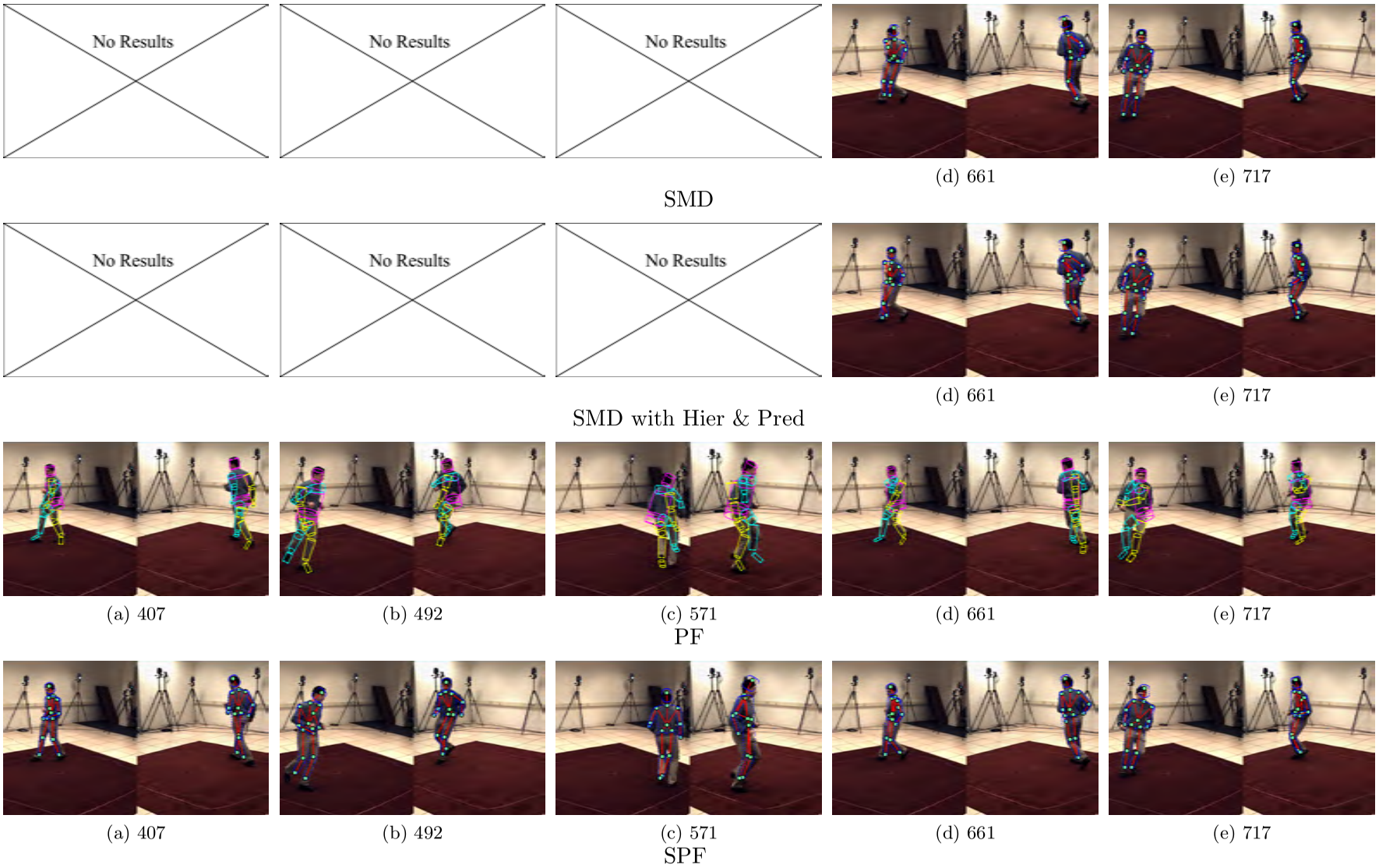
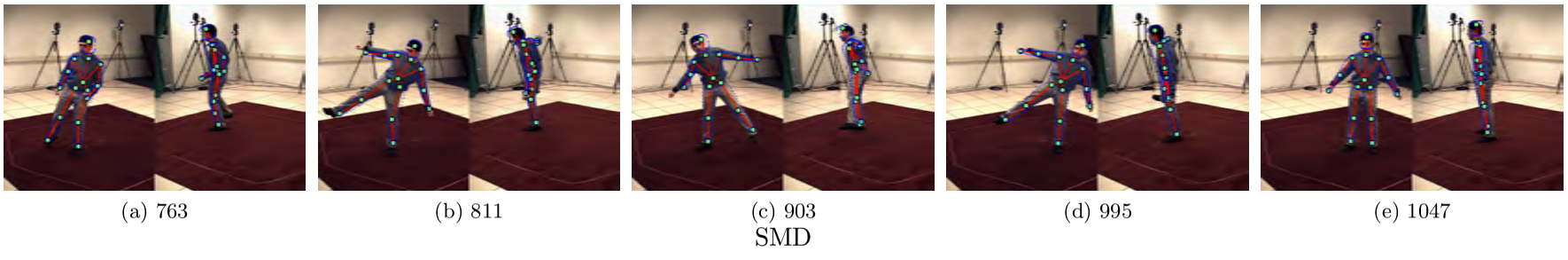
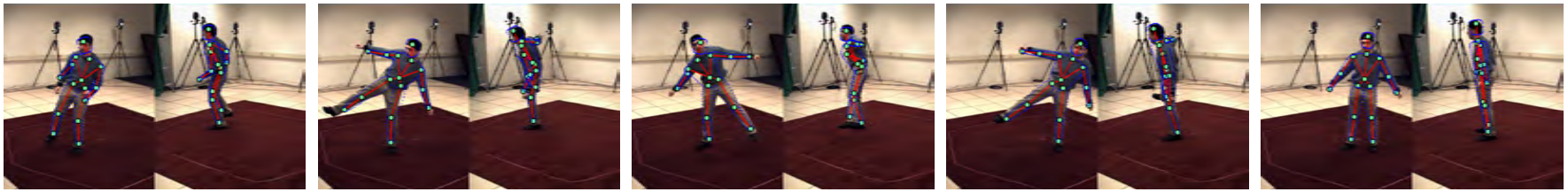


Figure 7.3: Graphical comparison of tracking results for selected frames of jogging sequence for S2 using SMD, SMD with Hierarchical & Prediction, PF and SPF approaches.



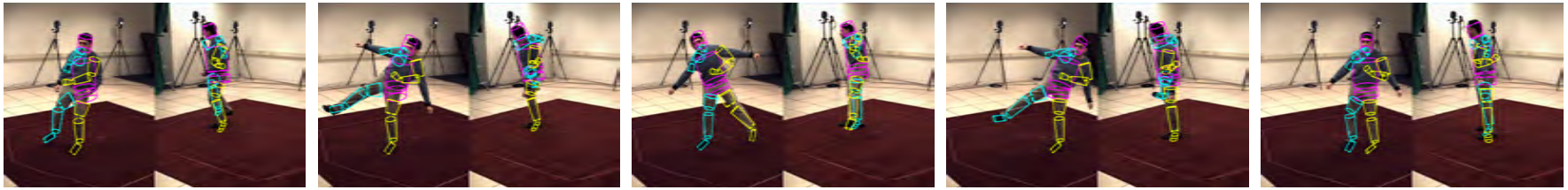
(a) 763 (b) 811 (c) 903 (d) 995 (e) 1047

SMD



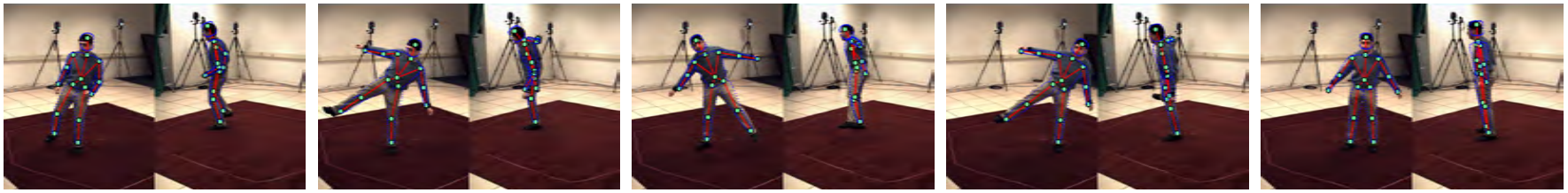
(a) 763 (b) 811 (c) 903 (d) 995 (e) 1047

SMD with Hier & Pred



(a) 763 (b) 811 (c) 903 (d) 995 (e) 1047

PF



(a) 763 (b) 811 (c) 903 (d) 995 (e) 1047

SPF

Figure 7.4: Graphical comparison of tracking results for selected frames of balancing sequence for S2 using SMD, SMD with Hierarchical & Prediction, PF and SPF approaches.

Figure 7.5: SPF tracking results for selected frames of the walking sequence for S2 — views from cam1, cam2 and 3-D model view from the perspective of cam1, shown from top to bottom

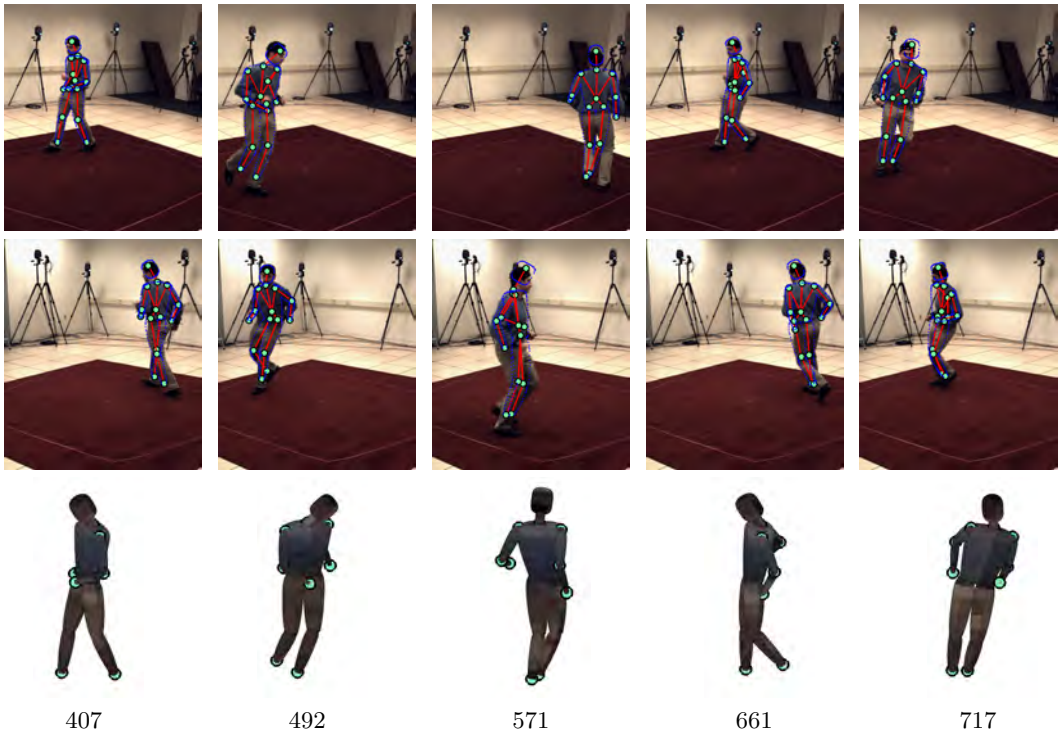


Figure 7.6: SPF tracking results for selected frames of the jogging sequence for S2 — views from cam1, cam2 and 3-D model view from the perspective of cam1, shown from top to bottom

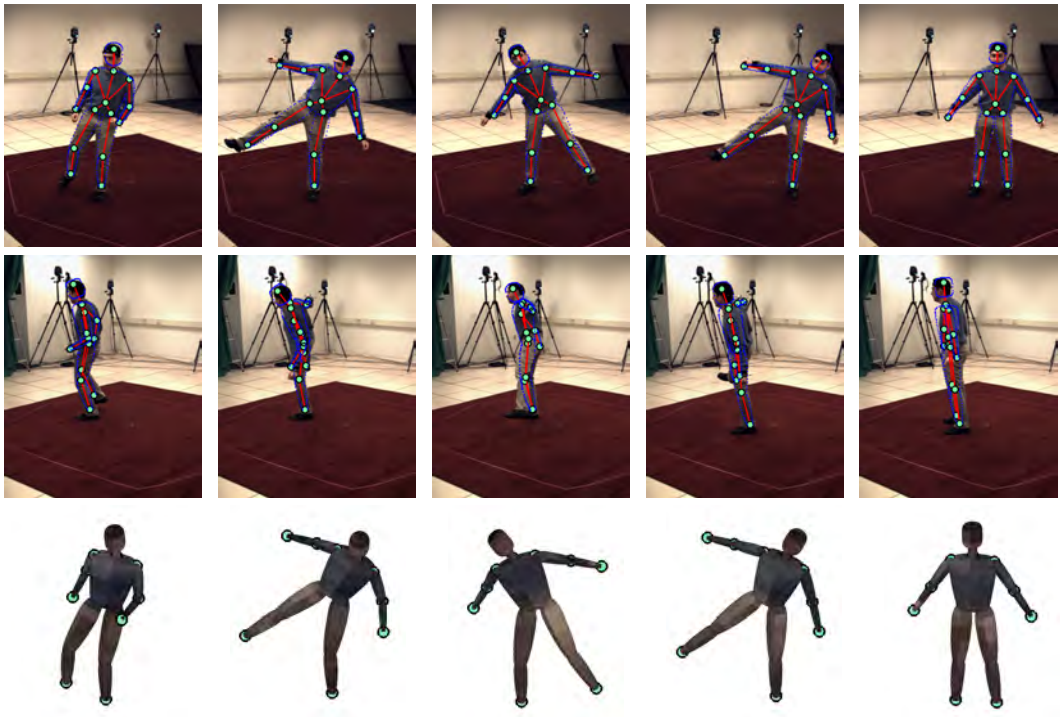


Figure 7.7: SPF tracking results for selected frames of the balancing sequence for S2 — views from cam1, cam2 and 3-D model view from the perspective of cam1, shown from top to bottom

7.2 Subject 'S4' Results

As was the case with the subject 'S2', the walking, jogging and balancing actions again follow on from one another in a continuous sequence for subject 'S4'. The error results for these action sections are displayed in Figure 7.8, separated by grey bars.

Once again, initial testing, except for the PF approach, was carried out on the balancing action sequence. For this subject, 'S4', the testing started at frame 822 and continued forward. Later on, further testing was performed by starting at the beginning of the walking sequence.

Results from all four optimisation approaches are displayed for the balancing action sequence. However, no SMD results are available in either the walking or jogging sequences and SMD H&P results are only available for the walking sequence. This is because the tracking of these two approaches performed very poorly in these sections, with the pose tracking diverging irrecoverably after only a few frames.

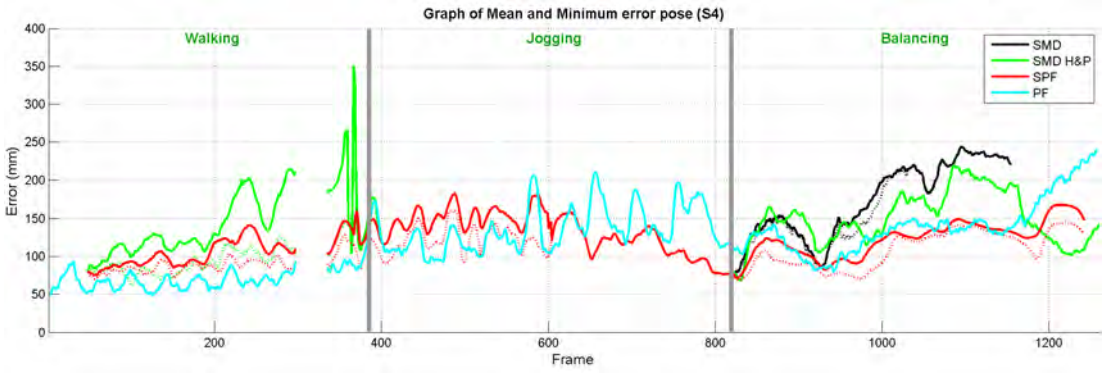


Figure 7.8: Graph of mean (solid line) and minimum (dotted line) pose errors for the walking, jogging and balancing sections of S4.

Table 7.3: Table of average mean errors and their standard deviations (mm) for the Walking, Jogging and Balancing sections of S4

Approach	Walking - S4		Jogging - S4		Balancing - S4	
	Avg Mean	Std Dev	Avg Mean	Std Dev	Avg Mean	Std Dev
SMD	-	-	-	-	169.16	47.71
SMD H&P	134.86	35.70	-	-	153.04	34.14
PF	65.95	8.99	135.65	28.89	122.68	20.19
SPF	101.99	16.60	133.06	25.88	115.55	20.34

7.2.1 Walking and Jogging

The walking and jogging actions are again examined together since they are similar actions.

Table 7.4: Table of average minimum errors and their standard deviations (mm) for the Walking, Jogging and Balancing sections of S4

Approach	Walking - S4		Jogging - S4		Balancing - S4	
	Avg Min	Std Dev	Avg Min	Std Dev	Avg Min	Std Dev
SMD	-	-	-	-	163.84	49.93
SMD H&P	91.30	15.16	-	-	153.04	34.14
PF	65.95	8.99	135.65	28.89	122.68	20.19
SPF	85.58	7.29	118.75	20.16	102.46	23.96

The particle filter based approaches performed much better with these actions than the exclusively SMD based optimisation. As mentioned with subject S2, this is due to their ability to examine multiple hypotheses and recover from errors which SMD does not have.

The pose error results for the walking and jogging sequences of subject S4 are displayed in Figure 7.8. The graph gives the mean (solid line) and minimum (dotted line) pose errors, over several runs, for the different approaches; while Table 7.3 and Table 7.4 display the average mean and minimum error results, together with the standard deviations, for the walking (frame 47 – 385) and jogging (frame 386 – 819) sections. Note that there are no error results from frame 298 – 334. This is due to poor ground truth motion capture results which led to Sigal and Black [83] suggesting that these frames not be used for evaluation.

From the graph it can be seen that the SMD H&P approach performed worst in the walking action sequence, followed by SPF, with PF showing the best results. In the jogging sequence it is more difficult to qualitatively discern as the PF approach seems to show better results in the beginning while SPF displays lower error results in the second half of the jogging.

7.2.1.1 Statistical Results

Statistical analysis of the results for the walking sequence reinforces what is observed. There is a significant ($p < 0.0005$) difference between the SMD H&P, PF and SPF mean and minimum error results. The PF produced the best results with the lowest average error values followed by SPF and SMD H&P last with the highest average error values and thus the worst results.

For the jogging section, there is no significant ($p = 0.202$) difference between the PF and SPF approaches when examining their mean error results. However, the minimum error results show a significant ($p < 0.0005$) difference with the SPF having a lower average minimum error and thus displaying better results than the PF.

7.2.1.2 Discussion

Graphical output of the walking and jogging action pose results for selected frames of S4 are displayed in Figures 7.9 and 7.10. These show a comparison of results between the SMD H&P (top row), PF (middle row) and SPF (bottom row) approaches for walking as well as PF (top row) and SPF (bottom row) for jogging. Again the original captured images are shown with the recovered body model overlaid. For good pose tracking results this overlaid body model should coincide with the subject in the underlying image.

Three principle failures, similar to those experienced with subject S2, were identified in the walking and jogging action sequences. They are: poor recovery from error (with the SMD H&P approach), limb tracking errors with fast occluded motion (in PF and SPF approaches), and torso misalignment (with SMD H&P, PF and SPF).

The SMD H&P tracking began well. However, after tracking errors were encountered, as observed in frame 287 where the legs joined and torso lagged, the errors 'snow-balled' and there was never, or at best only poor, **recovery from error**. This was observed in the error graph, Figure 7.8, where the errors continued to grow up to the end of the walking sequence when the pose tracking failed completely with the SMD H&P approach.

Both the PF and SPF approaches experienced **limb tracking problems** in fast, occluded motion sequences. This was observed with the arm tracking in the jogging sequence, Figure 7.10. In this sequence, poor arm tracking was detected as the arms moved quickly in front of, and behind, the torso. Recovery from this was observed in both cases. With the SPF results, the right arm is seen to join the body in frame 405 but shows partial recovery in frame 596 and full recovery by frame 790. Similarly, the PF results show a loss of tracking of the left (blue) arm in frame 405 which recovered by frame 596. However, this left arm tracking was then lost again in frame 790.

The last principle failure was that of minor **torso alignment errors**, observed with all approaches. For SMD H&P this error was easily observed in frame 287. Frame 501 shows an example of torso misalignment for the PF approach, especially lower torso, while this error for the SPF approach was observed in frame 596 where the torso appears to have rotated too much. This is a relatively minor pose tracking error which generally only has a small effect on the recovered pose. The main reason for this misalignment is the faster motion combined with the indistinct shape and outline of the torso which makes it difficult to accurately track in every frame.

7.2.2 Balancing

The pose error results for subject S4's balancing sequence are displayed in Figure 7.8 with the mean (solid line) and minimum (dotted line) pose errors, over several runs, given for the four different approaches. Table 7.3 and Table 7.4 include the average error for these mean and minimum error results over the section in which results for all the approaches are available (frame 822 – 1154).

By examining the graph it is clear that the SMD approach performed worst, closely followed by SMD H&P. The SPF and PF approaches show similar results but the SPF out-performed the PF method with slightly lower error values.

7.2.2.1 Statistical Results

The above assessment is reflected in the statistical results. The SMD, SMD H&P, PF and SPF results are all significantly ($p < 0.0005$) different from each other for both the mean and minimum error results. With the highest average mean and minimum error values, the SMD approach produced the worst results followed by SMD H&P. SPF has the lowest average error values and thus the best results with the PF approach second (see Table 7.3 and Table 7.4).

7.2.2.2 Discussion

A graphical display of the balancing action pose results for S4 can be seen in Figure 7.11. This shows a comparison of the SMD (top row), SMD H&P (second row) , PF (third row) and SPF (bottom row) methods at selected frames. The original captured images are shown with the recovered body model overlaid.

This balancing sequence is different from that of subject S2, in that the S4 subject performs a 180° rotation while moving from balancing on one leg to the other. This made it a slightly more challenging action sequence to track, which led to a greater spread in the error results between the approaches compared to that seen with subject S2.

With fairly basic movements observed in this sequence, tracking generally performed well. The only principle failure identified was lack of recovery from error.

The SMD tracking started off well, but encountered tracking failures in frame 976. This was due to leg and arm matching problems from which the approach was not able to recover. A full tracking failure occurred by frame 1154.

Similar errors were observed with the SMD H&P approach, first in frame 868 with the left arm and later in frame 1084 with the right leg. However, unlike the SMD approach, SMD H&P did not experience a complete tracking failure.

Partial recovery from error was achieved with the PF tracking approach. An example of this recovery is seen between frames 868 and 976, where the upper right

arm was able to recover from being linked to the torso. However, there was never complete recovery as observed throughout the sequence with the poor limb (especially arm) tracking.

The SPF approach demonstrated good recovery from error. This is illustrated in the graphical output where incorrect matching of the lower right and upper left arms in frame 1084 was corrected by frame 1192. The SPF tracking also shows much better matching, in general, than the PF tracking. This is because the particle filter part of SPF guides the tracking to the region of global minimum with the SMD improving the match locally.

It can be observed that the PF approach produced better results in the walking section than the SPF approach. The jogging section gave very similar results with SPF being slightly better and the final balancing section providing better SPF error results.

Statistical analysis over all three sections together (frame 47 – 1257), shows a significant ($p < 0.0005$) difference between the PF and SPF approaches for both the mean and minimum error results. The PF performed better when looking at the mean values, with an average mean of 118.5mm compared to that of 122.6mm for SPF. However, the SPF approach shows better minimum error results, with a smaller average minimum value of 107.5mm compared to 118.5mm of the PF.

This demonstrates that these two optimisation approaches produced similar results for the action sequences of subject S4. These two approaches also produced better results — generally good pose tracking with occasional minor loss followed by recovery of tracking — than any of the exclusively SMD based methods.

Figures 7.12 – 7.14 show the output of only the SPF results from two camera views along with the resultant coloured body model for the walking, jogging and balancing sequences.

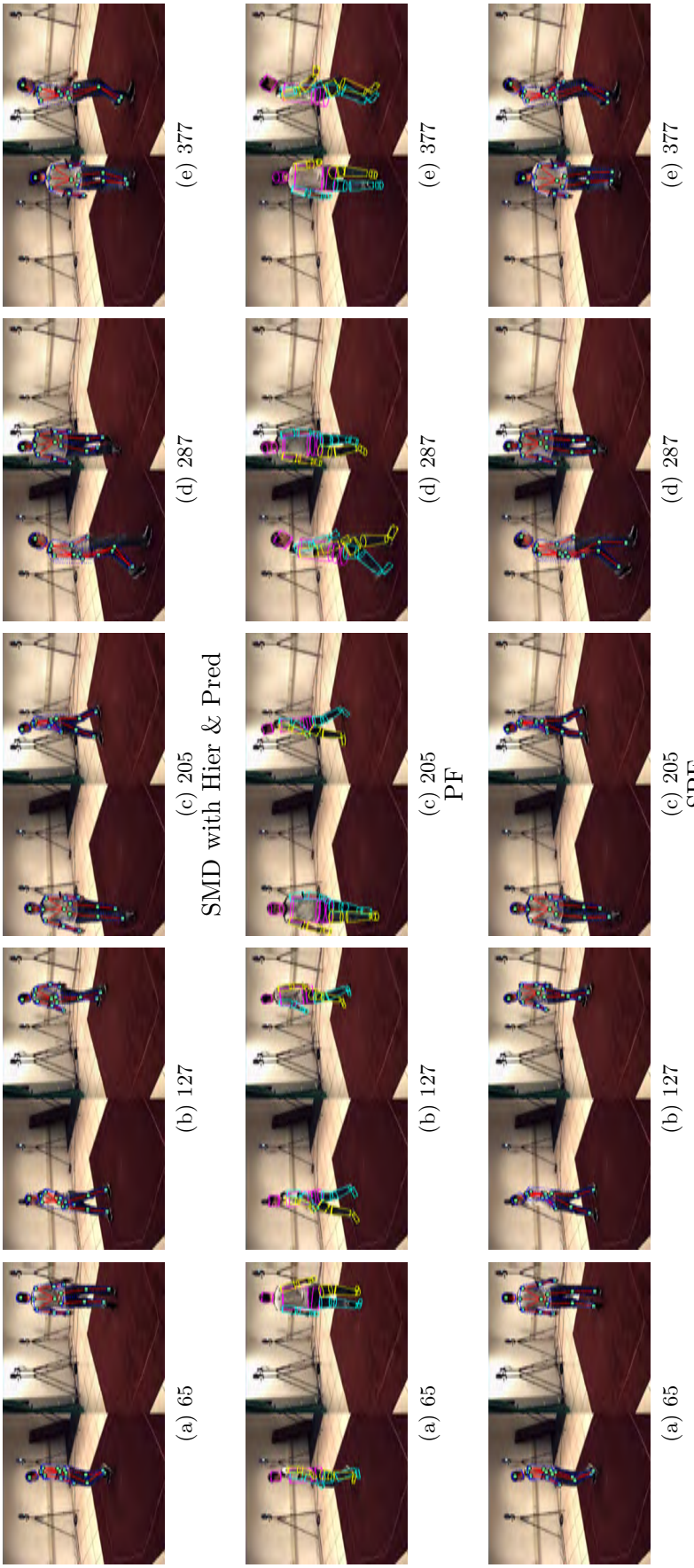


Figure 7.9: Graphical comparison of tracking results for selected frames of walking sequence for S4 using SMD with Hierarchical & Prediction, PF and SPF approaches.

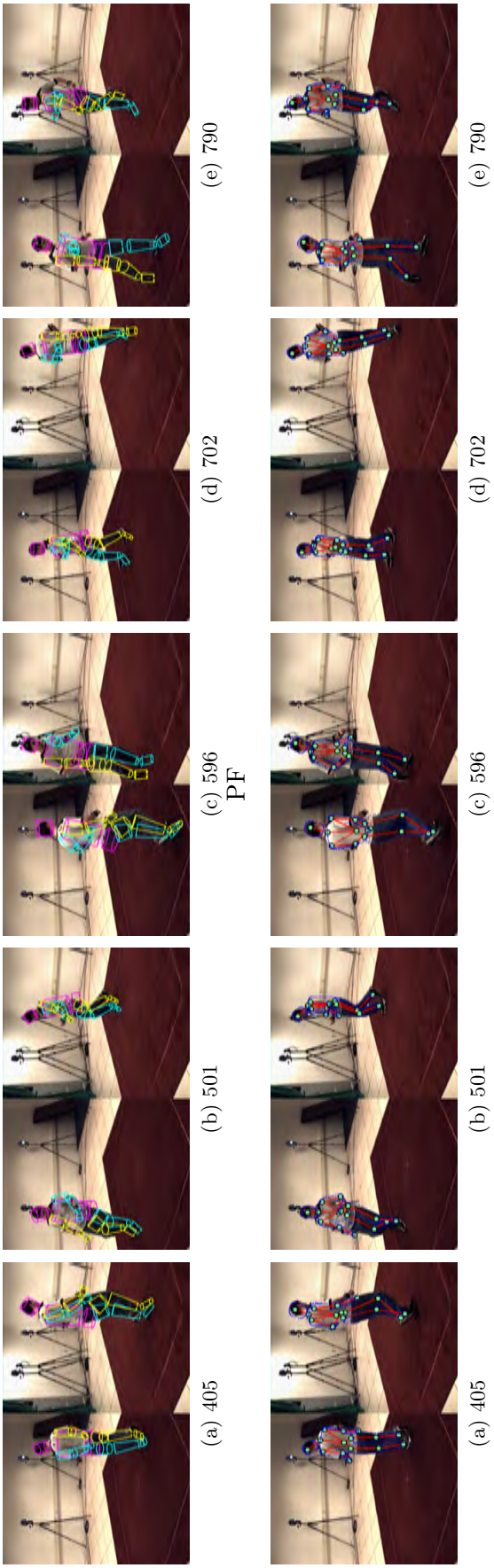


Figure 7.10: Graphical comparison of tracking results for selected frames of jogging sequence for S4 using PF and SPF approaches.

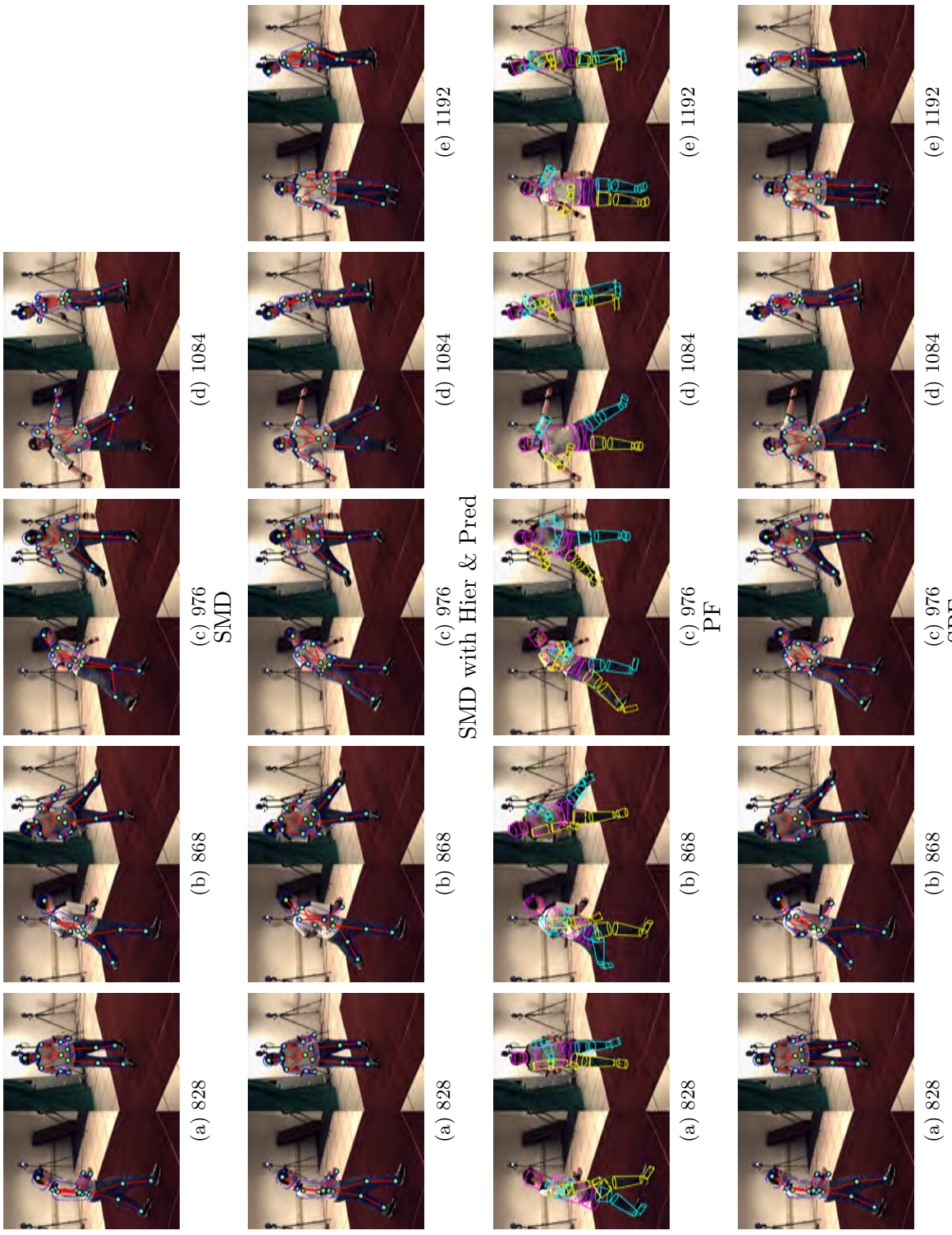


Figure 7.11: Graphical comparison of tracking results for selected frames of balancing sequence for S4 using SMD, SMD with Hierarchical & Prediction, PF and SPF approaches.

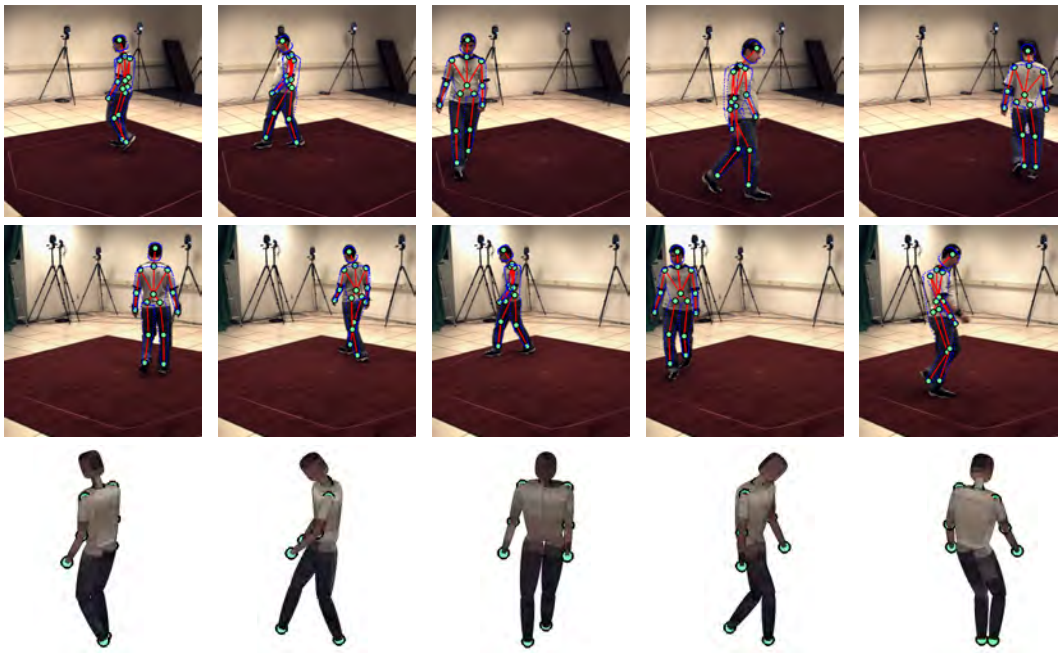


Figure 7.12: SPF tracking results for selected frames of the walking sequence for S4 — views from cam1, cam2 and 3-D model view from the perspective of cam1, shown from top to bottom

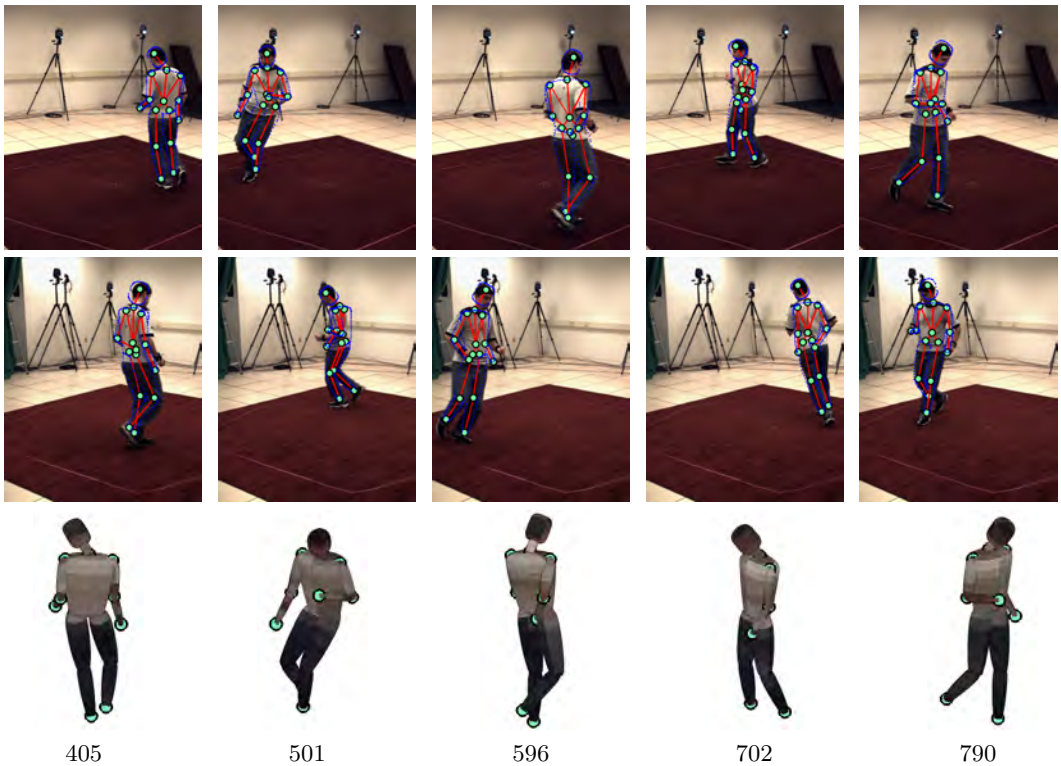


Figure 7.13: SPF tracking results for selected frames of the jogging sequence for S4 — views from cam1, cam2 and 3-D model view from the perspective of cam1, shown from top to bottom

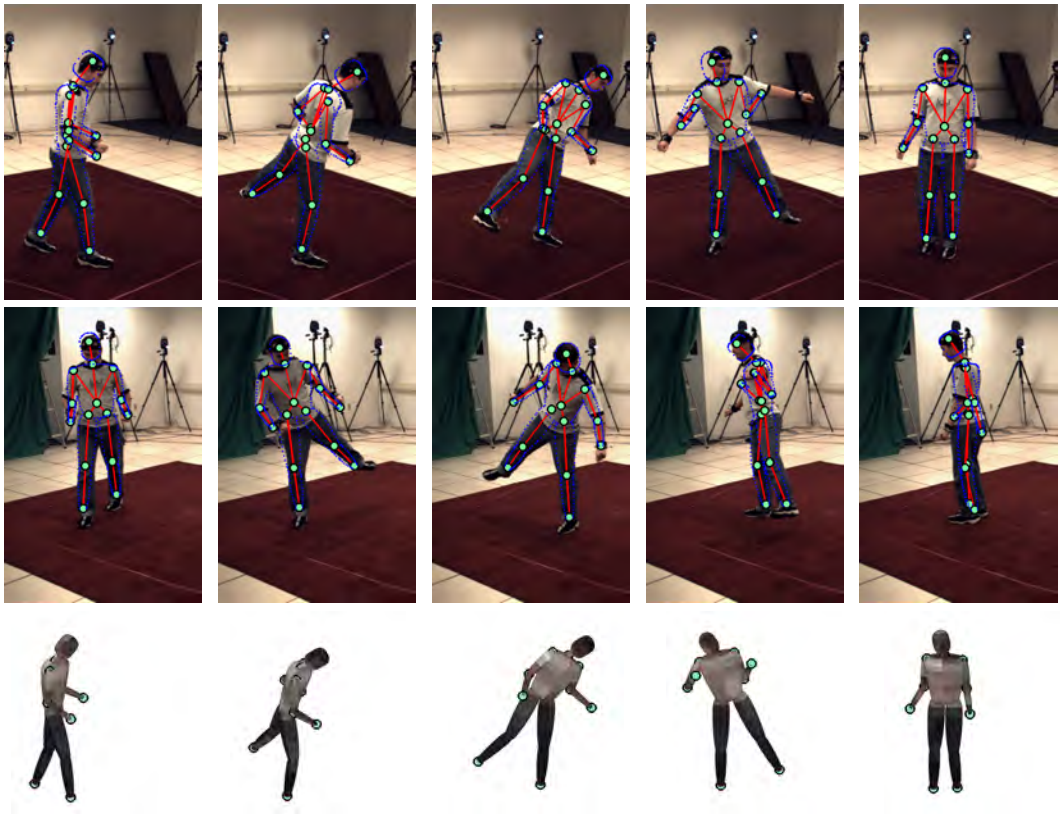


Figure 7.14: SPF tracking results for selected frames of the balancing sequence for S4 — views from cam1, cam2 and 3-D model view from the perspective of cam1, shown from top to bottom

7.3 Summary

Results were obtained by testing the system using the HumanEvaII datasets with subjects S2 and S4. This provided relatively basic balancing action sequences as well as more challenging walking and jogging sequences.

Through all the sequences, an improvement in pose tracking results was seen with the evolution of the optimisation approaches.

The SMD optimisation approach produced the worst results. It coped reasonably well with the balancing sequences, but failed in the walking and jogging sequences with the increase in clutter and occlusion.

SMD H&P helped to overcome some of the problems encountered by taking into account, through a hierarchical approach, the fact that all the parts of the human body model are not completely independent. It also added prediction to help combat occlusion errors. This improved the tracking to a certain extent, with some tracking results able to be obtained in the walking sequences.

The SPF approach combined the multiple hypotheses framework of PF with the better local optimisation of SMD, to achieve the best tracking results, over all the action sequences, for both subjects. Here generally good pose tracking was produced with occasional minor loss of tracking followed by tracking recovery.

A PF approach was also tested, with code provided by Sigal *et al.* [84], to provide a comparison with the SPF optimisation approach. As expected, the SPF method expressed better overall tracking performance than the PF approach.

The results were demonstrated graphically as well as with mean and minimum Euclidean distance pose error results. These error results were statistically analysed to confirm the observed qualitative findings.

Chapter 8

Conclusions and Future Work

8.1 Conclusion

This dissertation presents a system capable of correctly recovering the pose of a human without the use of any special clothing, markers or sensors. Instead, a model-based approach is followed which fits a superellipsoid human model to 2-D edge and 3-D volumetric data. This information is processed from image data obtained from the HumanEvaII datasets. These datasets provided captured images from four surrounding colour cameras, as well as ground-truth motion capture data which allows quantitative analysis of the results.

An illumination invariant silhouette detector forms the basis of the image processing. *A priori* probabilities, colour collinearity testing and darkness compensation are added to help improve the extracted segmentation results. Along with this, a colour-based difference vector edge detector is used to acquire edge maps. The use of colour helps to reduce the clutter by decreasing the detection of shadow edges. The volumetric reconstruction, which provides the important 3-D data, is performed using a voxel-based space carving approach. A precomputed look-up-table helps to increase the speed of the reconstruction while colour is introduced to the results to improve tracking.

A superellipsoid body model is favoured for its simplistic yet good body shape approximation. These superellipsoids are fitted on to a 24 parameter stick figure model which controls the model's pose. Basic hard-limit kinematic constraints are imposed on joint angle movement. The body model has a secondary purpose, to improve the visual display of the output by using a more realistic appearing human model than a plain stick figure.

The objective function used for optimisation consists of a surface alignment and edge alignment part, for the 3-D and 2-D data respectively. For the model's surface points, the closest matching surface voxels are found while the contour points find their

edge point matches using calculated gradient scores. Once the corresponding model and observed points have been obtained, the objective function is calculated by means of a slightly modified squared distance error measure.

The objective function was optimised via four approaches so that a comparison of them could be made. The SMD optimisation gave poor results and was not able to recover from any pose tracking errors encountered. The SMD with Hierarchical and Prediction approach gave slightly better results by helping to overcome some of the problem issues but was still unable to recover from errors.

Results from the PF approach were mostly better than those from the SMD based optimisation approaches. While it was able to recover from some errors, it suffered from a number of limb and torso tracking errors, especially in the jogging action sequence. The SPF approach, which combines the benefits of both the SMD and PF approaches, performed significantly better than the other two approaches throughout the action sequences.

SPF is able to recover from errors through the use of the multiple hypotheses framework it inherits from PF, which guides the optimisation to the region of the global minimum. Combined with this is its use of SMD optimisation, which enables it to handle the high state space dimensionality of full body human pose tracking and refine convergence to the absolute minimum.

The benefits of this SPF approach are shown through numerous results. Both statistical analysis and graphical observations are given to illustrate the improvement, with a number of different action sequences and a couple of subjects being used in the testing.

8.2 Future Work

The tracking system presented is modular, which allows any of the tracking stages to be substituted for an appropriate replacement or improvement fairly easily. It also allows easy addition of processes or more stages if required. As such, this system provides a very **good framework** for continued and future work on model-based markerless human pose tracking.

Presently, manual initialisation of pose is carried out at the start of any test sequence, to roughly position the body model close to the expected pose. For most tracking systems this approximation is required for the optimisation stage to perform adequately. Better automation of the system could be achieved through **automatic pose initialisation**. This could possibly be done through the use of some initial stance, such as with the arms and legs spread wide apart. This stance would provide a clear view of all body parts for a good initialisation.

The HumanEvaII dataset used for testing only contained four colour camera views of the capture scene. This resulted in some clutter and occlusion problems. It is believed that **increasing the number of colour camera** views could help to improve the accuracy of the pose tracking. A study on the optimal number and placement of cameras, to produce the best tracking results, would be very interesting and beneficial to future markerless pose tracking systems.

The current superellipsoid body model provides a reasonable body shape approximation. However, the use of more **complex body models** such as a mesh model (which appears to be becoming more popular) may help to improve model alignment and pose tracking, especially of the torso region. Linked to this is the **improvement of the body model constraints**. Even just the improvement of limbs being restricted from passing through each other could help improve tracking results and prevent some occlusion issues.

This pose tracking system is programmed in Matlab which provides a very easy and user friendly means to prototype systems and test proof-of-concept. However, it is not the fastest language and therefore the speed and efficiency of the system could be improved by implementing most, or all, of the system in **another language such as C or C++**.

Appendix A

Contents of CD

A number of files relating to this dissertation can be found on the attached CD. They are as follows:

- **HPT System - HE2 DataSet** folder:
  This folder contains the Matlab code for the Human Pose Tracking System presented in this dissertation. In the main folder a file, *Readme.pdf*, is provided which gives a brief outline of the code and explains how to run it.
- **Results - mean and min over multiple runs.xlsx**:
  The pose error results for the different optimisation approaches presented in the Results and Discussion chapter. Mean and minimum error values, over the multiple runs performed, are given for the S2 and S4 action sequences.
- **Neil Hendry - MSc Dissertation 2011.pdf**:
  Soft copy of this dissertation.
- **Turnitin Originality Report.pdf**:
  Originality report of the results on this dissertation from the Turnitin anti-plagiarism software. A copy of the dissertation, with highlighted matching sections, together with the matches' primary source and matching percentage is given.
- **Turnitin - reasoning.doc**:
  A document presenting a reasoning for the matching percentage given by the Turnitin anti-plagiarism software.

Bibliography

[1] Bresenham's line algorithm , June 2010. URL `http://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm`.

[2] T. Aach and A. Kaup. Bayesian algorithms for adaptive change detection in image sequences using Markov random fields. *Signal Processing-Image Communication*, 7(2):147–160, 1995.

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

[4] JK Aggarwal and Q. Cai. Human motion analysis: A review. In *IEEE Nonrigid and Articulated Motion Workshop, 1997. Proceedings.*, pages 90–102, 1997.

[5] K. Akita. Image sequence analysis of real world human motion. *Pattern recognition*, 17(1):73–83, 1984.

[6] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Computer VisionECCV'92*, pages 237–252. Springer, 1992.

[7] L. Bo, C. Sminchisescu, A. Kanaujia, and D. Metaxas. Fast algorithms for large scale conditional 3d prediction. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[8] Jean-Yves Bouguet. Camera Calibration Toolbox for Matlab, June 2008. URL `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[9] R. Bowden, TA Mitchell, and M. Sarhadi. Non-linear statistical models for the 3D reconstruction of human pose and motion from monocular image sequences. *Image and Vision Computing*, 18(9):729–737, 2000.

[10] M. Brand. Shadow puppetry. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, 1999.

[11] M. Bray, E. Koller-Meier, P. Muller, N.N. Schraudolph, and L. Van Gool. Stochastic optimisation for high-dimensional tracking in dense range maps. In *Vision, Image and Signal Processing, IEE Proceedings-*, volume 152, pages 501–512. IET, 2005.

[12] M. Bray, E. Koller-Meier, NN Schraudolph, and L. Van Gool. Fast stochastic optimization for articulated structure tracking. *Image and Vision Computing*, 25 (3):352–364, 2007.

[13] M. Bray, E. Koller-Meier, and L. Van Gool. Smart particle filtering for high-dimensional tracking. *Computer Vision and Image Understanding*, 106(1):116–129, 2007.

[14] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 8–15. IEEE, 1997.

[15] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1998. Proceedings*, pages 8–15, 1998.

[16] Q. Cai and JK Aggarwal. Tracking human motion using multiple cameras. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, 1996.

[17] J. Carranza, C. Theobalt, M.A. Magnor, and H.P. Seidel. Free-viewpoint video of human actors. In *ACM SIGGRAPH 2003 Papers*, pages 569–577. ACM, 2003.

[18] Z. Chen and H.J. Lee. Knowledge-guided visual perception of 3-D human gait from a singleimage sequence. *IEEE Transactions on systems, man and cybernetics*, 22(2):336–342, 1992.

[19] GKM Cheung, T. Kanade, J.Y. Bouguet, and M. Holler. A real time system for robust 3D voxel reconstruction of humanmotions. In *IEEE Conference on Computer Vision and Pattern Recognition, 2000. Proceedings*, volume 2, 2000.

[20] 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. 2003.

[21] D. Demirdjian. Enforcing constraints for human body tracking. In *Workshop on Multi-Object Tracking*, volume 418, 2003.

[22] J. Deutscher, A. Davison, and I. Reid. Automatic partitioning of high dimensional search spaces associated with articulated body motion capture. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–669. IEEE, 2001.

[23] J. Duetscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *CVPR*, page 2126. Published by the IEEE Computer Society, 2000.

[24] B.P. Flannery, W.H. Press, S.A. Teukolsky, and W. Vetterling. Numerical recipes in C. *Press Syndicate of the University of Cambridge, New York*, 1992.

[25] J.S. Franco and E. Boyer. Exact polyhedral visual hulls. In *British Machine Vision Conference*, volume 1, pages 329–338. Citeseer, 2003.

[26] P. Fua, A. Gruen, N. D Apuzzo, and R. Plankers. Markerless full body shape and motion capture from video sequences. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(5):256–261, 2002.

[27] J. Gall, J. Potthoff, C. Schnörr, B. Rosenhahn, and H.P. Seidel. Interacting and annealing particle filters: Mathematics and a recipe for applications. *Journal of Mathematical Imaging and Vision*, 28(1):1–18, 2007.

[28] J. Gall, B. Rosenhahn, and H. Seidel. An introduction to interacting simulated annealing. *Computational Imaging and Vision*, 36:319, 2008.

[29] J. Gall, B. Rosenhahn, T. Brox, and H.P. Seidel. Optimization and filtering for human motion capture. *International journal of computer vision*, 87(1):75–92, 2010.

[30] DM Gavrila and LS Davis. 3-D model-based tracking of humans in action: a multi-view approach. In *1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96*, pages 73–80, 1996.

[31] A. Griesser, SD Roeck, A. Neubeck, and LV Gool. GPU-based foreground-background segmentation using an extended colinearity criterion. In *Vision, Modeling, and Visualization (VMV)*, 2005.

[32] Andreas Griesser. Computer Vision Laboratory: Demos - Blue-C , October 2006. URL `http://www.vision.ee.ethz.ch/showroom/bluec/index.en.html`.

[33] M. Gross, S. Wurmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, *et al.* blue-c: a spatially immersive display and 3d video portal for telepresence. *ACM Transactions on Graphics*, 22 (3):819–827, 2003.

[34] R. Gross and J. Shi. The CMU motion of body (mobo) database. 2001.

[35] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *IEEE computer society conference on computer vision and pattern recognition*, pages 1106–1112. Institute of Electrical Engineers Inc (IEEE), 1997.

[36] N. Hendry and B. Naidoo. Video-based pose tracking of a human subject. In *Proceedings of The Military Information and Communications Symposium of South Africa (MICSSA)*, 2009.

[37] N. Hendry and B. Naidoo. Model-based markerless human pose tracking. In *Proceedings of The Military Information and Communications Symposium of South Africa (MICSSA)*, 2011.

[38] 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.

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

[40] NR Howe. Silhouette lookup for automatic pose tracking. In *Computer Vision and Pattern Recognition Workshop, 2004 Conference on*, pages 15–22, 2004.

[41] E. Huber, N.J.S. Center, and TX Houston. 3-D real-time gesture recognition using proximity spaces. In *Applications of Computer Vision, 1996. WACV'96., Proceedings 3rd IEEE Workshop on*, pages 136–141, 1996.

[42] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998.

[43] M. Isard and A. Blake. ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. *Computer Vision - ECCV '98*, page 893, 1998.

[44] R. Jain and K. Wakimoto. Multiple perspective interactive video. In *Multimedia Computing and Systems, 1995., Proceedings of the International Conference on*, pages 202–211, 1995.

[45] V. John, E. Trucco, and S. Ivekovic. Markerless human articulated tracking using hierarchical particle swarm optimisation. *Image and Vision Computing*, 28(11): 1530–1547, 2010.

[46] RE Kahn, MJ Swain, PN Prokopowicz, and RJ Firby. Gesture recognition using the perseus architecture. In *1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96*, pages 734–741, 1996.

[47] L. Kakadiaris and D. Metaxas. Model-based estimation of 3D human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1453–1459, 2000.

[48] R.E. Kalman *et al.* A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[49] R. Kehl. *Markerless motion capture of complex human movements from multiple views.* PhD thesis, Citeseer, 2005.

[50] R. Kehl and L. Van Gool. Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding*, 104(2-3):190–209, 2006.

[51] R. Kehl, M. Bray, and L. Van Gool. Full body tracking from multiple views using stochastic sampling. 2005.

[52] R. Kehl, M. Bray, and L. Van Gool. Markerless full body tracking by integrating multiple cues. In *ICCV Workshop on Modeling People and Human Interaction*, 2005.

[53] P.H. Kelly, A. Katkere, D.Y. Kuramura, S. Moezzi, and S. Chatterjee. An architecture for multiple perspective interactive video. In *Proceedings of the third ACM international conference on Multimedia*, pages 201–212. ACM New York, NY, USA, 1995.

[54] Steffen Knoop, Stefan Vacek, and Rüdiger Dillmann. Fusion of 2d and 3d sensor data for articulated body tracking. *Robot. Auton. Syst.*, 57(3):321–329, 2009. doi: http://dx.doi.org/10.1016/j.robot.2008.10.017.

[55] C.S. Lee and A. Elgammal. Modeling view and posture manifolds for tracking. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[56] H.J. Lee and Z. Chen. Determination of 3d human body postures from a single view. *Computer Vision Graphics and Image Processing*, 30(2):148–168, May 1985.

[57] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math*, 2(2):164–168, 1944.

[58] R. Li, M.H. Yang, S. Sclaroff, and T.P. Tian. Monocular tracking of 3d human motion with a coordinated mixture of factor analyzers. *Computer Vision–ECCV 2006*, pages 137–150, 2006.

[59] Chung-Chin Lin. Model-based human body motion analysis for MPEG IV video encoder. In *ITCC '01: Proceedings of the International Conference on Information Technology: Coding and Computing*, page 435, Washington, DC, USA, 2001. IEEE Computer Society.

[60] S. Liu and G. Trenkler. Hadamard, khatri-rao, kronecker and other matrix products. *Int. J. Inform. Syst. Sci*, 4:160–177, 2008.

[61] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2): 431–441, 1963.

[62] C. Menier, E. Boyer, and B. Raffin. 3d skeleton-based body pose recovery. In *Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization and Transmission, Chapel Hill (USA)*, 2006.

[63] R. Mester, T. Aach, and L. Dumbgen. Illumination-invariant change detection using a statistical colinearity criterion. *Lecture notes in computer science*, pages 170–177, 2001.

[64] B. Michoud, E. Guillou, and S. Bouakaz. Real-time and 3d human motion capture using multiple views. In *HUMO07*, pages 88–103, 2007.

[65] AR Micilotta, E.J. Ong, and R. Bowden. Real-time upper body detection and 3D pose estimation in monoscopic images. *Lecture Notes in Computer Science*, 3953:139, 2006.

[66] I. Mikić, M. Trivedi, E. Hunter, and P. Cosman. Articulated body posture estimation from multi-camera voxel data. In *IEEE Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii*, 2001.

[67] I. Mikić, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3): 199–223, 2003.

[68] G. Mori and J. Malik. Recovering 3d human body configurations using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28 (7):1052–1062, 2006.

[69] E. Nadernejad, S. Sharifzadeh, and H. Hassanpour. Edge Detection Techniques: Evaluations and Comparisons. *Applied Mathematical Sciences*, 2(31):1507–1520, 2008.

[70] H. Ning, W. Xu, Y. Gong, and T. Huang. Discriminative learning of visual words for 3d human pose estimation. 2008.

[71] J. O'Rourke and NI Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):522–536, 1980.

[72] R. Plankers, P. Fua, and N. D'apuzzo. Automated body modeling from video sequences. In *IEEE International Workshop on Modelling People, 1999. Proceedings*, pages 45–52, 1999.

[73] R. Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1-2):4–18, 2007.

[74] J.M. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 612–617. IEEE, 1995.

[75] J.M. Rehg, D.D. Morris, and T. Kanade. Ambiguities in visual tracking of articulated objects using two-and three-dimensional models. *The International Journal of Robotics Research*, 22(6):393, 2003.

[76] G. Rogez, J. Rihan, S. Ramalingam, C. Orrite, and P.H.S. Torr. Randomized trees for human pose detection. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[77] K. Rohr. Incremental recognition of pedestrians from image sequences. In *1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93.*, pages 8–13, 1993.

[78] R. Rosales and S. Sclaroff. Inferring body pose without tracking body parts. In *IEEE Conference on Computer Vision and Pattern Recognition, 2000. Proceedings*, volume 2, 2000.

[79] Rhonda Rose. Anthropometry and Biomechanics, July 2008. URL `http://msis.jsc.nasa.gov/sections/section03.htm`.

[80] K. Sato, T. Maeda, H. Kato, and S. Inokuchi. CAD-based object tracking with distributed monocular camera for security monitoring. In *Proceedings of the 1994 Second CAD-Based Vision Workshop, February 8-11, 1994, Champion, Pennsylvania*, page 291. IEEE Computer Society, 1994.

[81] S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[82] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Ninth IEEE International Conference on Computer Vision, 2003. Proceedings*, pages 750–757, 2003.

[83] L. Sigal and M.J. Black. Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. *Brown Univertsity TR*, 120, 2006.

[84] L. Sigal, A.O. Balan, and M.J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1):4–27, 2010.

[85] W. Silverman Bernard. Density Estimation for Statistics and Data Analysis, 1986.

[86] C. Sminchisescu. Consistency and coupling in human model likelihoods. In *IEEE International conference on automatic face and gesture recognition*, pages 27–32. Citeseer, 2002.

[87] C. Sminchisescu and B. Triggs. Covariance scaled sampling for monocular 3d body tracking. 2001.

[88] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *The International Journal of Robotics Research*, 22(6): 371, 2003.

[89] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Discriminative density propagation for 3d human motion estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 1, 2005.

[90] A. Sundaresan. *Towards markerless motion capture: model estimation, initialization and tracking.* PhD thesis, University of Maryland, College Park, 2008.

[91] A. Sundaresan and R. Chellappa. Multicamera tracking of articulated human motion using shape and motion cues. *Image Processing, IEEE Transactions on*, 18(9):2114–2126, 2009.

[92] Tomas Svoboda. Multi-Camera Self-Calibration, July 2009. URL `http://cmp.felk.cvut.cz/~svoboda/SelfCal/index.html`.

[93] L. Taycher, G. Shakhnarovich, D. Demirdjian, T. Darrell, Massachusetts Inst. of Tech. Cambridge Computer Science, and Artificial Intelligence Lab. *Conditional random people: Tracking humans with CRFS and grid filters.* Defense Technical Information Center, 2005.

[94] C. Theobalt, M. Magnor, P. Schuler, and H.P. Seidel. Combining 2d feature tracking and volume reconstruction for online video-based human motion capture. In *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on*, pages 96–103. IEEE, 2002.

[95] C. Theobalt, J. Carranza, M.A. Magnor, and H.P. Seidel. Enhancing silhouette-based human motion capture with 3D motion fields. 2003.

[96] R. Urtasun and T. Darrell. Sparse probabilistic regression for activity-independent human pose inference. 2008.

[97] M. Van den Bergh, R. Kehl, E. Koller-Meier, and L. Van Gool. Real-time 3D body pose estimation. In *Multi-Camera Networks: Concepts and Applications*. Elsevier, 2009. in press.

[98] M. Vondrak, L. Sigal, and O.C. Jenkins. Physical simulation for probabilistic motion tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[99] S. Wachter and H.H. Nagel. Tracking persons in monocular image sequences. *Computer Vision and Image Understanding*, 74(3):174–192, 1999.

[100] P. Wang and J.M. Rehg. A modular approach to the analysis and evaluation of particle filters for figure tracking. 2006.

[101] S. Wesolkowski and E. Jernigan. Color edge detection in RGB using jointly euclidean distance and vector angle. In *Proceedings of IAPR Vision Interface 1999*, pages 9–16, 1999.

[102] CKI Williams and CE Rasmussen. Gaussian processes for regression. *Advances in Neural Information Processing Systems*, (8), 1996.

[103] G. Xu and Z. Zhang. *Epipolar geometry in stereo, motion, and object recognition: a unified approach*, volume 6. Springer, 1996.

[104] X. Xu and B. Li. Learning motion correlation for tracking articulated human body with a rao-blackwellised particle filter. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[105] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision*, volume 1, pages 666–673, 1999.