THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Techniques for Binocular Markerless Visual Tracking of 3D Articulated Bodies

Andrew W. B. Smith

B. Eng. (Elec)

B. Bus. Man. (Economics)

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in 2016

School of Information Technology and Electrical Engineering

Abstract

This thesis advances methods for performing markerless visual tracking of articulated bodies using one or two cameras. The research presented aims to improve upon existing Bayesian inspired tracking methods, by examining the 'building blocks' of these tracking algorithms, in particular the measurement function design, the state space selection, and local optimization methods. Results presented in this thesis show that improvements can be made in all of these areas. These improvements are applicable to a variety of Bayesian tracking algorithms.

This thesis begins by examining literature relevant to the visual tracking problem. This includes the measurement functions used by other authors, focussing on the edge detection methods used in both tracking and segmentation problems. A general overview of the global search problem is given next, as a global search is a fundamental part of a Bayesian tracking algorithm. The combination of Newton like local optimization methods and the measurement functions used in visual tracking problems is examined next, and it is shown that Newton optimizers are not ideally suited to these measurement functions. The Bayesian tracking framework is then detailed, along with a review of several existing Bayesian tracking algorithms. Finally some non Bayesian tracking algorithms are discussed.

Following the literature review, details of the models used in the experiments presented in this thesis are given. These include the articulated human body model, the camera model, image gradient metrics, self occlusion treatment, and a generic colour based region measurement method.

The use of graph based approaches for edge measurements is then investigated. Graph based methods are commonly used in image segmentation problems, however have not been applied to visual tracking problems. A novel method for performing edge measurements using the 'shortest path' around the object's occluding contour is presented. Unlike in the segmentation problem, self occlusion models mean the weights or costs of some graph vertices can not be determined. Different treatments for occluded graph vertices are given and evaluated. It is shown that the graph based approach produces observational likelihoods that are more accurate and have significantly fewer local maxima than the edge measurement schemes previously used in tracking problems. While this approach is computationally more expensive than other methods, it is argued that this is offset by the reduced computational expense of the global search procedure used in tracking algorithms.

The choice of state space used in the tracking problems is examined next. While most authors have used a state space based on the joint angles of the human body, a Cartesian state space based on the world coordinates of limbs is proposed. While Cartesian based state spaces have been used by

other authors for representations of kinematic models, to the author's knowledge they have not been used for full kinematic models. It is shown that that the more linear relationship between state variables in the Cartesian space and the 3D locations of sampled points on the object improves dynamic model predictions and principal component analysis. It is also shown that the Cartesian formulation also increases the linearity between state variables and the image coordinates of sampled points on the object. This in turn improves the performance of local optimization methods which make localized quadratic approximations to the measurement function. While the Cartesian based space has a higher dimensionality than the rotation based space, the geometrically plausible region of the Cartesian space has the same content (area) as the rotation space, which negates the well known 'curse of dimensionality'. A simple method is given to project an implausible Cartesian state to a geometrically plausible state, as well as a method to dampen the measurement function curvature in these implausible directions.

Following this, a novel local optimization method is proposed. This optimization method is specific to visual tracking problems, and uses the camera geometry to infer interesting search directions. Treatments for choosing these search directions are given for both the monocular and two camera cases. A problem decomposition is also used to reduce the computational cost of the optimizer. This method is shown to outperform Newton based optimizations in a rotation based state space, and gives at worst equivalent results to a Newton based approach in a Cartesian state space, but at a significantly reduced computational cost.

Finally, tracking results are presented for a difficult image sequence using the combined ideas presented in this thesis. This sequence is a golfer performing a golf swing, which is a highly dynamic motion with large object velocities and accelerations.

Declaration by author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my research higher degree candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the policy and procedures of The University of Queensland, the thesis be made available for research and study in accordance with the Copyright Act 1968 unless a period of embargo has been approved by the Dean of the Graduate School.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis.

Publications during candidature

Peer reviewed publications:

- A. W. B. Smith and B. C. Lovell, "Visual Tracking for Sports Applications," in *APRS Workshop on Digital Image Computing*, B. C. In Lovell and A. J. Maeder, Eds., vol. 1(1), Brisbane, February 2005, pp. 79–84.
  Incorporated into Chapter 1 defining the scope and the aims of this thesis.

- A. W. B. Smith and B. C. Lovell, "Measurement Function Design for Visual Tracking Applications," in *Proceeding of the Eighteenth International Conference on Pattern Recognition*, vol. 1, 2006, pp. 789–792.
  Incorporated into Section 4.3 examining the construction of the 'cost trellis', as well as the quantitative evaluation of the performance of various edge measurement schemes presented in Section 4.8.5.

- A. W. B. Smith, "Self Occlusions and Graph Based Edge Measurement Schemes for Visual Tracking Applications," Awarded best student paper at *Digital Imaging Computing: Techniques and Applications*, Dec 2009.
  Incorporated into Section 4.4 examining the construction of the 'occluding contour cost trellis', as well as the quantitative evaluation of the performance of various edge measurement schemes presented in Section 4.8.

- A. W. B. Smith and B. C. Lovell, "Autonomous Sports Training from Visual Cues," in *Proceedings of the Eighth Australian and New Zealand Conference on Intelligent Information Systems*, B. C. In Lovell, D. A. Campbell, C. B. Fookes, and A. J. Maeder, Eds., vol. 1(1), Sydney, December 2003, pp. 279–284.

- X. Zhang, H. Wang, A. W. B. Smith, and B. C. Lovell, "Corner Detection Based on Gradient Correlation Matrices of Planar Curves," in *Pattern Recognition*, vol. 43(4), 2010, pp 1207-1223.

Publications included in this thesis

No publications included.

Contributions by others to the thesis

No contributions by others.

Statement of parts of the thesis submitted to qualify for the award of another degree

None.

Acknowledgements

There are many people who provided me with much support over the course of my dissertation. Firstly, I would like to thank Rick Baker for his vision of creating autonomous sports training technologies, towards which this dissertation was undertaken. I would also like to thank him for providing the equipment necessary for the completion of this dissertation through his company Hi-Tech Video. Furthermore I would like to thank him for keeping my motivation levels high by reminding me of the many potential uses my research could have.

I would like to thank my supervisor Professor Brian Lovell, for his teaching of the undergraduate course in signals and image processing at the University of Queensland (UQ), which sparked my interest in the field of computer vision. I would also like to thank him for introducing me to Rick Baker, which led to me undertaking this research. Lastly I would like to thank him for accepting me as his postgraduate student. I would also like my colleagues and co-workers at the School of Information Technology and Electrical Engineering at the University of Queensland. Thanks to the IRIS research group, Simon, Stefan, Carlos, Ben, Emanual, Christian and Andrew Mehnert, for your friendship and technical suggestions. Thanks also to Wayne Wilson for providing a productive and friendly working environment during my employment in my various roles working for you.

A special thanks also goes to Andrew Bradley for motivating me to tackle my thesis corrections, without which this thesis would not have been completed.

This research could not have been performed without financial support from several sources. I would like to thank Rick Baker and Hi-Tech Video, and the School of Information Technology and Electrical Engineering at the University of Queensland, for funding my scholarship without which I would not have completed this dissertation. I would also like to thank the Queensland state government for paying for my university tuition fees during the course of my candidature.

Keywords

Human motion estimation, human body tracking, visual tracking, measurement function design, state space selection, optimization, camera geometry, generative models, particle filtering

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code: 080104, Computer Vision, 100%.

Fields of Research (FoR) Classification

FoR code: 0801, Artficial Intelligence and image Processing, 100%.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Visual tracking of human movement has attracted much attention due to the wide variety of applications which could be performed automatically; however currently require human interaction and interpretation. These applications include sports training, rehabilitation, and security.

The goal of visual tracking is to estimate information about an object from an image sequence. Typically the data types to be inferred are: translational coordinates, articulated joint angles and deformation parameters. The human visual system provides empirical proof that accurate tracking of moving, deforming objects is an information processing problem with a robust real-time solution [64]. A full understanding of this human solution is still well beyond our grasp however.

1.1 Motivation

Biometric analysis has already established itself as an effective training tool for athletes, although most techniques rely on the use of retro-reflective markers or magnetic sensors to be placed on an athlete before such analysis can be performed. Replacing retro-reflective markers and magnetic sensors with visual tracking information reduces the setup time required to perform an analysis, allowing a larger volume of analysis to be performed at a reduced cost.

Advances in computer vision have lead to the development of systems that use visual information to provide sports training. Examples of these are Virtual Pat [30] and Pfinder [9] and the Sony EyeToy® [1]. These predominately use background subtraction techniques [35][55] to infer an object's position. This approach is not applicable when objects of interest occlude other objects of interest in the image plane, for example when a person's arms are in front of their body. Full body human trackers are the solution to this problem.

A full body human tracker can be used in a system which uses visual cues to obtain an athlete's postural information, and analyses this with respect to a learned ideal motion. Automated feedback can then be given based on differences between the athlete's motion and a technically correct motion.

1.2 Aim

The goal of this thesis is the advancement of full body visual tracking techniques capable of robustly estimating human posture in "non-laboratory" settings - settings where rigourous camera calibration and lighting control is not possible, and less than two cameras are available. In Section 2.4.3 numerous tracking algorithms are presented which search a representation of a probability distribution, looking for a global maximum and other high value local modes.

Whilst creating new global search strategies presents an opportunity for the presentation of elegant theory, this thesis aims to improve current global search strategies by examining and improving their 'building blocks'. The No Free Lunch Theorems for search and optimization discussed in Section 2.2 explains that, roughly speaking, if a search algorithm outperforms another search algorithm for a particular problem, then it will have worse performance over all other problems. Following this, this thesis follows the methodology used most predominantly throughout the works of Sminechisecu and Triggs [86, 87, 89–93], whereby the study of the properties / topology of the probability distributions to be searched is of critical importance, and the design of search / optimization techniques based around these properties. A good example of this philosophy is Isard and Blakes CONDENSATION [48] algorithm, which proved very successful and brought much attention to this field. The authors recognised the multi-modal nature of the observational likelihoods encountered in visual tracking, and specifically designed the algorithm around that consideration.

The author believes the general literature has a tendency to design new search / optimization methods without enough consideration to the specific nature of the probability distribution being searched. Included in this thesis is, to the author's knowledge, the first analytic study of the properties of the component parts of the measurement functions that underpin the encountered observational likelihoods, and the first attempt to redesign these components with the goal of creating likelihoods

that are most readily searched by existing methods. The techniques developed in this thesis are intended to fit into the framework of generative trackers, and so be useful extensions of existing trackers.

Another benefit of looking at the building blocks of the global search strategies is that it allows the problem to be viewed in a more modular way. The success and failure of a global search strategy can be broken down and the performance of each component part can be evaluated. This would allow more explanation of why one method outperforms another and create a basis for changing a component of one method with a better performing component of a different method.

This thesis aims to improve the performance of existing tracking algorithms in three ways. Firstly, an edge measurement technique is developed to improve the accuracy and reduce the number of local maxima in the observational likelihood searched using global optimization techniques. Reducing the number of local maxima increases the performance of all of the Bayesian tracking algorithms discussed in this thesis. Consistently generating observational likelihoods with similar properties also allows an informed decision of which tracking algorithm will be the most successful.

Secondly, the state space in which the tracking problem is formulated is examined. If the state space is formulated such that model states which are 'close' in this space have similar appearances in the image, then it is expected that these 'close' states should have similar measurement scores. Similar measurement scores in local regions of the state space helps improve the conditioning of the observational likelihood – while the number of local maxima is not necessarily reduced, each mode has a simpler geometric interpretation, *i.e.* , the modes have more spherical or elliptical shapes rather than long highly curved ridges ('tails'). It is then inherently easier to generate 'interesting' new states, which will have a similar image appearance to another previously calculated state.

Finally, this thesis aims to improve the performance of local optimizers commonly used as component of a global search strategy in visual tracking problems. The properties of Newton like local optimizers with respect to the measurement functions used in visual tracking algorithms are discussed, and it is shown that Newton like local optimizers are not necessarily ideal for the visual tracking problem. As such, the development of a novel local optimization scheme specifically tailored to the visual tracking problem is an aim of this thesis.

Following the foregoing discussion on the increased attention given to the search strategy itself, the work presented in this thesis is evaluated independently in each Chapter before the performance of the overall system is evaluated. It is hoped that the work presented in Chapters 4 and 5 will be viewed as being just as important a contribution as the new search strategy present in Chapter 6.

1.3 Scope

This thesis investigates human motion tracking methods that capture 3D postural information using tracking methodologies that generate hypothesis object states, using one or two cameras. The aim of this thesis is to improve the 'building blocks' of trackers that fit the above category. The image sequences analysed in this thesis are of people performing golf swings, where golf has been chosen as the sport of focus because of the golfer's limited translational motion but large rotational motions, so static cameras can capture a complex dynamic motion. The estimation of the golfer's body shape parameters is beyond the scope of this thesis, and so a known human body model and a known position in the first image of each sequence is assumed. The images sequences have been collected using two cameras, however monocular results are presented using the data from only one of these cameras.

No knowledge of the motion to be tracked is assumed, other than the position in the first frame. No detailed observational model parameters, such as colour templates, are assumed known by any proposed methods. It is assumed however that regions of the human, *e.g.* shirt, pants, *etc*, are of a uniform colour. Learned observational model parameters may be used by existing methods in comparisons with proposed methods. Throughout this thesis it is assumed the tracker will be deployed in an environment with significant background motion, rendering background subtraction techniques unreliable.

1.4 Evaluation of Results

(a) Data Set 1 Frame 1 (b) Data Set 2 Frame 1

FIGURE 1.1: Image Sequences

Ground truth data was not available for the image sequences analysed in this thesis. As recovering joint angles from images is fundamentally an estimation problem given the large image noise, results

of proposed methods are compared with another estimator. Since the human vision system is the best available estimator, where applicable results of proposed methods are compared with a data set created by hand labelling the image sequences. Two hand labelled data sets are used extensively throughout this thesis, with the first frame of each shown in Figure 1.1. Data set one is of a retired professional golfer, and has poor object–background contrast for the lower half of the golfer. Data set two is of an amateur golfer (the author) and has poor contrast between the upper body despite being an attempt to control the background. Two views are available for both data sets.

1.5 Organization of this Thesis

This thesis explores various aspects of the visual tracking problems. Chapter 1 gives an overview of this thesis. Chapter 2 reviews the previous research and related concepts: measurement functions, search and optimization theory, and Bayesian and non–Bayesian tracking algorithms. Chapter 3 gives details of the various models used throughout this thesis. Chapter 4 and 5 examine methods to improve the topology of the observational likelihood, facilitating the global search process. In Chapter 4, a graph based edge measurement approach is investigated, while Chapter 5 examines the choice of state space in which the tracking problem is formulated. In Chapter 6, a novel local optimization scheme is presented, which uses camera geometry to choose interesting search directions. In Chapter 7, tracking results for the combined research presented in Chapters 4, 5, and 6 are shown. A more detailed description of each Chapter is given below:

- Chapter 2 examines literature relevant to the visual tracking problem. The first section reviews the various types of measurements which have been used by other authors to test the match between an image and a hypothesized model state. These are split into two groups – edge based measurements, and region based measurements. Edge based measurements implicitly examine image derivatives, while region based measurements examine spatial patterns in the image. The properties of these measurement methods discussed include their accuracy, and their zone of attraction. Accuracy is defined as the agreement between the location of the maxima with its perceptual location, while the zone of attraction is the region where there is a monotonically increasing path to a specific probability maxima. The role of graph based approaches to image segmentation problems is also discussed.

  The next section of Chapter 2 examines global search and optimizations problems in general. It is discussed that the global search of multi-modal surfaces is in general *NP hard*, and that measurement functions used in visual tracking are multi-modal when applied to real image

data. The desirable properties of a measurement function are discussed in this context. The properties of Newton like local optimizers with respect to the measurement functions used in visual tracking problems is discussed, and it is shown that the derivatives of these measurement functions are not always available. As such, Newton based methods are not necessarily ideal for performing local optimizations in visual tracking algorithms.

The final section of Chapter 2 examines the visual tracking algorithms proposed by other authors. These algorithms are divided into two groups – Bayesian (generative), and discriminative. Bayesian tracking algorithms propagate a belief in the object's state described by a probability distribution, whereas discriminative methods seek a deterministic representation of the object's state. Due to the multi-modal nature of the likelihoods encountered, Bayesian methods are generally better suited to the visually tracking problem. Numerous Bayesian tracking algorithms are discussed as they are the focus of this thesis, as well as serval non-Bayesian methods.

- Chapter 3 gives details of the models used throughout this thesis. The human body model is discussed, including a description of each of the 43 degrees of freedom used to form the rotational state space. The link surface model used for each link in the kinematic chain is then discussed, including details of the calculation of each link's occluding contours in an image. Following this is a description of the self occlusion model, which is a component of each measurement function used in this thesis, as well as details of self occlusion model's calculation. Camera calibration is then detailed, including the calculation of the radial distortion model and correction, as well as the calculation of the camera projection matrices. The image gradient calculation scheme, which is a fusion of the intensity and colour gradients is discussed next. Finally, a general region based measurement is proposed, whose underlying assumption is that different body parts are of a uniform colour. The measurement function used in the optimization experiments performed in this thesis is a fusion of this region measurement and an edge measurement.

- In Chapter 4, a novel edge measurement scheme is proposed which utilizes a graph based approach to edge measurement. Graph based methods are commonly used in image segmentation problems, however to the author's knowledge have not been used in visual tracking problems. A formulation is given to construct a graph around the hypothesized edge locations of each link used to model the human. The edge score is a function of the energy of the shortest path traversing each trellis. This method is then extended to the creation of graphs representing the object's entire occluding contour. Unlike in segmentation problems, due to self occlusions the

shortest path must be calculated when the costs of some graph vertices are unknown. Methods for treating a graph's occluded vertices are proposed and evaluated. Algorithms for the computation of the shortest path for each treatment are also given.

It is shown that the proposed graph based approach produces observational likelihoods with significantly fewer modes than the edge measurement schemes previously used in tracking problems. While this approach is computationally more expensive, it is argued that this is offset by the reduced computational cost of the global search procedure. A technique to calculate an approximated Hessian for the proposed edge measurement scheme is derived, and a method to use restrict potential paths to model link surface deformation is also discussed.

- Chapter 5 examines the choice of state space in which the tracking problem is formulated. Most authors have used a state space based on the joint angles of the human body. An alternative state space is proposed, where the position of each link is described by the Cartesian coordinates of the end of each articulated limb. While similar state spaces have been used for representations of full body kinematic chains, to the author's knowledge this Cartesian based formulation has not been used in conjunction with a full kinematic model.

  This proposed state space is motivated by the desire to formulate the tracking problem in a space where 'close' states have similar appearances in the image, which in turn means that 'close' states will have similar measurement scores. While reformulating the tracking problem in such a space does not necessarily reduce the number of local maxima in the observational likelihood, each mode can be given a simpler geometric interpretation, *i.e.* , the modes have more spherical or elliptical shapes rather than long highly curved ridges ('tails'). It is then easier to generate other states which will have a similar appearance and measurement score to an existing state. In the commonly used rotation based space, the world location of a point on the object is a highly non-linear function of the state variables. In the proposed Cartesian space, these points are a more linear function of the state variables. It is shown that this increased linearity improves dynamic predictions, as well as principal component analysis, and local optimization.

  This Cartesian based space has a higher dimensionality than the rotation based space, and perhaps has been avoided due to the well known 'curse of dimensionality'. The region of the state space where the object is in a geometrically consistent state (limb lengths are preserved) has the same content as the original rotation space however. A simple method is given to project an implausible state onto the constraint surface, as well as a method to inflate measurement function curvature in directions orthogonal to the constraint surface.

- In Chapter 6, a novel local optimization method is proposed. Local optimizations are often used as a component of global search strategies in visual tracking problems. Earlier in this thesis it was discussed that optimizers should be tailored to specific problems, and that the measurement functions used in visual tracking problems are not ideally suited to a Newton optimizer, partly because their derivatives are not defined for all (interesting) points in the state space.

  The proposed local optimizer is specific to visual tracking problems, and uses the camera geometry to infer interesting search directions rather than by inferring a search direction implicitly from the noisy image data. Methods to calculate these search directions are given for both monocular and the two camera tracking problems. These search directions were motivated by the author's method for hand selecting trial (update) states when presented with undesirable optimization results. In the monocular case, the search directions of interest are chosen as the ambiguous depth direction and an 'image displacement maximizing direction', while in the two camera case search directions were chosen to only alter the object's appearance in one image. The proposed method also uses a problem decomposition to reduce the computational cost of the optimizer.

  The proposed method is shown to outperform Newton based optimizations in a rotation based state space, and gives at worst equivalent results to a Newton optimizer in a Cartesian state space, but at a significantly reduced computational cost. Extensions are given to allow the optimizer to escape local modes cause by mismatched edge assignments, and kinematic forwards–backwards ambiguities.

- In Chapter 7, the modifications proposed in Chapters 4 – 6 are integrated into a covariance scaled sampling style multiple hypotheses tracker. Successful tracking results are shown for challenging image sequences, in both the monocular and two camera tracking cases.

1.6 Original Contributions

A key component of the visual tracking algorithms considered in this thesis is searching the observational likelihood (or representation thereof) for large local modes. This can be considered as an optimization process over a hyper surface representing the observational likelihood. The novelty of this thesis comprises of two parts, the first aimed at improving the conditioning (smoothness) of this surface, thereby making it easier to search, while the second part relates to the technique used to search local areas of this surface.

1.6.1 Measurement Function Design

Many authors perform measurements on the object's expected edge locations, often by casting search lines at normals to the expected edge and finding edge features along them. While this method is fast, there are two major shortcomings to it. The first is that a small change in the state vector can cause a large change in the set of pixels used for measurement. The second is that this method is dependant upon reliably finding a set edge features for each image, which is extremely difficult where the object does not contrast well with the background. These problems reduce the smoothness of the observational likelihood, and hence increase the number of local modes.

To overcome these problems, a graph based edge measurement scheme is proposed in Chapter 3. This method is based upon finding the shortest path across a trellis constructed around the object's occluding contour. While these approaches are commonly used in image segmentation problems, to the authors knowledge they have not been used for tracking problems. Unlike in segmentation problems, measurement functions must model self occlusions. As such, several treatments for occluded trellis vertices are considered, along with methods to compute the shortest path for each treatment. It is shown that this graph based approach significantly reduces the number of modes in the observational likelihood. The worst performance in the two dimensional experiments was a 95% reduction in the number of modes. Jacobian and Hessian approximations are also given for the proposed method.

1.6.2 State Space Selection

Typically the state space chosen for human body tracking is based upon the joint angles of the human model. The relationship between these joint angles and the location of the person in the image is highly non-linear, and so two quite different state vectors can result in two similar occluding contours in an image. This can increase the difficulty of searching the observational likelihood, as distant points in the state space have similar probabilities because they have similar projections.

A state space composed of Cartesian coordinates of rotating kinematic chain links (limbs) is proposed as an alternative state space in Chapter 4. While Cartesian based state spaces have been used in conjunction with representations of full body kinematic models, to the authors knowledge they have not been used with a 'pure' kinematic models. In the Cartesian space, the function describing the world coordinates of a point on the model is a much more linear function of the state variables, which is then shown to improved the performance of dynamic predictions and principal component analyses. Furthermore, the location of a point on the model in the image is also a more linear function of the state variables. It is shown that this results in improved local optimization performance in the Cartesian based space, as local quadratic approximations to the measurement function hold in larger

regions.

Using a Cartesian based state space increases the dimensionality of the tracking problem, which has negative connotations given the well known 'curse of dimensionality' in tracking problems. The region of the Cartesian space which satisfies geometric (limb length) constraints has equivalent content to the rotation based space however. A simple method is given to project a geometrically inconsistent state onto the constraint surface, as well as a measurement function curvature dampening scheme, which is used to restrict variability in directions orthogonal to the constraint surface.

1.6.3 Local Optimization Method

A novel local optimization scheme is presented in Chapter 6. Many tracking algorithms used by other authors utilize a Newton like local search process. While these are good general methods, it is shown that measurement function used in visual tracking have undesirable properties for Newton like methods, particularly that the 1^{st} and 2^{nd} order partial derivatives are not always defined in the region of interest.

The proposed local optimizer is specifically tailored to visual tracking problems, and uses camera geometry to infer interesting search directions. In the monocular case, these search directions are aimed at isolating the ambiguous depth direction, while in the two camera case, the search directions are chosen to perturb the object's projection in only a single view. These search directions were motivated by the author's method for hand selecting trial points when presented with undesirable optimizer results. It is shown that the proposed method significantly outperforms a Newton optimizer in a rotation state space, and at worst performs equivalently to a Newton optimizer in a Cartesian state space, however at a greatly reduced computational cost. Extensions for escaping local modes commonly encountered by an optimizer, *e.g.* mismatched edge assignments are also given.

2

Literature Review

This chapter reviews concepts related to the visual tracking problem, and previous research by other authors. These concepts are divide into three sections. The first section reviews various measurement methods used by different authors to evaluate the observational likelihood of a hypothesized object state. Of particular interest are the properties of the observational likelihoods that can be expected from each measurement function.

The second section reviews search and optimization theory. It begins with a general discussion on the global search and optimization problem, with reference to the expected topology of the observational likelihoods that can be expected using the measurement functions described in the first section. Newton like local optimization techniques are examined, as they are often used as part of a global optimization strategy in visual tracking algorithms. The suitability of Newton like methods for the measurement functions used in visual tracking is discussed.

The third section reviews tracking strategies. These are categorized into two groups, Bayesian (*generative*) and discriminative tracking algorithms. Bayesian tracking algorithms describe the object's state at each time step as a probability distribution, whereas discriminative algorithms attempt to find an exact deterministic solution to the object's state. Bayesian algorithms are generally better

suited to the tracking problem because of image data's high noise levels, and so tracking is fundamentally an estimation problem; while discriminative algorithms have the advantage of generally being computationally cheaper.

2.1 Measurement Methods

This section discusses the various measurement methods used by different authors to derive a function to test the match between the image and a hypothesized object configuration. These are sorted into two categories, edge measurements and region measurements. Edge measurements utilize contrasts between the object of interest and the background, forming discontinuities in the image at the location of the object's edges. The role of image derivatives in finding these discontinuities is briefly discussed, as well as the use of graph based methods to find potential object edges. Region measurements utilize the notion that the object of interest can be broken down into homogenous regions, or more generally, that regions within object's boundary will exhibit some spatial correlation.

Three properties of the different measurement methods are examined. The first is the accuracy of the method. This is taken to mean the distance in the state space between the true object position and the nearest local maxima in the observational likelihood. The second is the size of the *zone of attraction*. The zone of attraction is the region where the observational likelihood monotonically increases toward the relevant local maxima, and so there is an expectation that a local optimizer seeded within the zone of attraction will converge to the relevant maxima. The final property discussed is the computational cost of each method.

2.1.1 Image Derivatives and Edge Detection

The edge of an object can usually be defined by discontinuities in an image, such that local image features are constant within an object and discontinuous across the object's boundaries. The detection of these discontinuities is difficult in practice due to the problem of choosing suitable discriminating features, which is made more difficult by the presence of noise and background clutter.

A common approach to detecting object boundaries is by direct classification of local features.

The Sobel filter [42] is an example of this:

$$D_x = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$D_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The discrete image gradient is then given by:

$$|\nabla I(x,y)|^2 = (D_x \star I(x,y))^2 + (D_y \star I(x,y))^2$$

where $I(x,y)$ is the image and $\star$ denotes convolution. Edges can then be chosen as points where $|\nabla I|^2$ exceeds a threshold. Such filters are computationally inexpensive due to their compact support, however have the disadvantage of rarely producing closed contours even on well defined boundaries.

Another edge filter, the Laplacian of Gaussian filter proposed by Marr and Hildreth [68], is guaranteed to produce closed contours. Marr and Hildreth proposed that edge detectors should be well localized in the frequency domain, which reduces the range of scales at which edges should be detected. They also proposed that these detectors should be spatially localized because image formation may be modelled locally, and because biological edge detectors are necessarily compact. Marr and Hildreth also assumed that that the spread of the filter in the spatial and frequency domains should be measured as the variance of the filter's impulse response. These cannot be simultaneously made small due to the uncertainty principle, with the minimum spatial frequency bandwidth product necessarily greater than or equal to $\frac{1}{4}\pi$. Leipnik [78] observed that 'optimal filters', with minimum spatial frequency bandwidth products, are Gaussian. Edges then occur where the magnitude of the first derivative is large, which corresponds to a minimum in the second derivative.

The unique rotationally invariant second order derivative operator is the Laplacian, and so Marr and Hildreth proposed that edges should coincide with zero crossing of a Laplacian of Gaussian (LOG) filter. While presented as an improvement over simpler methods such as proposed by Sobel and Canny, the LOG filter suffers from an inability to treat points where three objects meet. Despite the LOG filters desirable theoretically properties, it often produces unusable results in practice.

2.1.2 Edge Based Measurement Functions

Edge based measurement methods are perhaps the simplest measurement method available, and as such it is rare that an edge measurement method cannot be used. Edge based measurements typically

have good accuracy, so a maximal point can be expected in the measurement function at the true model state, however they usually have a small zone of attraction so modes in the measurement function tend to cluster.

Line Searches

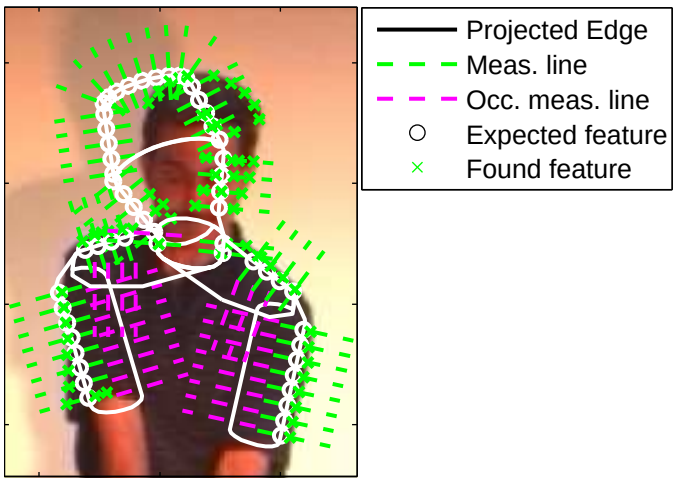
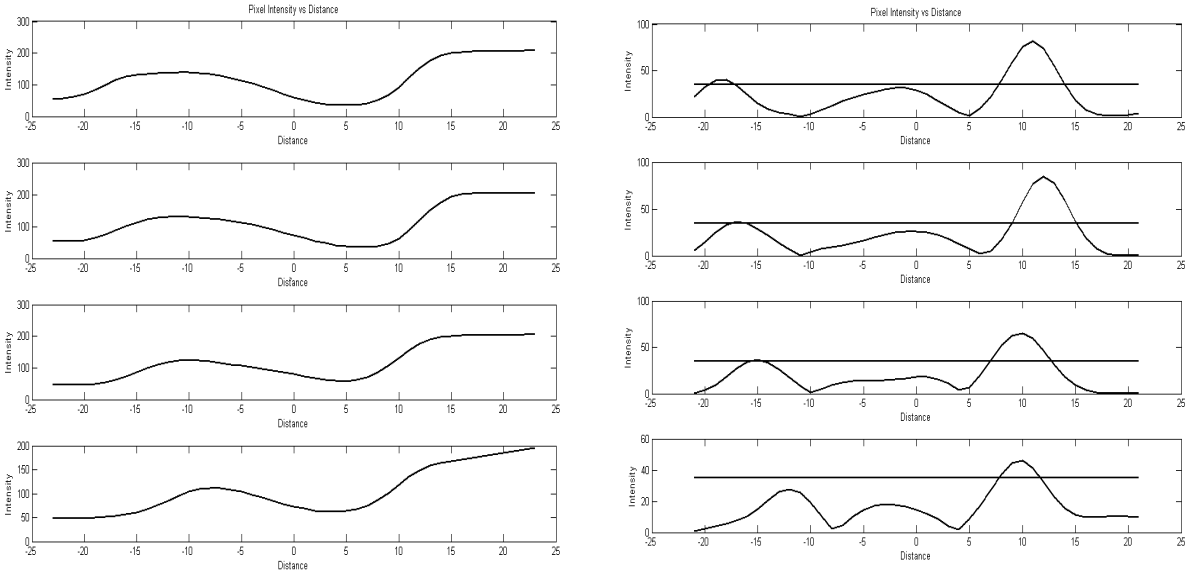


FIGURE 2.1: Line Search Example

Line search is a measurement approach based on casting search lines at normals to sampled points on the object's hypothesized edge locations. Edge features are points along these measurement lines that are judged to be features, *e.g.* where the image's first derivative is large. MacCormick [64] describes various methods to calculate a probability for each measurement line based on the distribution of features. Figure 2.1 show an example of search lines cast at normals to the projected edges of a articulated human body model.

Sminchisescu and Triggs [90, 91, 93] use a method where a point on the search line is considered an edge if it crosses an edge in an 'edge map', such as found by applying edge detection to the image set. They use heavy tailed distributions, such as Leclerc or Lorentzian, rather than incorporating 'non-detection' rates to determine the probability of each measurement line as done by Isard and Blake [48]. This method is computationally cheap as the edge map is only calculated once for each image, irrespective of the number of measurement function evaluations.

Isard and Blake [48] use a different approach, where a high pass filter is applied to pixel (intensity) values along each measurement line. Points where the filter response exceed a predetermined threshold are considered edges. This method produces oriented edges, so only the gradient magnitude in the direction of the search line is considered. Figure 2.2(a) show pixel intensity values from four of the search lines shown in Figure 2.1, with the output after high pass filtering shown in Figure 2.2(b). In this case edges were chosen as points exceeding the threshold indicated in Figure 2.2(b).



(a) Intensity Values vs Search Line Distance (b) Filter Response

FIGURE 2.2: High Pass Filtering along Search Lines

An advantage of the high pass filter method is that the shape of the high pass filter can vary depending upon the position of the sampled point on the object. If there is an expectation that some parts of the object will produce 'step like' edges, and other parts 'ramp like' edges, the filter used can be adapted to give improved results.

Line search methods are usually accurate in the location of global maxima, however they also tend to have smaller basins of attraction and hence more local modes than other methods. One reason for this is that the set of features detected in the background is not necessarily sparse, so a hypothesized edge location may have a feature set similar to what is expected even if the features are not from the object itself.

Multiple Feature Assessment Line Searches

Considering the edge feature set $\mathcal{E}$ along a measurement line, comprising of n distances between a found edge feature and the hypothesized edge location. When only a single feature is considered, the probability of this feature set can be reduced to $p(\mathcal{E}) = p(\min(\mathcal{E}))$. This can be generalized to consider all features along a measurement line, by finding the probability that one of the n features corresponds to the object's edge and the other $n-1$ features correspond to background clutter. Letting $\mathcal{C}(n|s)$ represent the probability of finding n clutter edges along a measurement line of length s, then:

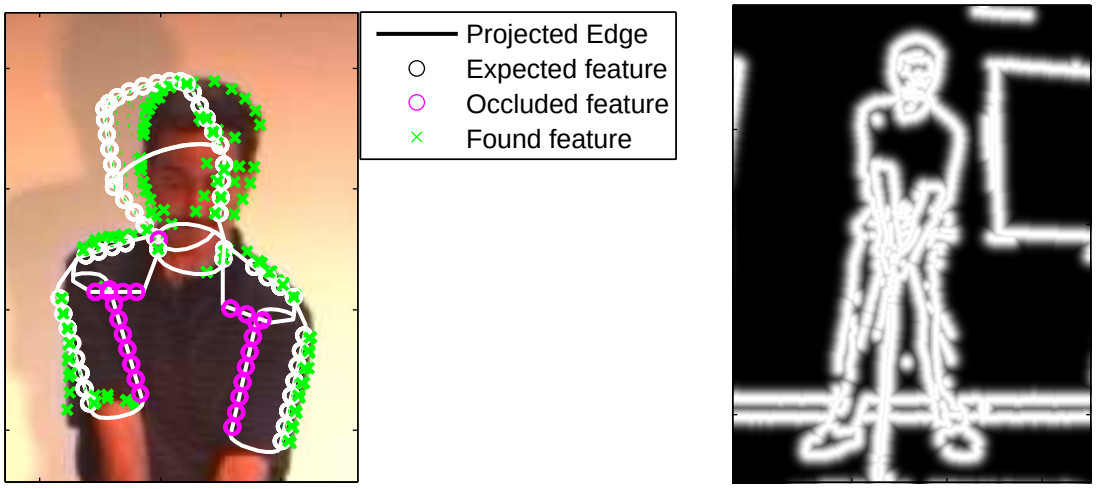
$$p(\mathcal{E}) = \frac{\mathcal{C}(n-1|s)}{s^{n-1}} \frac{(n-1)! \sum_{i=0}^{n} p(\mathcal{E}_i)}{n!} \tag{2.1}$$

$$= \frac{\mathcal{C}(n-1|s)}{n s^{n-1}} \sum_{i=0}^{n} p(\mathcal{E}_i) \tag{2.2}$$

The derivation of this formula is given by MacCormick and Blake [65]. $\mathcal{C}(n|s)$ can be learnt by casting measurement lines on an images without the object present, however a uniform distribution $\mathcal{C}(n|s) = s^{-1}$ is also a justifiable choice.

Nearest Edge Search

The nearest edge approach is based upon finding the distance between a sampled point on the object's hypothesized edge, and the nearest point in the image designated as an edge. Again edge detection is performed to determine which points in the image set are designated edges. Sampled points lie on the occluding contour of the projection of the object in a hypothesized state. This method differs from the line search method in that detected edge features do not need to lie at normals to the object's boundary. This eliminates the potential to use customized filters to detect edges, however it is not common practice to use these with line search methods unless prior information is available. The probability of each sampled point is calculated in a similar fashion to the line search methods. This edge measurement method was used by Balan, Sigal, and Black [10], Deutcher and Reid [33], and Deutcher, Davison, and Reid [32]. An example of this measurement process is shown in Figure 2.3(a).



(a) Nearest Edge Measurements (b) Edge Distance Map

FIGURE 2.3: Nearest Edge Search Example

An advantage of this method is that the radial search direction from sampled points removes the problem of measuring entirely different sets of pixels under object translation or re–parametrization, as discussed later in Section 2.3.4, and so fewer local modes are expected using this method. Another advantage of this method is that the distance between each pixel in the image and the nearest pixel designated as an edge needs only to be calculated once per frame, irrespective of the number of measurement function evaluations. These distances can be used to generate a 'map' of the probability that a point on a hypothesized edge location corresponds to an edge in the image. An example of such

a map is shown in Figure 2.3(b), where the light regions correspond to high probabilities. The use of an edge 'map' makes the nearest edge approach computationally very efficient.

A disadvantage of this method is that the search is undirected, making it more likely a randomly located feature (an edge not associated with the object of interest) will be found near a sampled point. As such less information is conveyed in the set of found edge features than when search lines are used, and may cause the nearest edge method to be less accurate than the line search method. A related issue is that because the search is not directed sampled points must be spaced further apart to prevent their detected edge features from being correlated. Where this method is used in experiments performed later in this thesis only the nearest edge feature is considered.

Evaluation of Measurement

Sminchisescu [87] introduces a matching consistency and data coupling term to improve the conditioning of his measurement function. The technique enforces a unique edge assignment principle as well as an edge symmetry principle, and include the possibility of adding further so called *Gestalt* properties. It is shown that this reduces the modality of the measurement function. This result was derived from an experiment tracking a naked arm. It would be an interesting study to see how well the symmetric properties of an object are preserved in the presence of loose fitting clothing, particularly as one side of the clothing may hang due to gravity.

2.1.3 Graph Theoretic Methods and Edge/Boundary Detection

A graph $G = (V, E)$ is a collection of vertices V connected pairwise by edges $E \subseteq V \times V$. A typical approach in image processing is to use image pixels as the graph vertices, with edges connecting neighbouring pixels. Typically in image processing the edge structure of the graph is arranged to be 4-connected, meaning that in the two dimensional case a pixel $I(x_i, y_i)$ is connected to another pixel $I(x_j, y_j)$, if and only if $|x_j - x_i| + |y_j - y_i| = 1$, where $x_i, y_i, x_j, y_j \in \mathbb{Z}$.

The computation of shortest paths is a fundamental problem in graph theory. The shortest path between two vertices is defined as the path between the two vertices with the minimum total vertex and edge costs. Flexible graph structures and the choice of edge and vertex costs allow many problems to be transformed into shortest path problems. There are many computationally efficient methods for solving shortest path problems.

Algorithm 1 details an algorithm for obtaining the shortest path between the first and last column of a rectangular trellis, with u rows and v columns. In this algorithm two vertices, $V(i, j)$ and $V(i^*, j^*)$, are connected if $|j - j^*| = 1$ and $|i - i^*| \leq 1$. Source vertices S are the acceptable start

locations, while sink vertices T are the acceptable end locations. The minimum cost to reach vertex $V(i,j)$ with weight (cost) $M(i,j) \geq 0$ is stored in $C^*(i,j)$, while the row index of the previous column on the shortest path is stored in $b(i,j)$.

1) Initialize the first column enforcing start locations:

foreach *row* $i = 1 : u$ **do**

$$L^*(i,1) = \begin{cases} M(i,1) & \textbf{if}(i,1) \in S \\ \infty & \textbf{otherwise} \end{cases}$$

$b(i,1) = i$;

end

2) Find the shortest path to each point in the trellis (i,j):

foreach *column* $j = 2 : v$ **do**

foreach *row* $i = 1 : u$ **do**

2a) Update the path cost:

$$C^*(i,j) = M(i,j) + \min(C^*(i-1,j-1), C^*(i,j-1), C^*(i+1,j-1));$$

2b) Update the backpointers:

$$b(i,j) = \arg\min_i (C^*(i-1,j-1), C^*(i,j-1), C^*(i+1,j-1));$$

end

end

3) Locate the end point with shortest path:

$$(P[v], v) = \arg\min_{(i,v)\in T}(C^*(i,v));$$

4) Follow the back pointers:

foreach *column* $j = v - 1 : 1$ **do**

$P[j] = b(P[j+1], j+1)$;

end

Algorithm 1: Shortest Path by Dynamic Programming

Graph theoretic methods and their continuous domain extensions [14] are commonly used to detect object boundaries in image segmentation [4–7, 11, 19, 20, 41, 90]. They present an advantage over pure image derivative based edge detection in that they can be guaranteed to produce closed contours/surfaces, and they require consistent image derivative value in large regions of the image. Depending upon the problem formulation, the vertex weights can be interpreted as the probability that a pixel is an edge based. The shortest path is then a maximum likelihood path through the potential edges.

Segmentation problems are less well constrained than most tracking problems, because most tracking problems assume some degree of knowledge of the object's shape. To alleviate the difficulties of the less constrained problem, images to be segmented are generally produced in a more controlled environment, *i.e.* higher quality images. Many of the methods referenced above produce optimal solutions to the segmentation problem, and so undesirable results are the result of the problem

formulation. One such problem is determining appropriate edge and vertex weights, or more generally choosing an appropriate metric. In poor quality images selecting a metric to give the desired result is highly problematic.

In the tracking problem the shape of the object is known, along with its deformation or articulation modes. Hence it is possible to derive a space consisting of every possible 'outline' of the object under a given projective transform. While it is highly desirable to design a space such that the output of the segmentation algorithms referenced above is the optimal contour constrained to this space of possible 'outlines', such a method is unknown to the author.

2.1.4 Region Based Measurement Methods

Region based measurements have variable accuracy depending on the method, but typically have have a large zone of attraction and so modes do not cluster.

Silhouette Methods

Silhouette methods are based on segmenting images into foreground and background regions. Deutscher and Reid [33], Sminchisescu [87], and Urtasun, Fleet, and Fua [99] use background subtraction techniques [35, 55] to perform this segmentation. Bray, Koller-Meier, Schraudolph, and Van Gool [16] uses a skin colour detection algorithm to perform the segmentation. While research continues on improving the robustness of such segmentation techniques, the accuracy is largely dependent upon the control that can be exerted over the tracking environment.

By using a controlled environment and multiple cameras (four or more), Luck, Hoff, Small, and Little [62, 63], Small [85], and Mikic, Trivedi, Hunter, and Cosman [70] produce silhouettes by background subtraction and apply back projection techniques [23, 34] to produce a convex volumetric representation of the object. A physics based scheme is used to fit a model state to the volumetric representation, rather than fitting a model state directly to the image.

Some authors such as Guo and Qian [44], Rosales and Sclaroff [77], and Sminchisescu, Kanaujia, Zhiguo, and Metaxas [88], use silhouette information to directly infer the object's state. Such methods are classed as *discriminative* algorithms and are discussed further in Section 2.5.

Optical Flow Methods

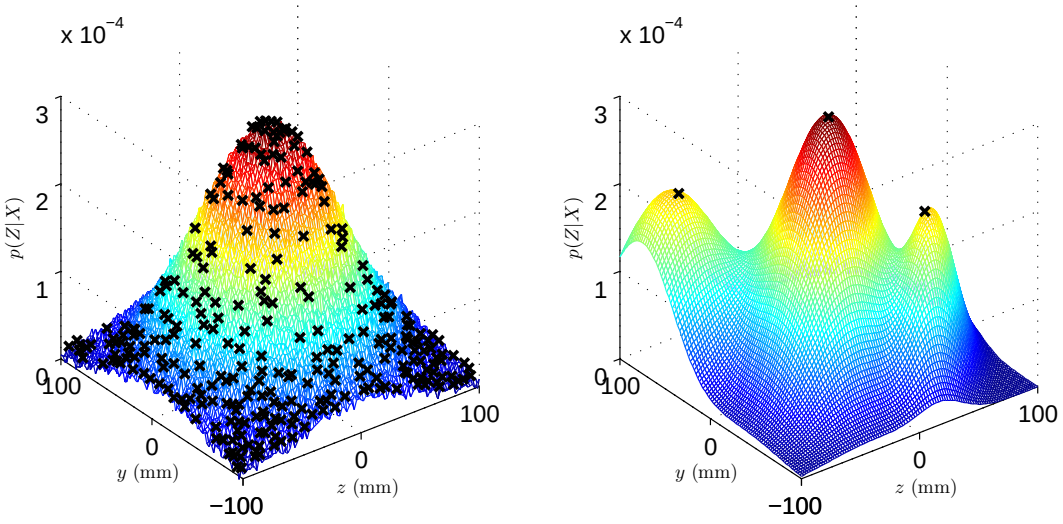
Optical flow methods use optical flow fields which describe the temporal 'flow' of image data. Black and Anandan [12] give methods to compute these flow fields. Optical flow has been used by Bregler and Malik [18], Cham and Rheg [21], Ju, Black, and Yacoob [52], Yacob and Davis [108], Wachter

and Nagel [103], Sundaresan and Chellappa [97] and Sminchisescu and Triggs [90]. These techniques rely upon having a registered intensity (or colour) template of the object to be tracked. Sidenbladh, Black, Fleet [82], and Urtasan *et al.* [99] use templates which are temporally updated.

Sminchisescu's [86] 'Image Flow Correspondence Field' is an exception which does not require a template, as it makes the hypothesized model state more probable according to the distance the interior of the object has moved.

2.2 Search and Optimization Overview

Bayesian tracking methods propagate a belief in the state $\mathbf{x}$ at time steps k, $k \in \mathbb{N}$. Searching for interesting regions in the observational likelihood $p(\mathbf{z}^k|\mathbf{x}^k)$ or posterior likelihood $p(\mathbf{x}^k|\mathbf{z}^{1:k})$ is an integral part of a tracking algorithm, with finding the globally maximum value of these distributions of particular importance. Finding the *global maximum* of an arbitrary function is an intrinsically difficult problem, and is in general *NP hard*.



(a) Gaussian with random noise, 586 Modes (b) Mixture of three Gaussians, 3 Modes

FIGURE 2.4: Different objective functions

Wolpert and Macready [106, 107] propose the interesting *No Free Lunch Theorems* for search and optimization. Roughly speaking, if an algorithm outperforms another algorithm for a particular class of problems, then it will have worse performance over all other problems. As such, care must be taken to select a algorithm that performs well on the distributions commonly encountered in visual tracking problems. Figure 2.4 shows two different likelihoods, with the positions of local maxima shown as black crosses. These likelihoods would intuitively be best searched by a different style of search algorithm. Following this, it is important to know the properties of the distribution to be searched

(and hence the measurement functions from Section 2.1) *before* selecting a search algorithm.

Through the evolution of research into visual tracking there is evidence of the incorporation of knowledge of the distribution being searched into the tracking algorithm. While these methods are discussed further in Section 2.4, some points of note are:

- The introduction of particle filtering due to the multiple modes in the observational likelihood [48].
- The use of curvature information during sampling, because camera geometry produces 'uncertain directions' in the observational likelihood [90, 92].
- Incorporating worm holes in the state space, due to sudden changes in a 2D object model's projected edges [46], and monocular depth ambiguities [91].

2.3 Newton-like Local Optimization

Newton like optimization methods are a type of iterative optimizer used to find *stationary points* in an objective function $\mathbf{f}(\mathbf{x})$, where a stationary point is a point at which the gradient is zero in all directions. Newton methods approximate a local region of the objective function as a quadratic function, parameterized by the gradient (Jacobian) $\mathbf{J}(\mathbf{x}) \equiv \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and the curvature (Hessian) $\mathbf{H}(\mathbf{x}) \equiv \frac{\partial^2 \mathbf{f}}{\partial \mathbf{x}^2}$. As such, Newton like optimizers perform a *local* search of the objective function, and so the output is generally highly dependent upon the *seed* estimate, which is the initial guess of the stationary point's location. Newton like optimizations are discussed in many books, *e.g.* Fletcher [38], Cooper and SteinBerg [25], and Jongen, Meer, and Triesch [51].

In a Newton based optimizer, an estimate of the location of a stationary point is given by solving the first order Taylor series approximation at the point X:

$$\begin{aligned} \mathbf{0} &= \mathbf{J}(X + \delta X) && (2.3) \\ &= \mathbf{J}(X) + \mathbf{H}(X)\delta X && (2.4) \\ \delta X &= -\mathbf{H}(X)^{-1}\mathbf{J}(X) && (2.5) \end{aligned}$$

Algorithm 2 gives an implementation of a basic Newton optimizer, where $\|.\|_\infty$ can be replaced by any type of norm, but but generally $\|.\|_2$ or $\|.\|_\infty$ is used.

The efficiency of the algorithm depends upon the extent that a local region of objective function can be modelled as a quadratic function. If the objective function is a quadratic function itself the

1) Starting with an initial (seed) estimate X_0, at iteration $k = 0$, with gradient $\mathbf{J}(X_0)$ and Hessian $\mathbf{H}(X_0)$, set *converged* = *false*.

while ¬ converged **do**

2a) Update the number of iterations, $k = k + 1$.

2c) Calculate the update distance, $\delta X_k = -\mathbf{H}(X_{k-1})^{-1}\mathbf{J}(X_{k-1})$.

2d) Calculate the new estimate, $X_k = X_{k-1} + \delta X_k$.

2e) Calculate the gradient $\mathbf{J}(X_k)$ and Hessian $\mathbf{H}(X_k)$.

2f) Determine if convergence has been achieved:

if $\|\mathbf{J}(X_k)\|_\infty \approx 0$

Set *converged* = *true*.

end

end

3) **Return** X_k.

Algorithm 2: Undamped Newton Optimization

exact solution is obtained in one iteration. The region around a point where the objective function approximates a quadratic is referred to as the *trust region*.

2.3.1 Hessian Dampening

The pure Newton optimization method solves for stationary points, however in most applications it is desirable to find either a local maximum or a local minimum. Considering a 1 dimensional problem where a local maximum is desired. If $\mathbf{H}(X_{k-1}) > 0$, the relevant stationary point is a minimum and hence $\mathbf{f}(X_k) < \mathbf{f}(X_{k-1})$, obviously not the desired result. If $\mathbf{H}(X_{k-1}) = 0$ then $\mathbf{H}(X_{k-1})^{-1}$ does not exist and the next iteration cannot be calculated. These situations occur in the n dimensional case when the ith eigenvalue e_i of $\mathbf{H}(X_{k-1})$ is zero or has the wrong sign (any $e_i \leq 0$ in a maximization problem). Furthermore, calculating the inverse of the Hessian in floating point arithmetic can be highly sensitive to rounding errors when the smallest and largest eigenvalues have significantly different magnitudes. Dampening is then also used to keep the eigenvalues within a desired (relative) range. In most formulations a scaled symmetric positive definite matrix $\mathbf{W}$ is added to the Hessian to calculate the damped update:

$$\delta X_k = -(\mathbf{H}(X_{k-1}) + \lambda\mathbf{W})^{-1}\mathbf{J}(X_{k-1}) \tag{2.6}$$

where λ is the scaling term, with $\lambda \leq 0$ and $\lambda \geq 0$ used for maximization and minimization problems respectively. The identity matrix is often used for $\mathbf{W}$. Fletcher [38] provides methods for determining a suitable value for λ.

2.3.2 Trust Regions

A form of dampening can also be used when the step size δX_k is larger than the *trust region*. Because the quadratic approximation does not hold outside of the trust region, there is no valid expectation of the objective function value outside of the trust region. As such moving to a point outside of the trust region may lead to a point where $\mathbf{f}(X_k) - \mathbf{f}(X_{k-1}) \ll 0$, which is undesirable for a maximization problem. The update step size is restricted to lie within this trust region:

$$\delta X_k = \min(\|\delta X_k\|_\infty, R_{k-1}) \times \frac{\delta X_k}{\|\delta X_k\|_\infty} \tag{2.7}$$

where R_{k-1} is the trust region around point X_{k-1}, and again $\|.\|_\infty$ can be replaced by any norm.

A simple method to update the trust region R_k at iteration k is given by Fletcher [38], where the actual function value increase:

$$\mathbf{g}(\delta X_k) = \mathbf{f}(X_k) - \mathbf{f}(X_{k-1}) \tag{2.8}$$

is compared to the predicted increase (using the quadratic approximation to the function):

$$\mathbf{g}^*(\delta X_k) = \mathbf{J}(X_{k-1})^T \delta X_k + \frac{1}{2} \delta X_k^T \mathbf{H}(X_{k-1}) \delta X_k \tag{2.9}$$

and the trust region is updated based on this ratio:

$$R_k = \begin{cases} \frac{R_{k-1}}{4}, & \textbf{if } \frac{\mathbf{g}(\delta X_k)}{\mathbf{g}^*(\delta X_k)} < \frac{1}{4} \\ 2R_{k-1}, & \textbf{if } \frac{\mathbf{g}(\delta X_k)}{\mathbf{g}^*(\delta X_k)} > \frac{3}{4} \textbf{ and } \|\delta X_k\|_\infty = R_{k-1} \\ R_{k-1}, & \textbf{otherwise} \end{cases} \tag{2.10}$$

2.3.3 Quasi-Newton Approaches

Quasi-Newton approaches are Newton like optimizers which use an approximation to the Hessian at each iteration. In the standard Newton approach, when the derivatives of the objective function are not available analytically they must be calculated by a numerical method such as central differences. In general, calculating the Hessian using central differences requires $4\left(\frac{d^2+d}{2}\right)$ objective function evaluations, where d is the dimensionality of the state space. The measurement (objective) functions used in visual tracking are typically computationally intensive, making the exact calculation of the Hessian computationally expensive. Quasi-Newton methods use approximations to update the Hessian to reduce the computational cost of the optimizer.

One commonly used method to update the Hessian is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [38], where the Hessian is updated using the gradient at the previous and current

state estimates, $\mathbf{J}(X_{k-1})$ and $\mathbf{J}(X_k)$, the previous Hessian $\mathbf{H}(X_{k-1})$, and the last update direction δX_k:

$$\gamma = \mathbf{J}(X_k) - \mathbf{J}(X_{k-1}) \tag{2.11}$$

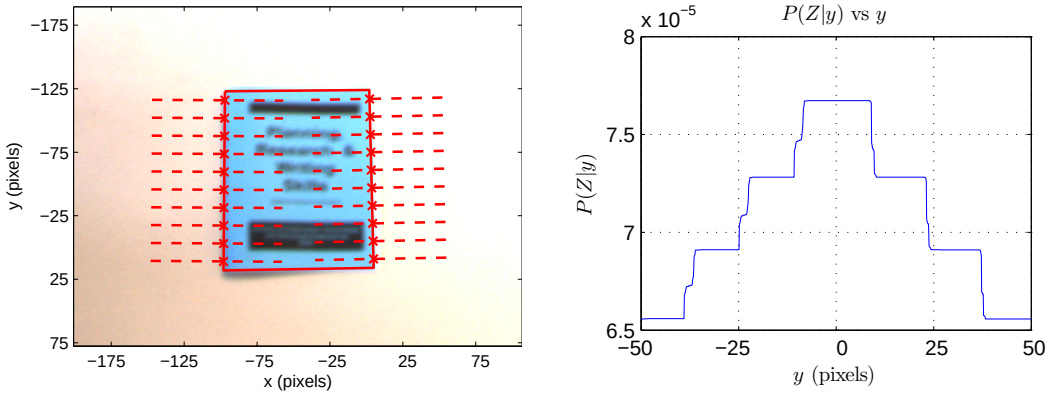
$$\mathbf{H}(X_k) = \mathbf{H} + \left(1 + \frac{\gamma^T \mathbf{H} \gamma}{\delta^T \gamma}\right) \frac{\delta \delta^T}{\delta^T \gamma} - \left(\frac{\delta \gamma^T \mathbf{H} + \mathbf{H} \gamma \delta^T}{\delta^T \gamma}\right) \tag{2.12}$$

where $\mathbf{H} = \mathbf{H}(X_{k-1})$ and $\delta = \delta X_k$ have been used for notational simplicity.

2.3.4 Suitability of Newton-Like Local Optimization for visual tracking

While Newton like optimizers are good general methods, the efficiency of the optimization depends upon the size of the *trust region*. Points with large high order derivatives will have small trust regions. Points where any 1^{st} or 2^{nd} order partial derivative of the objective function is undefined, designated *irregular points*[1], occur in the measurement (objective) functions used in visual tracking. Two potential sources of these irregular points are discussed below:

Discrete Features and Line Searches



(a) Original image and measurement lines $y = 0$ (b) Observational likelihood

FIGURE 2.5: Original image and probability distribution for an uncluttered desktop

The line search edge measurement method discussed in Section 2.1 use measurement lines cast at normals to a sampled point on the object's projected boundary. A probability or weight is given based on the distance of found features along these normals. An example of this is shown in Figure 2.5(a). While a small change in the object's state will cause only a small change in the sampled points location in the image, it potentially changes the entire set of pixels measured by the measurement line.

To demonstrate this, consider finding the y position of a book on an uncluttered desktop. To emphasize the effect of changing the entire set of measured pixels along a measurement line, only

[1] Not strictly the standard definition of an irregular point.

horizontal measurement lines have been used. Figure 2.5(a) shows the original image and measurement lines for the object state $X = 0$, while Figure 2.5(b) shows the likelihood as the object state is translated in the y direction.

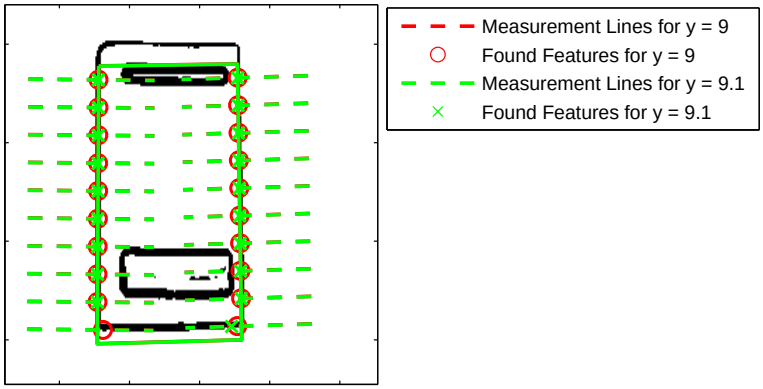


FIGURE 2.6: Feature set difference for two similar object states

The observational likelihood consists of a series of different levels, where the level changes as sampled points 'fall off' features in the edge map. Figure 2.6 shows the set of features found for two similar object states, $X = 9$ and $X = 9.1$, displayed over the edge map. At $X = 9$, the bottom left measurement line detects a feature near the hypothesized edge location. At $X = 9.1$ however, this sampled point has 'fallen off' the edge map and so no feature is detected. This causes a 1^{st} order discontinuity in the measurement function, irrespective of the probability distribution used to assess the likelihood of the found features along the measurement lines.

The likelihood shown in Figure 2.5(b) has a dense set of stationary points, where $\mathbf{J}(X) = 0$. As Newton like optimizers search for stationary points they are poorly suited to searching this probability distribution, because the seed estimate will most likely be a stationary point. Furthermore, the inverse of the Hessian $\mathbf{H}(X)^{-1}$ does not exist at **any** point. Practically, computing the Jacobian and Hessian using central differences, $\mathbf{J}(X) = \frac{f(X+\delta)-f(X-\delta)}{2\delta}$, makes an improvement step possible if 2δ is longer than each 'ledge'. While this example is somewhat contrived in that vertical measurement lines would greatly reduce this effect, it is still expected that this would adversely effect the performance of a Newton like optimization. In typical monocular tracking camera setups, there are generally many more horizontally orientated search lines than vertically orientated. This combination of discrete features, line searches, and Newton like optimization was used by Sminchisecu and Triggs [90].

An effect similar to the one described occurs when the set of sampled points on the object's hypothesized boundary changes. Figure 2.7 illustrates this, with solid lines representing search lines arising from one set of sampled points, and dashed lines search lines arising from another set. The difference in the choice of sampled points could easily arise from a different parametrization of the

object. Note that the two choices of sampled points produce entirely different sets of pixels being measured. Typically search lines are spaced such that they are the minimum distance apart such that edge features on different search lines are uncorrelated for a random image [64]. Because different sampled points for search lines produce search lines that are at most half of this distance apart, their is an expectation that edge features located along two corresponding search lines will produce a correlated set of features. However an entirely different set of image pixels is being measured, which can result in sudden changes in the measurement. While blurring an image helps to alleviate this, it does not remove the effect entirely, particularly when a discrete set of edge features is used.

Although not using a line search and discrete edge measurements, Bray, Koller-Meier, Schraudolph, and Van Gool [16, 17], and Kehl, Bray, and Van Gool [57] recognize the effect that sample placement has on the measurement function. In their 'stochastic meta-descent' optimization strategy, at each iteration sample placements are drawn stochastically rather than using a fixed set of sample placements on the object. They show improved performance of their local optimization compared to the quasi-Newton Levenberg-Marquardt optimization method, possibly because the stochastic sample placement has the effect of averaging the measurement function over different sample placements. Their measurement functions uses 3D measurements obtained from structured lighting or volumetric reconstructions, discussed further in Section 2.5.2, which produces a far denser set of measurable points than from 2D edge measurements.

Self Occlusions

Self occlusion modelling is an important part of the measurement functions used in visual tracking, as regions of the object can lie between a camera and other parts of the object. When an edge of an object is occluded by another part of the object, it is important to know that no meaningful edge

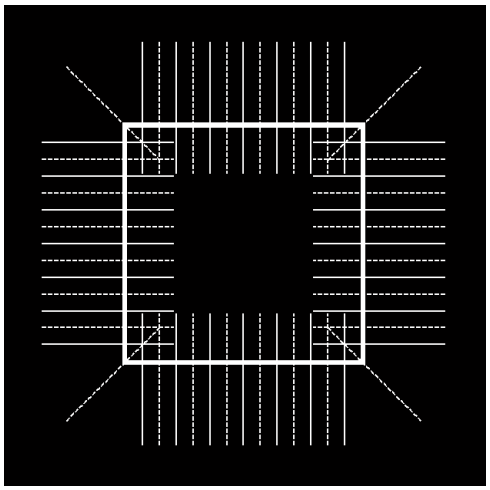


FIGURE 2.7: Different sets of sampled points used for measurement lines

measurements can be performed. Similarly, when two regions of an object that are the same colour are adjacent to each other, it useful to know that an edge is not expected to be detected between the regions. Figure 2.1 showed an example of occluded measurement lines.

When self occlusion models are used, the likelihood for the ith sampled point on the object's edge, $p_i(\mathbf{x})$, becomes a function of both it's position in the image, $\mathbf{r}_i(\mathbf{x})\colon \mathbb{R}^d \rightarrow \mathbb{R}^2$, an occlusion function $\mathbf{O}(\mathbf{r}_i, \mathbf{x})$, where $\mathbf{O}\colon \mathbb{R}^2 \times \mathbb{R}^d \rightarrow \{0, 1\}$ is a dichotomous function describing whether $\mathbf{r}_i$ is occluded. Since $\mathbf{O}$ is dichotomous and hence not differentiable everywhere, there must exist *irregular points* in any measurement function which models self occlusions in a dichotomous fashion.

2.4 Bayesian (Generative) Tracking Algorithms

Bayesian trackers propagate a belief in the state $\mathbf{x}$ at time steps t, $t \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of natural numbers. Describing the object's location by a probability distribution is effective because its true position often cannot be determined exactly, due to the high noise levels present in images. The Bayesian approach is subset of *generative* algorithms, denoted as generative because they involve generating hypothesized object states and evaluating a measure of the correctness of fit. In a Bayesian algorithm, this measure of correctness of fit is a probability. In some algorithms presented in this section, the Bayesian description is misused somewhat to include algorithms that use a weighting or cost function representation of the underlying probability distribution.

Arulampalam, Maskell, Gordon, and Clapp [8] provide an informative tutorial on Bayesian tracking algorithms. Considering the state sequence $\{\mathbf{x}^t, t \in \mathbb{N}\}$ given by:

$$\mathbf{x}^t = \mathbf{f}_s^t(\mathbf{x}^{t-1}, \mathbf{v}^{t-1}) \tag{2.13}$$

where $\mathbf{f}_s^t$ is the system process, and $\mathbf{v}^{t-1}$ is a possibly time varying i.i.d process noise sequence. The objective is then to estimate $\mathbf{x}^t$ using the set of measurements $\mathbf{z}^t$:

$$\mathbf{z}^t = \mathbf{f}_m^t(\mathbf{x}^t, \mathbf{u}^t) \tag{2.14}$$

where $\mathbf{f}_m^t$ is the measurement process, and $\mathbf{u}^t$ is a possibly time varying i.i.d measurement noise sequence. $\mathbf{z}^t$ is used to denote the set of available measurements at time step t, whereas $\mathbf{Z}^t = \{\mathbf{z}^i, i = 1, ..., t\}$ denotes the entire set of measurements up to time step t. Similarly $\mathbf{X}^t$ denotes the entire set of state beliefs up to time step t. The aim of the tracker is to find filtered estimates of $\mathbf{x}^t$ based on the set of all available measurements $\mathbf{Z}^t$. From a Bayesian perspective, the tracking problem is then to recursively calculate some degree of belief in the state space $\mathbf{x}^t$ at time step t. As such, it is necessary to construct the posterior probability density function (PDF) $p(\mathbf{x}^t|\mathbf{Z}^t)$, which can be calculated recursively in two stages: predication and update.

The prediction stage involves calculating the prior PDF $p(\mathbf{x}^t|\mathbf{Z}^{t-1})$. Assuming the previous posterior $p(\mathbf{X}^{t-1}|\mathbf{Z}^{t-1})$ is known[2], this can be done via the Chapman-Kolmogorov [74] equation:

$$p(\mathbf{x}^t|\mathbf{Z}^{t-1}) = \int p(\mathbf{x}^t|\mathbf{X}^{t-1})p(\mathbf{X}^{t-1}|\mathbf{Z}^{t-1})d\mathbf{x}^{t-1} \tag{2.15}$$

In the update stage the new measurement $p(\mathbf{z}^t|\mathbf{x}^t)$ is used to calculate the posterior density via Bayes' rule:

$$p(\mathbf{x}^t|\mathbf{Z}^t) = \frac{p(\mathbf{z}^t|\mathbf{x}^t)p(\mathbf{x}^t|\mathbf{Z}^{t-1})}{p(\mathbf{z}^t|\mathbf{Z}^{t-1})} \tag{2.16}$$

where $p(\mathbf{z}^t|\mathbf{Z}^{t-1})$ is a normalizing constant given by:

$$p(\mathbf{z}^t|\mathbf{Z}^{t-1}) = \int p(\mathbf{z}^t|\mathbf{x}^t)p(\mathbf{x}^t|\mathbf{Z}^{t-1})d\mathbf{x}^t \tag{2.17}$$

Filtering algorithms which provide an exact solution to the posterior density $p(\mathbf{x}^t|\mathbf{Z}^t)$ are classed as optimal Bayesian filters, whereas algorithms which approximate the posterior density are considered sub-optimal. An optimal filter implies that no other filter can outperform it. From a practical viewpoint, for an optimal Bayesian solution to exist the posterior density $p(\mathbf{x}^t|\mathbf{Z}^t)$ must be able to be determined analytically. Generally this is not the case, however solutions do exist in restrictive cases.

2.4.1 The Kalman Filter

Szeliski and Terzopoulos [98] introduce a framework for visual tracking based on the traditional Kalman filter [56][104]. The Kalman filter assumes that the posterior density at all times is Gaussian and hence can be parameterized by a mean and covariance. This assumption allows the Kalman filter to provide an exact solution to the posterior density $p(\mathbf{x}^t|\mathbf{Z}^t)$ at time step t, and as such is an optimal Bayesian filter. The original formulation of the Kalman filter [56] does not exploit any specific error distribution information beyond mean and covariance, as Bayes' rule is not applied. In the non-Gaussian case the Kalman filter may no longer provide an optimal solution.

Assuming the posterior density $p(\mathbf{X}^{t-1}|\mathbf{Z}^{t-1})$ at time step $t-1$ is Gaussian, $p(\mathbf{X}^t|\mathbf{Z}^t)$ will also be Gaussian provided certain assumptions hold [47]:

1. The process noise $\mathbf{v}^{t-1}$ and measurement noise $\mathbf{u}^t$ are drawn from known Gaussian distributions.

2. The system process $\mathbf{f}_s^t(\mathbf{x}^{t-1}, \mathbf{v}^{t-1})$ is a known linear function of $\mathbf{x}^{t-1}$ and $\mathbf{v}^{t-1}$.

3. The measurement function $\mathbf{f}_m^t(\mathbf{x}^t, \mathbf{u}^t)$ is a known linear function of $\mathbf{x}^t$ and $\mathbf{u}^t$.

[2] the initial pdf $p(\mathbf{x}^0|\mathbf{z}^0) \equiv p(\mathbf{x}^0)$ as $\mathbf{z}^0$ is the empty set.

The system and measurement processes, given by equations 2.13 and 2.14 respectively, can then be rewritten as:

$$\begin{aligned} \mathbf{x}^t &= F^t\mathbf{x}^{t-1} + \mathbf{v}^{t-1} \\ \mathbf{z}^t &= H^t\mathbf{x}^t + \mathbf{u}^t \end{aligned}$$

where F^t and H^t are known matrices defining the linear systems. The restricted case where $\mathbf{v}^{t-1}$ and $\mathbf{u}^t$ both have zero mean and are statistically independent is considered here, with Q^{t-1} and R^t used to denote their respective covariances. F^t, H^t, Q^t and R^t are written with the superscript t as they are possibly time variant.

Using $\mathcal{N}(x;\mu,\Sigma)$ to denote a Gaussian distribution with mean μ and covariance Σ, the Kalman filter algorithm derived using equations 2.15 and 2.16 can be viewed as the recursive relationship [8]:

$$\begin{aligned} p(\mathbf{x}^{t-1}|\mathbf{Z}^{t-1}) &= \mathcal{N}(\mathbf{x}^{t-1};\mu^{t-1|t-1},\Sigma^{t-1|t-1}) \\ p(\mathbf{x}^{t}|\mathbf{Z}^{t-1}) &= \mathcal{N}(\mathbf{x}^{t};\mu^{t|t-1},\Sigma^{t|t-1}) \\ p(\mathbf{x}^{t}|\mathbf{Z}^{t}) &= \mathcal{N}(\mathbf{x}^{t};\mu^{t|t},\Sigma^{t|t}) \end{aligned}$$

with

$$\begin{aligned} \mu^{t|t-1} &= F^t\mu^{t-1|t-1} \\ \Sigma^{t|t-1} &= Q^{t-1} + F^t\Sigma^{t-1|t-1}(F^t)^T \\ S^t &= H^t\Sigma^{t|t-1}(H^t)^T + R^t \\ K^t &= \Sigma^{t|t-1}(H^t)^T(S^t)^{-1} \\ \mu^{t|t} &= \mu^{t|t-1} + K^t(\mathbf{z}^t - H^t\mu^{t|t-1}) \\ \Sigma^{t|t} &= \Sigma^{t|t-1} - K^tH^t\Sigma^{t|t-1} \end{aligned}$$

where S^t is the covariance of the innovation term $\mathbf{z}^t - H^t\mu^{t|t-1}$, and K^t is the Kalman gain. Blake and Isard [13] describe various methods to set the initial conditions of a Kalman filter.

2.4.2 The Extended and Unscented Kalman Filters

The extended and unscented Kalman filters are used in cases where either the system process or measurement process are non-linear, however the posterior can still be described by a single mean and covariance. Both attempt to model the non-linearity locally using a linear function. As this is an approximation, both methods are sub-optimal Bayesian filters.

The extended Kalman filter [8, 104] utilizes the first term in a Taylor series expansion of the non-linear system and / or measurement processes, $\mathbf{f}_s^t$ and $\mathbf{f}_m^t$ from Equations 2.13 and 2.14. The

approximation $\hat{F}^t$ of $\mathbf{f}_s^t$ and $\hat{H}^t$ of $\mathbf{f}_m^t$ are given by:

$$\begin{aligned} \hat{F}^t &= \frac{d\mathbf{f}^t(\mathbf{x} = \mu^{t-1|t-1})}{d\mathbf{x}} \\ \hat{H}^t &= \frac{d\mathbf{h}^t(\mathbf{x} = \mu^{t|t-1})}{d\mathbf{x}} \end{aligned}$$

When the higher order terms of the Taylor series expansion are negligible the extended Kalman filter can be expected to perform well. It can be shown that the extended Kalman filter predicts the mean correctly up to the second order term in the Taylor series expansion, and the covariance up to the forth order term [53]. The local linear model can use higher order terms in the Taylor series, however becomes much more computationally expensive.

The unscented Kalman filter [53, 54] was introduced as an alternative method for modelling non-linearities in the system and / or measurement processes. The unscented Kalman filter is based on the intuition that it is easier to approximate a probability distribution than an arbitrary non-linear function. To achieve this, an arbitrary number of sample points $\mathcal{S}$ are deterministically drawn, so that they exhibit specific characteristics such as a known mean and covariance. The sample points are then propagated through the non-linear function giving a set of transformed points. Using the system process as an example, $\mathcal{S}' = \mathbf{f}_s^t(\mathcal{S})$. The mean and covariance of $\mathcal{S}'$ can then be recovered, and in this case where the system model is being approximated, giving the new prior distribution $p(\mathbf{x}^t|\mathbf{X}^{t-1})$. The unscented Kalman filter predicts both the mean and covariance correctly up to the forth order term. This is an improvement on the extended Kalman filter, with similar computational cost. The unscented Kalman filter can also be used to approximate discontinuous functions.

2.4.3 Particle Filtering

The particle filtering approach was first used in visual tracking by Isard and Blake [48] as a method to represent the posterior density at time step t, $p(\mathbf{x}^t|\mathbf{Z}^t)$ by a set of sampled points (particles). The motivation for using particle filters is that the measurement function is highly multi-modal in visual tracking problems, mostly due to feature clutter in the background. This violates the Kalman filter assumptions that the posterior density can be parameterized by a single mean and covariance. Particle filters do not make any assumptions about the nature of the posterior distribution, and the particles set can be used to describe multiple modes in this distribution. Since realistic tracking applications have highly multi-modal posterior distributions, maintaining multiple hypotheses about the object position is highly advantageous.

Each particle used in the particle filter represents a sampled point in the state space, and as such is a vector with length one greater than the dimensionality of the state space, where the extra degree

freedom encodes the particle's probability. The Condensation particle filtering algorithm presented by Blake and Isard [48] is shown in Algorithm 3. Note that the probability $p(\mathbf{x}^t|\mathbf{X}^{t-1})$ and hence prior $p(\mathbf{x}^t|\mathbf{Z}^{t-1})$ is not explicitly calculated, but rather is implicit in the position of the particles. Also note that the resampling operation is biased, where a disproportionate number of samples are drawn form high probability regions. This leads to a more accurate description of the posterior distribution in the high probability regions.

1) Starting with a set of n particles $\mathcal{S}^{t-1} = \{s_1^{t-1}, \ldots, s_n^{t-1}\}$ with probabilities $\boldsymbol{\pi}^{t-1} = \{\pi_1^{t-1}, \ldots, \pi_n^{t-1}\}$ that represent the posterior density $p(\mathbf{X}^{t-1}|\mathbf{Z}^{t-1})$ at time step $t-1$, n new samples are randomly drawn via a weighted sampling operation to form the set $\mathcal{S}^*$.

2) $\mathcal{S}^*$ is then propagated temporally via the system process, giving the new sample set $\mathcal{S}^t = \mathbf{f}_s^t(\mathcal{S}^*, \mathbf{v}^{t-1})$.

3) The measurement density $p(\mathbf{z}^t|\mathbf{x}^t)$ is approximated by finding the probability of each particle in the set $\mathcal{S}^t$, $\pi_i^t = p(\mathbf{z}^t|\mathbf{x}^t = \mathcal{S}_i^t)$.

4) Estimates of the object's state are calculated from the approximated posterior density. For example, the mean $\hat{x}^t$ is approximated by:

$$\hat{x}^t = E[\mathbf{x}^t|\mathbf{Z}^t] \approx \frac{\sum_{i=1}^{n} \pi_i^t \mathcal{S}_i^t}{\sum_{i=1}^{n} \pi_i^t}$$

Algorithm 3: Particle Filtering Algorithm

Particle filtering has been shown by many authors to be an effective method for performing visual tracking [21, 24, 31, 32, 39, 43, 46, 48, 49, 66, 69, 75, 90]. The principal drawback of the particle filtering approach is that the number of particles (and thus computational cost) required to approximate the posterior density grows exponentially with the dimensionality of the search space, which is know as the *curse of dimensionality*. MacCormick and Blake [66] show that the number of particles required for successful tracking is given by:

$$n \geq \frac{D_{min}}{\alpha^d} \tag{2.18}$$

where D_{min} and $\alpha \ll 1$ are constants, and d is the dimensionality of the search space. As a typical human model has at least 30 degrees of freedom, tracking using particle filters is very computationally expensive. To overcome this, various methods such as: partitioned sampling [66], importance sampling [49], annealed particle filtering [31], partitioned annealed particle filtering [32], covariance scaled sampling [90], and hyperdynamic sampling [92], have all been proposed to concentrate particles in regions of interest.

While the Condensation algorithm has been proven a highly effective strategy by many authors, it does have some drawbacks. King and Forsyth [58] discuss the Condensation algorithm as a Markov

Chain [40] and find that:

- Expectations computed have high variance so different tracker runs can yield very different answers.
- Expectations computed by a particular instance of Condensation have low variance making it look stable.
- The representation collapses to a single mode in time roughly proportional to the number of samples.
- The mode to which the representation collapses may bear no relationship to the dynamic model.
- The tracker appears to be following tight modes even in the absence of meaningful measurement.

Condensation can be thought of as a special case of Markov Chain Monte Carlo (MCMC) simulation. MCMC simulations are run for longer than the *burn in* time, allowing the simulation to converge. This is undesirable in particle filtering however as it is desirable to maintain multiple hypotheses. Hence it is the *burn in* time which is of interest, which is a property of the Markov chain itself. Using Condensation with a finite number of particles requires being able to bound the second eigenvalue of the Markov chain *below*, which can be as difficult as bounding it *above* [40].

2.4.4 Annealed Particle Filtering

The annealed particle filter [31] is a variation of the original Condensation [48] algorithm. In this variant, the steps used for each time step in the Condensation algorithm are repeated a number of times equal to the number of annealing layers, A, for each frame. The applied dynamics are stripped down to purely noise – there is no expected motion because it is the same frame being examined. It is based on the annealing technique presented by Kirkpatrick, Gellatt, and Vecchi [59], and independently by Černý [100]. A sequence of distributions $p_0(Z|\mathbf{x})$ to $p_{A-1}(Z|\mathbf{x})$ are used, where p_a differs only slightly from p_{a+1}, $a \in \{0, 1, ..., A-1\}$. The initial distribution $p_{A-1}(Z|\mathbf{x})$ is designed so that the Markov chain used to sample from it allows movement between larger areas of the state space [73]. Deutscher, Blake and Reid [31] advocate using a weighting function, $\mathbf{w}(Z, \mathbf{x})$, to represent the probability density $p(Z|\mathbf{x})$. The weighting function is adjusted with each annealing layer so that $\mathbf{w}_a(Z, \mathbf{x}) = \mathbf{w}(Z, \mathbf{x})^{\beta_a}$, with $1 = \beta_0 > \beta_1 > ... > \beta_{A-1} > 0$. This has the effect of smoothing the weighting function initially and then peaking the weighting function more heavily as the particles concentrate. Deutscher *et al.* use a heuristic to choose the values of β_a based on the *particle survival rate*, where the particle

survival rate, α, is defined as the percentage of particles in the set which will be resampled. Neal [73] suggests sampling from the distribution given by the weighted product of the observational likelihood and the prior:

$$\mathbf{w}_a^t(\mathbf{z}^t, \mathbf{x}^t) = \mathbf{w}_0^t(\mathbf{z}^t, \mathbf{x}^t)^{\beta_a} \mathbf{w}_0^t(\mathbf{x}^t, \mathbf{Z}^{t-1})^{1-\beta_a} \tag{2.19}$$

Figure 2.8 shows an example of the annealing process used to approximate an unknown one dimensional probability density function (PDF). This PDF was obtained from an image of a book on a cluttered desktop, exhaustively sampled at a one pixel resolution. This image is shown in Figure 2.9, where the location of the book at the global maximum is displayed, as well as the location of the book corresponding to the next largest maxima. Each graph plots $p(Z|X)$ vs x translation. Vertical y translation was set to the correct value and kept constant.

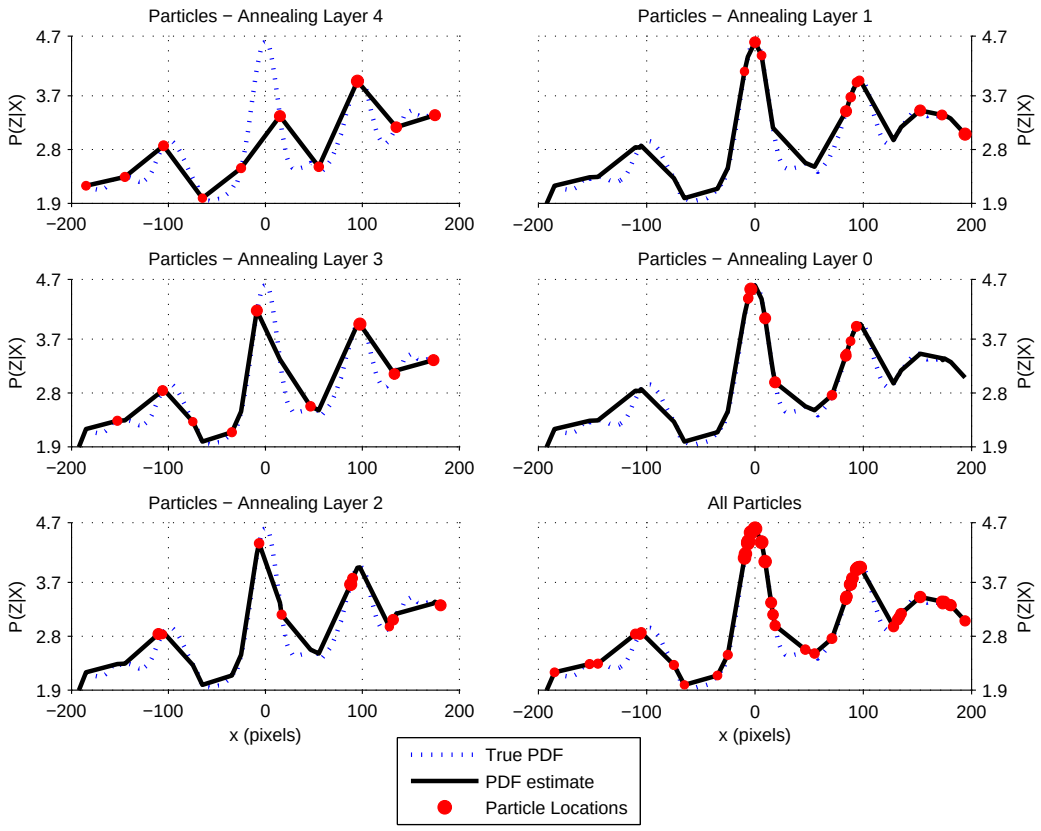
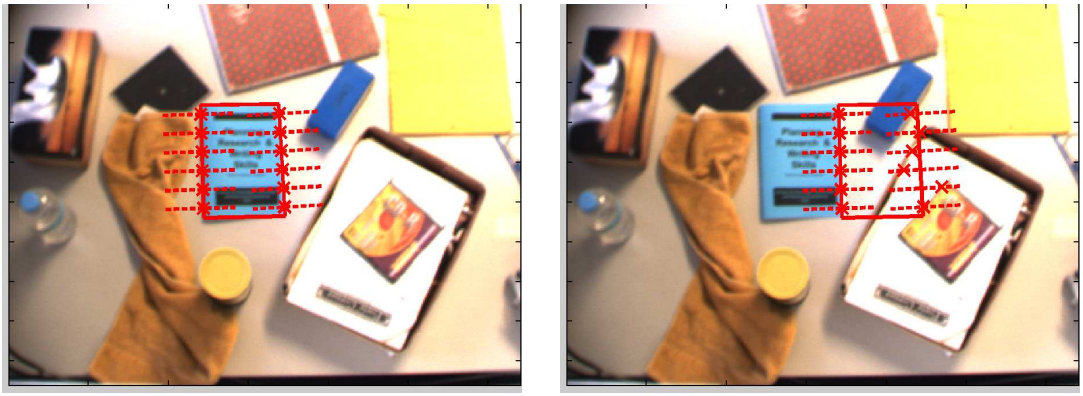


FIGURE 2.8: The annealing process used to estimate an unknown probability density function

In the first annealing layer, no prior knowledge about the book's location was assumed, so particles were drawn uniformly across the possible range of x values. A weighted sampling operation was used, with noise dynamics governed by a Gaussian with zero mean and a standard deviation of 10. The estimated PDF is a linear interpolation of the probabilities of the particles. Note how the estimated PDF for each successive annealing layer more correctly models the true PDF in the high probability regions. Also note that if more annealing layers were added, the particles would converge to a single



(a) True object position at $x = 0$

(b) Object position at $x = 100$, corresponding to the second largest local maxima

FIGURE 2.9: Object positions at $x = 0$ and $x = 100$

mode as discussed in Section 2.4.3, and so the number of particles and annealing levels must be chosen carefully. This convergence to a single mode will occur even if the likelihood consists of two identical modes. The shape of the underlying likelihood being searched then has an impact on the number of samples and annealing level used, as large number of small modes will increase the time taken for convergence to occur.

Deutscher, Davison, and Reid [32] show that the convergence speed of the annealing process can be improved by dynamically adjusting the sampling noise. Typically the variance of the sampled particles reduces as the estimated PDF begins to converge to the true PDF, and reducing the sampling noise narrows the search in the regions of interest. In multi-dimensional problems, some dimensions may have have a larger influence on the probability than others. In such cases the variance in these 'important' dimensions reduces more quickly than in the others. This is investigated later in this thesis.

2.4.5 Hyperdynamic Annealed Particle Filtering

Sminchisescu and Triggs [92] propose a method to alleviate the problem of particles filters becoming trapped in sub–optimal local maximas. Their 'hyperdynamic' sampling scheme enhances the sampling rate near *transition points* in the distribution. This method is based on the approach developed by Voter [101, 102], to estimate transition rates between different atomic arrangements. When searching for maximums, transition points are points in the state space with negative curvature along all but one of the eigenvectors of the measurement function's Hessian matrix at the point, $\mathbf{H}(X)$, while the remaining eigenvector $\mathbf{V}_1$, has positive curvature $e_1 > 0$, and vanishing derivative along

this eigenvector:

$$g_{p1} = \mathbf{V}_1^T \mathbf{J}(X) = 0 \tag{2.20}$$

$$e_1 > 0 \tag{2.21}$$

$$e_i < 0, i \in \{2, ..., d\} \tag{2.22}$$

with $\mathbf{J}(X)$ the Jacobian at X. A *saddle point* is a transition point that is locally maximal in all but one eigenvector direction and locally minimal in the remaining one. A transition point must be passed when moving from one local maximum to another, and defines the boundary of the zone of attraction discussed in Section 2.1. The set of transition points surrounding a mode form its transition boundary. Hence finding a transition point facilitates finding a new local maximum.

While annealing methods typically sample modified distributions in order to increase mixing, the hyperdynamic sampler specifically focusses on regions more likely to contain transition points. This is achieved by biasing the weighting function with a term that reduces the weight of points unlikely to be transition points, concentrating samples around the transition points to facilitate finding new modes. The bias term f_b is given by:

$$f_b = \frac{h_b}{2}\left(\frac{e_1}{\sqrt{e_1^2 + g_{p1}^2/m_d^2}} - 1\right) \tag{2.23}$$

where h_b is a constant controlling the strength of the bias, and m_d is an estimate of the average distance between nearby minimums. On the transition boundary, $g_{p1} = 0$ and $e_1 > 0$, and so $f_b = 0$. Equation 2.22 gives a good approximation to the structure of the transition surface around saddle points, however its approximation in Equation 2.23 may also be satisfied in other areas such as around local minima. Practically this is of little importance since the unbiased weight is large in these areas, but it does gives insight into the useful range of h_b,

The bias introduced into the weighting function means some form of fair sampling must be performed later, such as given in [3, 80, 94]. The hyperdynamic sampling method is shown to significantly reduce the time annealing methods spend trapped in local modes.

The disadvantage of this method is the extra computational cost incurred in computing the extra curvature information. This can be reduced by numerically estimating the principal positive eigenvector $\mathbf{V}_1$, by maximizing:

$$\left[\mathbf{J}(X + \eta\mathbf{V}_1) - \mathbf{J}(X - \eta\mathbf{V}_1)\right]/2\eta \tag{2.24}$$

subject to:

$$\|\mathbf{V}_1\| = 1 \tag{2.25}$$

where $\|.\|$ denotes the 2–norm.

2.4.6 Transition Point Road Maps

The transition point road maps approach, as presented by Sminchisescu and Triggs [89, 93], seeks to find transition points in the surface to be searched. As such, it is similar to the hyperdynamic annealed particle filtering method discussed in Section 2.4.5. It is removed from pure particle filtering approaches as local optimizations are used to find saddle points, rather than using the likelihood a particle is a transition point to determine its weight for the weighted resampling operation. A saddle point is a transition point that is locally maximal in all but one eigenvector direction and locally minimal in the remaining one. Thus, a saddle point is also a stationary point, and so can be expressed as:

$$\begin{aligned} \mathbf{J}(X) &= \mathbf{0} && (2.26)\\ e_1 &> 0 && (2.27)\\ e_i &< 0, i \in \{2, ..., d\} && (2.28) \end{aligned}$$

where e_i is again an eigenvalue of the Hessian $\mathbf{H}(X)$. A local Newton optimizer seeded slightly in the positive direction of the locally minimal eigenvector will converge to a different mode than if seeded slightly in the negative direction, making the saddle points an effective method of finding new local modes.

Two methods for finding sample points are presented by Sminchisescu and Triggs [93]. The first is an eigenvector tracking approach where a modified Newton optimization step is used to find saddles points. To stabilize the iteration near a saddle point, the magnitude of the eigenvalues of the Hessian $\mathbf{H}(X)$ is increased while the sign is preserved. This is expressed in the eigenbasis of the Hessian, $\mathbf{H}(X) = \mathbf{V}\mathbf{D}\mathbf{V}^T$ where $\mathbf{D}$ is a diagonal matrix of the eigenvalues, $\mathbf{D} = \mathrm{diag}(e_1, e_2, ..., e_d)$, and $\mathbf{V}$ is a matrix of the eigenvectors. In the undamped case, the Newton update given in Equation 2.5 can be modified:

$$\begin{aligned} \bar{g}_i &\equiv \mathbf{V}_i^T \mathbf{J}(X) \\ \mathbf{U} &= \left\{ \frac{\bar{g}_1}{e_1}, \frac{\bar{g}_2}{e_2}, ..., \frac{\bar{g}_d}{e_d} \right\} \\ \delta X &= -\mathbf{V}\mathbf{U}^T && (2.29) \end{aligned}$$

where $\bar{g}_i$ is the projection of the gradient onto the ith eigenvector, and $\mathbf{U}$ controls the step length along each eigenvector. For damped Newton optimizations as discussed in Section 2.3.1, $\mathbf{U}$ can be reformulated as a function of the dampening variable λ. $\mathbf{U}(\lambda)$ can then be written as:

$$\mathbf{U}(\lambda) = \left\{ \frac{\bar{g}_1}{e_1 + \varsigma_1\lambda}, \frac{\bar{g}_2}{e_2 + \varsigma_2\lambda}, ..., \frac{\bar{g}_d}{e_d + \varsigma_d\lambda} \right\} \quad (2.30)$$

where $\varsigma_i \in \{-1, 1\}$ is the desired sign pattern along the ith eigenvector. Setting $\lambda > \max_i(-\varsigma_i e_i, 0)$ ensures that the iteration moves uphill along all eigenvectors with $\varsigma_i = -1$, and downhill along the others.

While the above description provides a method for updating an estimate of a saddle points location, a problem exists in associating the ith eigenvector of the Hessian at iteration k, $\mathbf{V}_i^k$ with an eigenvector $\mathbf{V}_j^{k-1}$, $j \in \{1, 2, ..., d\}$, at the previous iteration. This is necessary to ensure a consistent sign pattern between iterations, so $\varsigma_i^{k-1} = \varsigma_j^k$ for associated eigenvectors $\mathbf{V}_i^k$ and $\mathbf{V}_j^{k-1}$. The reader may refer to Sminchisescu and Triggs [89, 93] for details regarding this association.

2.4.7 Covariance Scaled Sampling (CSS)

In tracking problems, there are often some state space directions which alter the object's appearance in the image less than others. The ambiguous depth direction in the monocular tracking case is an example of this. As such there exists *uncertain directions* in the observational likelihood $p(Z|X)$ such that $p(Z|X) \approx p(Z|X + \delta X)$ for significantly large δX.

Sminchisescu and Triggs [90] exploit this property of the observational likelihood during the sampling operation of a particle filter. Instead of drawing a sample from a Gaussian distribution with a predetermined covariance, they instead infer the covariance from the curvature of the of the measurement function at the point to be sampled. Samples are then drawn from the population:

$$\mathcal{N}(\mu = X, \Sigma = s\mathbf{H}(X)^{-1}) \tag{2.31}$$

where $\mathcal{N}(\mu, \Sigma)$ is a Gaussian distribution with mean μ and covariance Σ, $\mathbf{H}(X)$ is the Hessian of the measurement function at the point X, and s is the covariance scaling term. This concentrates samples along the directions of $\mathbf{H}(X)$ with small curvature, the uncertain directions.

This process is equivalent to modelling the posterior density as a set of n Gaussian distributions $\mathcal{N}^* = \{\mathcal{N}_1(\mu_1, \Sigma_1), ..., \mathcal{N}_n(\mu_n, \Sigma_n)\}$, and so unlike a 'pure' particle filtering algorithm makes assumptions about the shape of the posterior distribution – that it can be modelled by a set of Gaussian distributions. As the number of Gaussian distributions required to model the posterior must be kept low to limit the computational cost, Sminchisescu and Triggs [90] prune the number of mixture components used to represent the posterior to an small arbitrary number. As such, the effectiveness of this algorithm depends on how well the assumption that the posterior can be modelled by a compact set of Gaussians holds. This effectiveness is highly dependant on the success of the measurement function on the underlying image data.

Several authors have addressed the problem that sampling based searches such as the annealed particle filter (Section 2.4.4) converge slowly to modes [21, 24, 46, 69, 75], especially when the

measurement likelihood peaks deep in the tail of the prior. Sminchisescu and Triggs [90] alleviate this problem by performing a Newton optimization for a selected subset of the particles. This has the advantage that the Hessian of a point to be resampled is known from the final iteration of the optimization. Sminchisescu and Triggs use a line search measurement scheme, which has some undesirable properties used in conjunction with a Newton optimizer as discussed in Section 2.3.4. To improve the conditioning of the posterior distribution, the edge measurements are integrated with: intensity features arising from optical flow data (Section 2.1.4), and non–dynamic joint angle priors.

The non–dynamic joint angle priors are used for joint angles that are difficult to measure, such as the shoulder rotation, and are 1D Gaussian distributions with large variances. Such priors are not necessarily applicable to the tracking of highly athletic motions however, as there is an expectation that each joint will move through a large angular range, and the observational likelihood should be diminished in these ranges. It is also difficult to select the (effective) scaling of these priors without knowing the 'strength' of the measurement function, in general how well the object contrast with the background. These priors will have a smoothing effect on the observational likelihood, but the extent to which they reduce the number of local modes in the measurement function is not known to the author.

Another interesting part of the Newton like optimization used by Sminchiescu [86] is the Jacobian and Hessian approximations. These approximations assume that, when a sampled point $\mathbf{r}$ on the object's occluding contour and its normal direction $\mathbf{n}$ are perturbed by a small change in the object state, that the detected edge feature set $\mathcal{E}$ remains unchanged. Similarly, it is assumed that the point $\mathbf{r}$ remains in its current state of occlusion. While these assumptions help to guarantee that Jacobian and Hessian are always well defined, following the discussion Section 2.3.4 and noting that this measurement method is the same method used by Sminchisescu, there is no guarantee that traversing a calculated update direction will help maximize the objective function.

2.4.8 Kinematic Jump Sampling

In monocular tracking problems, there is an inherent forwards–backwards depth ambiguity for each link $\mathbb{L}$ in the kinematic chain. Considering a line $\mathcal{L}$ containing the camera's position $C_1 \in \mathbb{R}^3$ and the link's end $\mathbb{L}_e$, and a sphere $\mathcal{S}$ describing the possible positions the link's end can take given the link's start position $\mathbb{L}_s$ remains unchanged, *i.e.* a sphere centered at $\mathbb{L}_s$ with radius equal to the link's length $\mathbb{L}_l$. A forwards–backwards ambiguity exists if there are two points of intersection between $\mathcal{L}$ and $\mathcal{S}$, which will always occur except in the rare occurrence that $\mathcal{L}$ is contained within the sphere's tangent space. The link at either of these points will look almost identical in an image, ignoring a

minor change to the link's apparent width.

Sminchisescu and Triggs [91] address this issue by proposing a kinematic jump sampling process for use within a particle filtering framework. When a particle is selected for (re)sampling, the new sample point drawn may be reflected (in the forwards-backwards sense) about a kinematic sub-chain of the kinematic model. A method is proposed where many predetermined kinematic sub-chains 'vote' for which of the sub-chains the new particle is reflected about. The weight of each vote is dependent upon the curvature of the measurement function at the point being resampled. Sminchisescu and Triggs show that drawing new sample sets using this method produces more 'interesting' sample sets than when these reflections are not used.

2.4.9 Parallel Filter Banks

The large number of degrees of freedom (DOFs) required to model the human body is an inherent problem in visual tracking, as the number of particles (hence computational cost) required for particle filtering based approaches grows exponentially with the dimensionality of the state space. This so called *curse of dimensionality* can be addressed by decomposing the tracking problem from tracking a single object represented by 25-50 parameters into tracking several objects each described by only a smaller number of parameters (e.g. body parts described by affine transforms). A further computational step is used to find configurations of each component part that match the geometric constraints of the full object.

These approaches rely on being able to detect each component part individually (a "bottom up detector"). Felzenszwalb and Huttenlocher [37], Moon and Chellappa [71], and Ramanan, Forsyth, and Zisserman [76] use filters designed for their given component part representation, whereas Sigal, Bhatia, Roth, Black, and Isard [83] and Sheikh, Datta, and Kanade [81] use filters learnt from training data.

The individual component detectors then need a framework to combine their results into the single object in a geometrically plausible configuration. Moon and Chellappa use a ridged arrangement of individual components, while Felzenszwalb and Huttenlocher, and Ramanan *et al.* arrange the individual components into a non-rigid "pictorial structure", where each part is the vertex in a graph, with each edge being analogous to a virtual spring which binds the parts together. Sigal *et al.* use a similar approach however with learnt conditional densities between the parts replacing the virtual springs, and allowing dependencies unconnected parts where prior motion knowledge is available (i.e. when a known motion is being tracked).

While these approaches are computational more efficient than approaches without this problem

decomposition, it is limited in that measurements cannot be taken from the 'joins' between links. Later in this thesis it is shown that performing measurements in these regions reduces the number of local maxima in the observational likelihood, while improving its accuracy. The effectiveness of this approach is the dependent upon how well each link can be localized independently.

2.5 Discriminative Tracking Algorithms

Discriminative tracking algorithms find deterministic solutions to the state sequence $\{\mathbf{x}^t, t \in \mathbb{N}\}$. These algorithms use measurement data directly to hypothesize the model state, whereas generative algorithms generate hypothesized model states and use measurement data to evaluate their likelihoods.

2.5.1 Mappings from Silhouette Spaces

Some authors such as: Guo and Qian [44], Rosales and Sclaroff [77], and Sminchisescu, Kanaujia, Zhiguo, and Metaxas [88], use silhouette information in discriminative algorithms to perform tracking. These methods rely on learning mapping between a feature (in this case silhouette) space, and the state space comprising of the joint angles of the subject. This approach is not focussed upon in this thesis because the learning process relies on a fixed camera position relative to the subject.

2.5.2 Volumetric Tracking

When multiple cameras (typically four or more) are used for tracking, it is possible to form a volumetric hull of the object, allowing an object model to be fitted to the volumetric hull. The volumetric hull space is a discrete representation of $\mathbb{R}^3$, where each discrete point is referred to as a voxel. Typically the volumetric hull is formed from a set of silhouette images, where each silhouette image is a binary image indicating if a ray from the camera center through the image plane intersects the object. Background subtraction methods, such as proposed by Elgammal, Harwood, and Davis [35], can be used to form these silhouette images. The union of these rays for all points in all silhouettes defines a generalized cone within which the object must lie [34]. Many algorithms exist for constructing volumetric hulls from a set of silhouette images. An example of one method is given by Cheung, Kanade, Bouguet, and Holler [23].

Luck, Hoff, Small, and Little [62, 63] and Small [85] perform human tracking using the volumetric hull approach. They use background subtraction information from four cameras to create the silhouette set, and construct the volumetric hull from this. The model state X^t is deterministically found by a physics based fitting algorithm using the volumetric hull at time step t, denoted $\mathbf{V}^t$, and

the previous object model state, $\mathbf{x}^t = f(\mathbf{x}^{t-1}, \mathbf{V}^t)$. They achieve human body tracking at around 25 frames per second. Kehl, Bray, and Van Gool [57] also use a volumetric hull approach, however fit the model to the volumetric hull using a 'stochastic meta-descent' optimization strategy, rather than the physics based fitting algorithm.

Corazza *et al.* [26] fit a body model by embedding location information for ten joint centers in a subject specific free-form surface, where the optimal locations of joint centers in the 3-D mesh are learnt. The model was shown to be sufficiently accurate for both kinematic (joint centers) and morphological (shape of the body) information to allow accurate tracking with generative tracking systems.

Sundaresan and Chellappa [96] solve for human posture by building an adjacency matrix for the voxel data, and use a Laplacian Eigenmap to transform different nonrigid chains (such as the limbs in the human body) to nodes on separate smooth 1D curves in the new space, according to their position along the articulated chain. This enables a spline-fitting algorithm to segment the different articulated chains, as well as allowing more complex poses such as those where the limbs form loops.

The disadvantage of this method is the high number of cameras required to form the volumetric hull, and the reliance upon successful background segmentation. Background segmentation is particularly problematic in outdoor settings.

2.5.3 Strong Motion Models

Strong motion models can be used to track humans in video sequences when the behaviour to be tracked is known in advance. Urtasun, Fleet, and Fua [99] track a golfer from monocular images using strong motion models. The strong motion models allow the state space dimensionality of their human model to be reduced from 72 dimensions down to 4 via a principal component analysis [50]. The image sequence is preprocessed and the position of the golf club is found by a specific club tracking algorithm [61]. Key positions for the club are used to transform the image sequence into a time normalized sequence, so the entire swing lasts for one unit of normalized time. An objective function utilizing both image measurements and deviation from the learnt motion model at the normalized time is minimized to give the state position at the current time step.

It could be argued that with this approach the problem has become constrained to the point where it becomes trivial. Urtasun *et al.* believe this is not the case as tracking fails when selected image measurements are removed from the objective function, causing the objective function to have multiple solutions. To illustrate failure they show a solution to the model state that should probably not be in the state space at all (Chapter 5 discusses state spaces further). Urtasun *et al.* also show an example

of mostly successful tracking of a shorten swing (the back lift is not as high). The tracking errors are stated to be caused from the motion being different to the training data, and propose using a larger training set as a solution to this problem.

Broadening the training set is problematic however. This approach is driven from having a strong motion model, and adding different swings to the training set will weaken this motion model. Image measurement based probability distributions or cost functions will be multi-modal, particularly in the monocular case as there is a forward-backwards ambiguity. It is highly unlikely that with a broadened training set, there will be no instances where this ambiguity can not be resolved from the motion model. This problem is inherent in deterministic trackers, for unlike Bayesian approaches they can not maintain multiple hypotheses.

2.5.4 Histogram of Oriented Gradients

Dalal and Triggs [29] present an innovative method for human detection based upon the histograms of oriented gradients. This is a learning approach where more than one thousand images of people are used to learn the distribution of gradients using a support vector machine. This approach proved highly successful when applied to the human detection problem.

The human detection problem differs from the tracking problem as the output of the detector is binary indicator of whether a person is present in the image, rather than a state vector containing the person's position and posture. Despite this, the method certainly has promise as part of the tracker's initialization. A deeper understanding of the gradients patterns learnt by the SVM could also provide extra measurement information that could be utilized by generative trackers.

3

Modelling

This chapter describes the various models used in the experiments presented later in this thesis. The 43 degrees of freedom kinematic chain representing the human body is explained. Details of the link projection method are given, as well as a method to generate a self occlusion map for a hypothesized model state. The cameras used for video capture are detailed next, followed by the projection and radial distortion models. The image gradient metric used implicity by all of the edge measurement functions considered in this thesis is given next. Finally, a region measurement scheme which assumes body parts are of a uniform colour is discussed.

3.1 Human Body Modelling

In the experiments presented in this thesis the human body is represented using an articulated model – a series of *links* joined together to form a kinematic chain. Each link represents a bone in the human body, and has a set of permitted rotations and a surface model.

3.1.1 Degrees of Freedom

Human tracking applications generally use about 30 degrees of freedom (DOFs) to model a person. The primary application of the work presented in this dissertation is tracking a human during a golf swing, which is a highly dynamic motion. As such a 43 DOFs model is used, consisting of 3 translational and 40 rotational DOFs as shown in Figure 3.1.

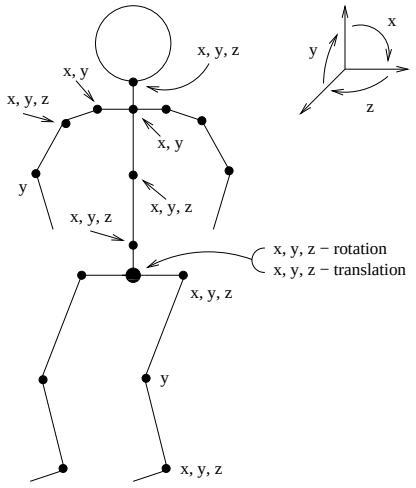


FIGURE 3.1: The 43 DOFs of the human model

These DOFs were selected by hand fitting a generic 30 DOF model to an image sequence showing a golf swing. DOFs were adding as necessary to ensure the model was able to correctly represent the golfer's posture. Some choices are not intuitive, for example using three DOFs to model the ankle which is a simple hinge. In this case, the first two DOFs were used because the angle of the hinge relative to the shin varies between people, and it is not feasible to infer this angle from the image data. The final degree of freedom was added to compensate for the deformable nature of the foot, and that the foot can move inside the shoe without altering the shoe's position. This deformation lead to situations where the sole of the shoe was observed to be flat on the ground, which could not be modelled without the final degree of freedom.

Another region of interest is the golfer's back. The back is modelled simplistically using two non-deformable links. This is far from a realistic model of the complex human vertebrae, and image sequences seem to indicate that the back compresses and uncompresses during a golf swing, the result of using two rigid links to approximate the spine's curvature. The simplistic model presented here is sufficient for the golfers in the video sequences collected, however a more realistic model may be required for a broader selection of golfers.

3.1.2 Link Surfaces

Truncated elliptical conics have been used to 'flesh' out each of the links, similar to the approach used by Deutcher *et al.* [31]. These were chosen because they have straight sides, a convenient property for the graph based edge measurement function presented in Chapter 4. An unfortunate consequence of straight sides is that the quadric representation [28] is degenerate however, and so the elegant dual space projection approach can not be used.

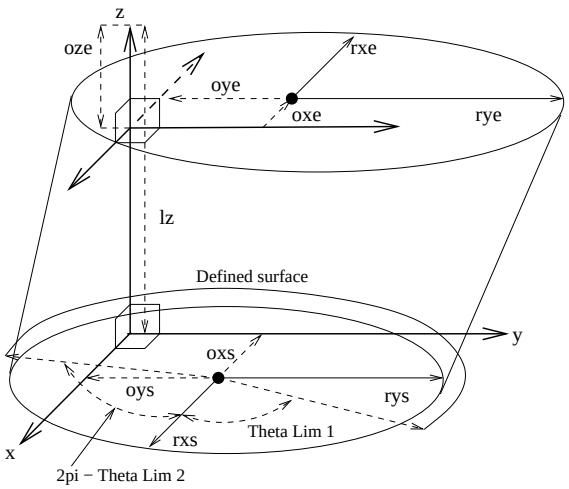


FIGURE 3.2: The truncated ellipse surface model

The end of each elliptical conic is an ellipse lying parallel to the x-y plane of the link's coordinate frame. The link's coordinate frame is defined with the link running along the z axis, the link's start at the origin, and the x and y axis chosen to give each ellipse the desired major and minor axis directions. The link has a *natural* transformation to rotate it into the world coordinate system. Each ellipse has radius $\{r_x, r_y\}$, and ellipse center offset $\{o_x, o_y, o_z\}^T$ measured from the applicable end of the link. The center offset $\{o_x, o_y, o_z\}^T$ was required to model parts where the bone is not in the center of the fleshed out region, as is most notable with the torso and spine. Theta limits $\{\theta_{l1}, \theta_{l2}\}$ are used to model parts with flat sides, such as the shoes. The ellipses's are only defined between these limits.

These parameters of the truncated elliptical conics are shown in Figure 3.2, however $o_z^s = 0$ was used for simplicity, where the superscript s denotes the start ellipse. As inferring the body model from visual data is beyond the scope of this thesis, it can be assumed these parameters were manually selected.

Calculating Occluding Contours

As mentioned above, a dual space projection methodology is not suitable for finding the straight edges and hence *occluding contour* of a truncated conic's projection. The projection of the world coordinate

of a point at θ on an ellipse, $\mathbf{r}(\theta) \in \mathbb{R}^2$, has the inhomogeneous form:

$$\begin{aligned} \mathbf{r}^*(\theta) &= \{\mathrm{x}^*(\theta), \mathrm{y}^*(\theta), \mathrm{z}^*(\theta)\}^T \\ \mathrm{x}^*(\theta) &= c_1^x \cos(\theta) + c_2^x \sin(\theta) + c_3^x \\ \mathrm{y}^*(\theta) &= c_1^y \cos(\theta) + c_2^y \sin(\theta) + c_3^y \\ \mathrm{z}^*(\theta) &= c_1^z \cos(\theta) + c_2^z \sin(\theta) + c_3^z \end{aligned} \tag{3.1}$$

with scalar c_i^j, $i \in \{1, 2, 3\}$ and $j \in \{x, y, z\}$. The homogenous form is then:

$$\begin{aligned} \mathbf{r}(\theta) &= \{\mathrm{x}(\theta), \mathrm{y}(\theta)\}^T \\ &= \left\{ \frac{\mathrm{x}^*(\theta)}{\mathrm{z}^*(\theta)}, \frac{\mathrm{y}^*(\theta)}{\mathrm{z}^*(\theta)} \right\}^T \end{aligned} \tag{3.2}$$

The projected ellipse's tangent direction, $\frac{d\mathbf{r}(\theta)}{d\theta}$, is then given by:

$$\begin{aligned} \frac{d\mathbf{r}(\theta)}{d\theta} &= \left\{ \frac{d\mathrm{x}(\theta)}{d\theta}, \frac{d\mathrm{y}(\theta)}{d\theta} \right\}^T \\ \frac{d\mathrm{x}(\theta)}{d\theta} &= \frac{1}{\mathrm{z}^*(\theta)^2} \left(\mathrm{z}^*(\theta) \frac{d\mathrm{x}^*(\theta)}{d\theta} - \mathrm{x}^*(\theta) \frac{d\mathrm{z}^*(\theta)}{d\theta} \right) \\ \frac{d\mathrm{y}(\theta)}{d\theta} &= \frac{1}{\mathrm{z}^*(\theta)^2} \left(\mathrm{z}^*(\theta) \frac{d\mathrm{y}^*(\theta)}{d\theta} - \mathrm{y}^*(\theta) \frac{d\mathrm{z}^*(\theta)}{d\theta} \right) \end{aligned} \tag{3.3}$$

which can be written more concisely as:

$$\frac{d\mathrm{x}(\theta)}{d\theta} = \frac{c_1^{dx} \cos(\theta) + c_2^{dx} \sin(\theta) + c_3^{dx}}{\mathrm{z}^*(\theta)^2} \tag{3.4}$$

$$\frac{d\mathrm{y}(\theta)}{d\theta} = \frac{c_1^{dy} \cos(\theta) + c_2^{dy} \sin(\theta) + c_3^{dy}}{\mathrm{z}^*(\theta)^2} \tag{3.5}$$

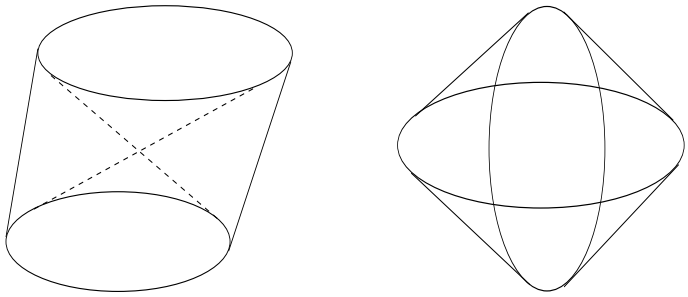
with scalar c_i^j, $i \in \{1, 2, 3\}$ and $j \in \{dx, dy\}$.

Theta values at the edges are points on the start and end ellipse whose tangents are collinear. Using the subscripts s and e to denote the start and end ellipses, this can be expressed by the equations:

$$\mathbf{0} = \left\langle \frac{d\mathbf{r}_s(\theta_s)}{d\theta_s} \times \frac{d\mathbf{r}_e(\theta_e)}{d\theta_e} \right\rangle_z \tag{3.6}$$

$$\mathbf{0} = \left\langle (\mathbf{r}_s(\theta_s) - \mathbf{r}_e(\theta_e)) \times \frac{d\mathbf{r}_s(\theta_s)}{d\theta_s} \right\rangle_z \tag{3.7}$$

where $\langle \times \rangle_z$ denotes the z component of the vector cross product, *i.e.* $\langle \mathbf{r}_s(\theta_s) \times \mathbf{r}_e(\theta_e) \rangle_z = x_s(\theta_s) y_e(\theta_e) - y_s(\theta_s) x_e(\theta_e)$. Using equations 3.6 and 3.7, θ_s can be calculated as the roots of a 4th order polynomial. These solutions are generally not the points $\mathbf{R}(\theta) \in \mathbb{R}^3$ whose tangent plane (orthogonal to the ellipse plane) contains the camera. These tangent plane solution are only valid when the camera and the ellipse lie in the same plane. This can be shown visually using Figure 3.3(a), and noting that the top ellipse's edge theta values will change if the bottom ellipse is translated to the right.



(a) Two Valid Edges (b) Four Valid Edges

FIGURE 3.3: Four real edge solutions examples

Figure 3.3 shows two examples where there are four real solutions to this polynomial. In Figure 3.3(a) the two edge solutions that intersect each other can be discarded, or more generally any edge solution which intersects another edge solution can be discarded. Because the start and end ellipses are parameterized in the same way, $|\theta_s - \theta_e| \gg 0$ for invalid solutions.

Figure 3.3(b) presents a more problematic case where all four edges are valid. This case generally occurs when the camera lies in the direction of the link's z axis, particularly when the start and end ellipses have significantly different eccentricities. In this case the edge solution i with maximum length is used, and is paired with the solution j with the next largest length and satisfying:

$$\text{sign}\left(\frac{d\mathbf{r}_s(\theta_s^i)}{d\theta_s} \cdot \left(\mathbf{r}_e(\theta_e^i) - \mathbf{r}_s(\theta_s^i)\right)\right) \neq \text{sign}\left(\frac{d\mathbf{r}_s(\theta_s^j)}{d\theta_s} \cdot \left(\mathbf{r}_e(\theta_e^j) - \mathbf{r}_s(\theta_s^j)\right)\right) \tag{3.8}$$

where $\cdot$ denotes the dot product. The other edge solutions are simply discarded. The condition given in Equation 3.8 guarantees a simple shape. Generally two pairs of edge solutions are found using this method, $\{\theta_s^{E1}, \theta_e^{E1}\}$, and $\{\theta_s^{E2}, \theta_e^{E2}\}$. In some cases however there are no real solutions to equations 3.6 and 3.7, which means the projection of one end's ellipse is entirely contained within the projection of other end's ellipse. In this case, the occluding contour is simply the larger of the projected ellipses.

When surface theta limits, previously discussed in Section 3.1.2, are used and an edge solution is found in the non-defined section of the ellipse, the solution is mapped back to the appropriate theta limit. Surface limits can produce cases where there are valid edges, despite there being no real solutions to equations 3.6 and 3.7. In this case one end's ellipse is entirely contained within the projection of other end's ellipse, but not entirely contained within the projection of the 'truncated' ellipse. In this case the edge theta values are simply the ellipse surface limits.

3.1.3 Self Occlusion Model

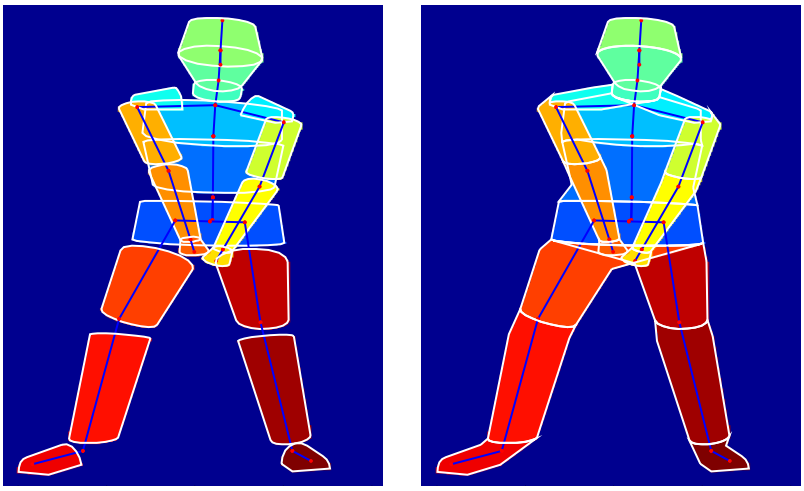
In the formulation of the self occlusion model used throughout this thesis, a pixel on a link's occluding contour is considered to be occluded if:

- it lies within the occluding contour of another link which is closer to the camera,
- or it is within five pixels of another pixel 'belonging' to another link in the same colour group as the current link.

Each colour group is the set of links expected to be the same colour. This second criteria is then used as an edge feature is not expected in this case as the pixels are expected to be the same colour. The colour groups are assumed to be known a priori, however no expectation of the actual colour of the group is assumed.

A comprehensive self occlusion map has been used, where the occluding link number for each pixel in the image (for the current model state) is stored. This comprehensive map facilitates calculating a self occlusion penalty score, and the region consistency scores discussed later Section 3.5.2. Since the only source of occlusions modelled in this thesis are self occlusions, the self occlusion map is referred to as the occlusion map.

Conic link surface models generally model only limited sections (lengthwise) of the body part of interest. This is so different link's edges do not intersect in an anatomically plausible configuration. As such a joining process is used to remove these 'holes' from the occlusion map.



(a) Occlusion Map (b) Joined Occlusion Map

FIGURE 3.4: Occlusions Maps

Figure 3.4 shows an example of the occlusion map before and after joining. It can be seen that if joining was not performed, it would be erroneously expected that there were small visible edge regions of the lower torso due to the gap between the upper and lower arms shown in Figure 3.4(a).

Joining Occluding Contours

The joining process shown in Figure 3.4(b) is surprisingly non-trivial. While formulating rules for joining for a given object state is simple, care must be taken to ensure that a small change in the

object's state does not cause a significant change in the occlusion map. Chaotic behavior in the joining function may propagate through the measurement function. While it is intuitive to formulate the joining process as a series of case statements, it is important to ensure a smooth transition between cases. To ensure these smooth transitions the joining process used here obeys two rules:

- The added join section does not contain any points within the occluding contour it is being joined to.
- All points contained in the original occluding contour are also contained in the occluding contour after joining.

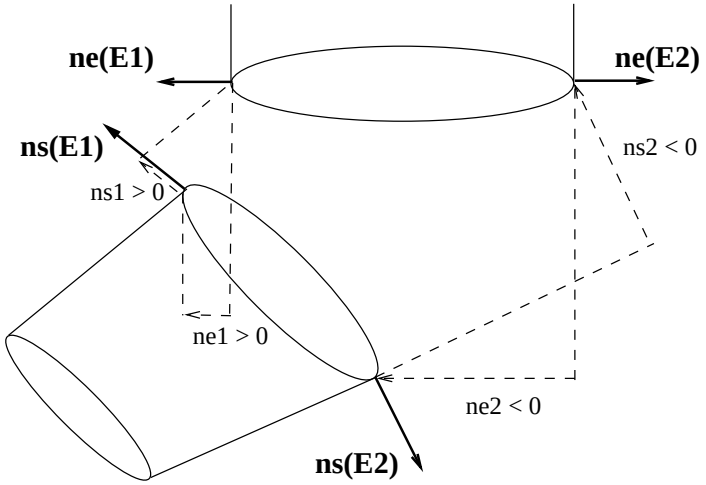


FIGURE 3.5: Link Joining Example

The joining process is performed after the projection of the links' surfaces. Considering the case of joining the start of link A to the end of link B, and following the notation used in Section 3.1.2 where $\mathbf{r}_s(\theta_s) \in \mathbb{R}^2$ denotes the image coordinate for values of θ on the start ellipse. For clarity an example situation is given in Figure 3.5. The outfacing normal direction is:

$$\mathbf{n}_s(\theta_s) = \varsigma \left\{ -\frac{d\mathrm{y}_s(\theta_s)}{d\theta_s}, \frac{d\mathrm{x}_s(\theta_s)}{d\theta_s} \right\}^T \tag{3.9}$$

where $\varsigma \in \{-1, 1\}$ is chosen to give an outfacing normal:

$$\mathbf{n}_s(\theta_s) \cdot \frac{d^2\mathbf{r}_s(\theta_s)}{d\theta_s^2} < 0 \tag{3.10}$$

with $\cdot$ denoting the dot product. Given theta values for the two edges at the start of link A and the end of link B, $\boldsymbol{\theta}_s^E = \{\theta_s^{E1}, \theta_s^{E2}\}$ and $\boldsymbol{\theta}_e^E = \{\theta_e^{E1}, \theta_e^{E2}\}$, a simplified algorithm for computing theta values at the joins, $\boldsymbol{\theta}_s^J$ and $\boldsymbol{\theta}_e^J$, is given in Algorithm 4.

Algorithm 4 is simplified as it does not address cases where:

for *the two edges* E_i, $i \in \{1, 2\}$ **do**

1) Calculate the projection distance of the edge points along the ellipse normals:

$$n_s^i = \mathbf{n}_s(\theta_s^{Ei}) \cdot \left(\mathbf{r}_e(\theta_e^{Ei}) - \mathbf{r}_s(\theta_s^{Ei})\right)$$

$$n_e^i = \mathbf{n}_e(\theta_e^{Ei}) \cdot \left(\mathbf{r}_s(\theta_s^{Ei}) - \mathbf{r}_e(\theta_e^{Ei})\right)$$

2) **switch** (n_s^i, n_e^i) **do**

case $(n_s^i < 0)$ *and* $(n_e^i < 0)$
Set θ_s^{Ji} and θ_e^{Ji} to the first collinear tangent solution after θ_s^{Ei}, where collinear tangent solutions are found using the same method for finding edge locations described in section 3.1.2.

case $(n_s^i \geq 0)$ *and* $(n_e^i \geq 0)$
Set $\theta_s^{Ji} = \theta_s^{Ei}$, and $\theta_e^{Ji} = \theta_e^{Ei}$

case $(n_s^i \geq 0)$
Set $\theta_s^{Ji} = \theta_s^{Ei}$, and θ_e^{Ji} such that $\mathbf{n}_e(\theta_e^{Ji}) \cdot \left(\mathbf{r}_s(\theta_s^{Ei}) - \mathbf{r}_e(\theta_e^{Ji})\right) = 0$

case $(n_e^i \geq 0)$
Set $\theta_e^{Ji} = \theta_e^{Ei}$, and θ_s^{Ji} such that $\mathbf{n}_s(\theta_s^{Ji}) \cdot \left(\mathbf{r}_e(\theta_e^{Ei}) - \mathbf{r}_s(\theta_s^{Ji})\right) = 0$

end

end

Algorithm 4: Simplified Joining Algorithm

- the ends of the two link's intersect[1].
- one of links has surface theta limits.

Another note is that the condition $(n_s^i < 0)$ and $(n_e^i < 0)$ does not guarantee the existence of a collinear tangent solution, one of many 'quirky' cases. In hindsight using a deformable 3D shape to join the link's surfaces should have been investigated.

Calculating the Occlusion Map

To calculate the occlusion map, vertical scan lines were intersected with the joined occluding contour for each link. A link order (ascending) was created based on the link's centroid's distance from the camera. The order was iterated through and pixels between the intersection points and not already assigned to another link, were assigned to the current link. For concave occluding contours each scan line has at most two intercepts, however convex shape may have more (but always an even number). This method was found to produce a good balance between accuracy and computational efficiency. Vertical scan lines were used because the image is stored with each column of pixels contiguous in memory.

[1]The solution to projected ellipse intersections can be formulated as the roots of a 4^{th} order polynomial, which is most easily derived in $\mathbb{R}^3$ by solving for collinear lines from the camera to points on each ellipse. An example where there are 4 intersection points was shown in Figure 3.3(b).

Occlusion Likelihoods

It is often advantageous to bias the measurement function towards states which have more assigned pixels in the occlusion map. An example of this is when propagating a state where one leg occludes the other. As the legs begin to move away from each other, it is desirable that the measurement function makes states that follow this separation more likely than states that remain 'locked' onto a single leg. As such an occlusion likelihood $p(O|X)$ is used:

$$\log\left(p(O|X)\right) = \frac{-\alpha}{\sum_{i=0}^{C}\sqrt{n_i}} \tag{3.11}$$

where C is the number of cameras, n_i is the number of occluding pixels for camera i, and α is a scaling term. This formulation was chosen empirically, with $\alpha \approx 100$ found to be a good choice.

Occluding Contour Calculation Discussion

In Sections 3.1.2 and 3.1.3, rigorous (computationally intensive) methods were used to calculate each link's occluding contour and then to join these contours. Alternatively, approximate methods can be used to calculate these. Smith and Lovell [95] calculated edge positions by generating sample points, $\mathbf{R}(\theta) \in \mathbb{R}^3$, on each link's end ellipses and testing to find which sample points were edges. Each link's occluding contours were joined by simply joining these edge locations where possible.

In Section 4.8.5 an experiment testing the number of local maxima in the observational likelihood for different edge measurement methods, performed by Smith and Lovell [95] is repeated. The results in this thesis show a marked reduction in the number of modes for all measurement schemes, compared to the results obtained by Smith and Lovell. This marked reduction was the result of using the more precise link projection and link joining methods described in this chapter. The author was surprised at how much reworking these two areas improved the modality of the observational likelihoods. This effect is probably more pronounced in this experiment, where the subject is quite large in the image, as the cameras are close to the subject.

3.2 Camera Model

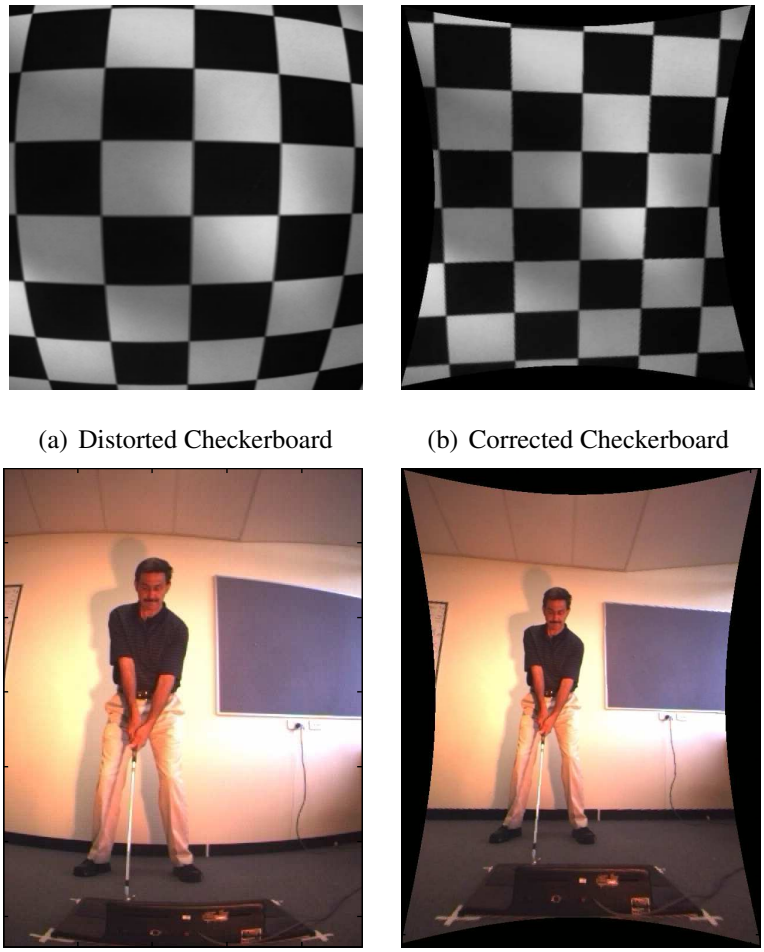
The video sequences examined in this thesis were captured using 1–2 Dragonfly cameras from Point Grey Research [2] , that synchronously capture 640×480 colour images at 30 frames per second.

Since it is desirable to keep the system as small as possible, low focal length (high field of view) lenses were used so the cameras can be placed as closely as possible to the golfer. This introduced significant radial distortion. To compensate for this, a transformation from the distorted image to an

Model Order	L2 Norm Error	CAIC Value
0	1695.114	∞
1	84.9189	101.2427
2	5.4762	26.6286
3	5.3464	31.3275
4	5.2819	36.0917
5	5.2782	40.9165
6	5.2770	45.7441

Table 3.1: CAIC values for various approximation orders

undistorted image was learnt using a technique described by Hartley and Zisserman [45], where Taylor series coefficients are chosen to make real world straight lines straight in the image. Consistent Akaike's Information Criteria (CAIC) [15] was used to choose the order of the Taylor series approximation. The results are shown in Table 3.1, where a second order model was chosen as it has the smallest CAIC value.



(a) Distorted Checkerboard (b) Corrected Checkerboard

(c) Original Image (d) Corrected Image

FIGURE 3.6: Radial Distortion Correction

The Taylor series models a transform from the undistorted image to the distorted image, however

the inverse of this transform is of more practical use. As such, a map of the transform's inverse was calculated, where each entry contains the index of the undistorted pixel in the distorted image. The Taylor series is inverted for each pixel in the undistorted image using a local optimization process.

Figures 3.6(a) and 3.6(c) show examples of radial distortions, while Figures 3.6(b) and 3.6(d) show the same images after correction. Note that the checkerboard grid lines are straight after the correction. After radial distortion correction, the camera projection matrices were learnt from real world and image point correspondences, using the DLT algorithm with non-linear optimization described by Hartley and Zisserman [45].

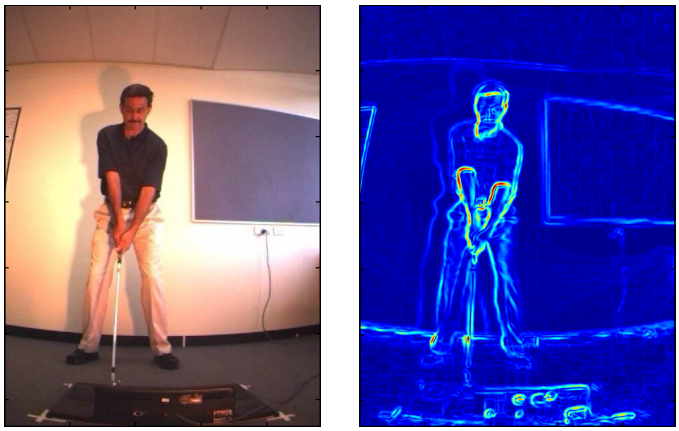
3.3 Image Gradient Metric and Edge Detection

As colour cameras were used for video capture, a combined colour and intensity based gradient metric was used. Colour based gradients are more robust to the effects of shadowing than intensity based gradients, but are susceptible to noise in dark regions of the image. The combined gradient is simply a linear blending of the intensity and colour gradient, where the colour gradient is calculated from the magnitude normalized image $I_n(x, y)$, where:

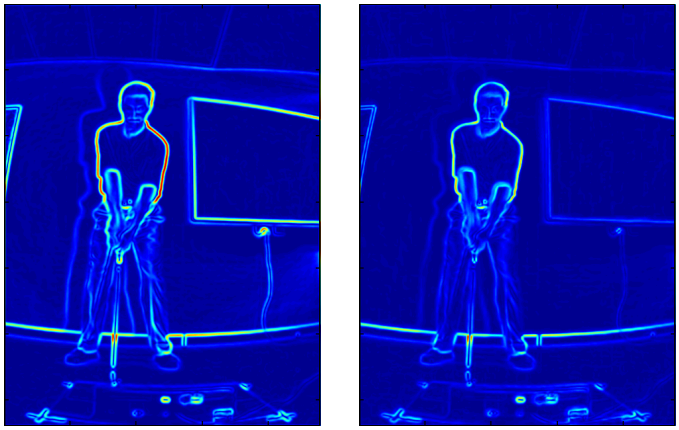
$$\begin{aligned} c^2 &= I_n^R(x,y)^2 + I_n^G(x,y)^2 + I_n^B(x,y)^2 \\ c^2 &= \frac{3}{2} \times 255^2 \end{aligned} \tag{3.12}$$

Here the constant c was chosen such that the maximum possible intensity gradient ($\sqrt{3} \times 255$) equalled the maximum possible colour gradient. The RGB colour space was used for the colour gradient rather than a perceptual colour space for simplicity. For operations where the sign of the x and y gradient is important, such as edge *thinning*, the colour gradient was given the same sign as the intensity gradient. The blending function was set such that a pixel $\|I(x, y)\| = 0$ uses 90% intensity gradient and a pixel $\|I(x, y)\| = \sqrt{3} \times 255$ uses 5% intensity gradient. The image was blurred using a Gaussian kernel before gradient and normalization calculation.

Figure 3.7 shows the results of the intensity, colour, and combined gradient schemes. While the intensity gradient generally outperforms the colour gradient, the colour gradient is more robust to the shadowed regions around the legs. While it could be expected that the colour gradient would be invariant to shadow effects, the shadowing observed here is the shadowing from one or more of the multiple light sources. Each light source skews the pixel distribution towards its own colour, and so when a point is shadowed both its intensity and colour changes, with the colour changing to a smaller extent. This effect is discussed further in Section 3.5.1.

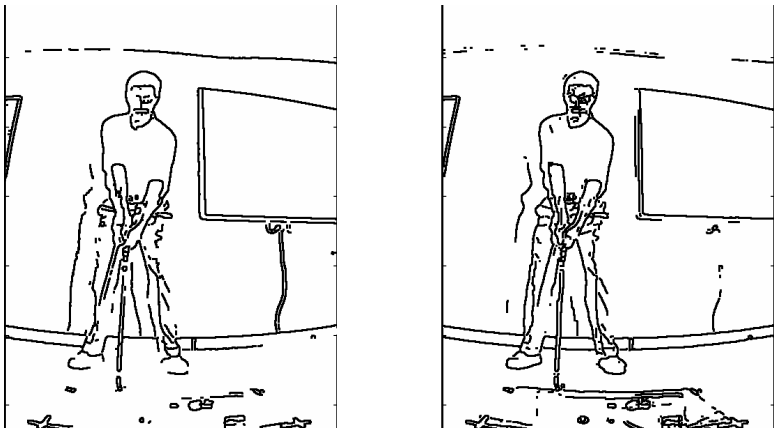


(a) Original Image (b) Colour Gradient



(c) Intensity Gradient (d) Combined Gradient

FIGURE 3.7: Different gradient schemes



(a) Intensity Edges (b) Combined Edges

FIGURE 3.8: Edge Maps using different gradient schemes

The combined gradient is an attempt to get the best of both the intensity and colour gradients. However, neither method performs well in distinguishing the outside of the left leg from the background. Figure 3.8 shows the edge maps produced using the intensity and combined methods.

3.4 Learning feature models along measurement lines

Given a training set, it is possible to learn a measurement model by examining the feature set $\boldsymbol{\mathcal{E}}$, which is comprised of all features on all measurement lines for all frames in the training set. Each measurement line is cast at a normal to the object's surface in its hand labelled position (see Section 2.1.2). Described here is a treatment for learning the Poisson measurement process discussed by MacCormick [64]. The Poisson measurement process uses variables λ, $\hat{\mathbf{r}}$, $\boldsymbol{\Sigma}$, and $\mathbf{q}_{01}$. A description of these variables is given in Table 3.2.

Variable	Description
λ	The background feature density in features per pixel.
$\hat{\mathbf{r}}$	The set of expected feature locations on a measurement line.
$\boldsymbol{\Sigma}$	The set of variances associated with the expected feature locations.
$\mathbf{q}_{01}$	The set of non-detection probabilities of the expected feature locations – the probability the measurement line crossed the true feature (generally true object boundary) however a feature was not detected.

Table 3.2: Variables used in the Poisson model for edge evaluation

To determine λ, $\hat{\mathbf{r}}$, $\boldsymbol{\Sigma}$, and $\mathbf{q}_{01}$, a histogram of the feature set $\boldsymbol{\mathcal{E}}$ is formed. λ is determined by examining the feature density at distances along the measurement line further than an edge feature could possibly lie. This background feature density is used to calculate the number of noise driven features expected in each histogram bin.

To determine expected feature locations the histogram bin frequencies are first smoothed to reduce the number of local modes. A K-means Gaussian fitting algorithm [79] is used to solve for both $\hat{\mathbf{r}}$ and $\boldsymbol{\Sigma}$, seeded using local maxima in the smoothed histogram whose values are more than twice the expected noise driven features. The number of measurement lines which did not contain a feature within 3 standard deviations of each expected feature locations is used to determine $\mathbf{q}_{01}$. An example of this process is shown in Figure 3.9.

Figure 3.9 is a feature histogram learnt from a training set tracking a golfer's arms (38 frames, 722 measurement lines), where the arms were modelled by a B-Spline. The red line shows the learnt feature distribution. The maximum at 0 pixels is the external edge of the golfer's arms, while the local maximum at 10 pixels corresponds to the interior edge of the golfer's arms. The reason for the poor fit of the Gaussian mixture is that there are a larger number of features than expected in the 15 to 25

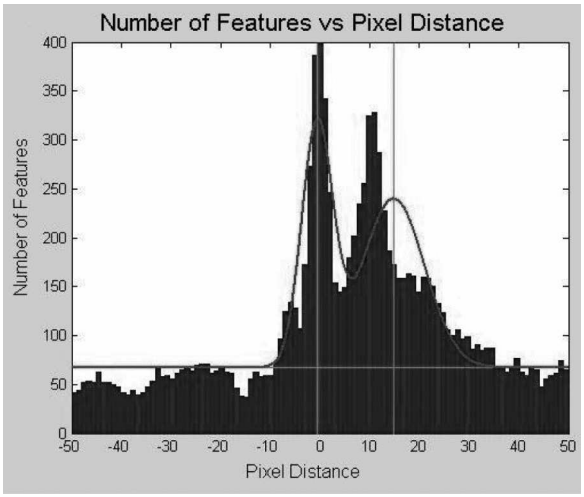


FIGURE 3.9: Learning Feature Targets

pixel range, corresponding to where the measurement line crosses the other arm in some frames of the training data. In this thesis, learnt feature models are only used in comparisons to existing edge measurement methods. Only the primary Gaussian mixture is used in these cases.

3.5 Region Based Measurement

Integrating multiple measurement cues can significantly reduce the number of local maxima in the observational likelihood. The high noise levels in images reduces the reliability of any individual measurement scheme. As such a region based measurement, as described in Section 2.1.4, is used to complement the edge measurement used in many of the experiments presented in this thesis.

3.5.1 Discussion

While background subtraction methods are a popular region based measurement method [10, 33, 86, 87, 99], they require assumptions about the nature of the background, *i.e.,* that it is static. Due to the intended application of this research work, it was decided that this assumption is too restrictive. Optical flow is another popular approach to performing region based measurements [18, 21, 52, 82, 99, 103, 108], however it was again determined that the requirement of registered object templates was too restrictive.

The 'Image Flow Correspondence Field' used by Sminchisescu and Triggs [90] is a promising approach as it does not require object templates. In practice however, optical flow techniques[2] proved unreliable on the video sequences collected during this thesis. This is due to the highly dynamic

[2]Thank you to Michael Black for providing optical flow code.

motion and the close proximity of the cameras to the subject. The highly dynamic motion necessitated the use of a fast shutter speeds (short exposure times) to limit motion blurring, and the cameras discussed in Section 3.2 are not specialized high speed cameras. As such multiple lighting sources were used to sufficiently illuminate the scene. Each lighting source skews the colour distribution towards the lighting source's colour, *i.e.* a red light will make the image look more red. The highly dynamic motion means objects travel significant distances between frames, and so the illumination contributions from each lighting source vary from frame to frame. Shadowing is the greatest source of this variation, where the illumination from a particular light source is removed between frames. It is then very difficult to match regions of consecutive images, as the statistical properties of the region have changed. Compounding this is the large search distances required in the optical flow algorithm. The result of this was the optical flow algorithm falsely designating impractically large image regions as 'outliers'. These effects also significantly reduced the performance of background subtraction methods.

3.5.2 Region Consistency Measurement

As it was desired to complement the edge measurement with a region based cue, a *region consistency* measure was used to provide a region based measurement. The region consistency measure is based on the assumption that the object can be broken into several colour groups, and that each colour group's pixel distribution can be parameterized by a single mean and covariance. In the case of articulated human tracking, these colour groups are taken to be the: skin, shirt, pants, and shoes. The head is not used in the skin colour group because the presence of hair makes the colour group's pixel distribution bi-modal. Letting μ_i and Σ_i designate the mean and covariance for colour group i across all cameras, the region consistency likelihood $p(R|X)$ is:

$$\log\left(p(R|X)\right) \propto \sum_{i=1}^{n} \sqrt{\frac{|\Sigma_i|}{\|\mu_i\|^6}} \tag{3.13}$$

where n is the number of colour groups, and $|.|$ denotes the determinant. The determinant's root $\sqrt{|\Sigma|}$ is used as it is proportional to the volume of a confidence interval for the distribution. The denominator $\|\mu\|^6$ is used as $|\mathrm{Cov}\left(\mathrm{X}\right)| = \frac{|\mathrm{Cov(sX)}|}{s^6}$ for a three dimensional data set X, although practically it is useful to set a minimum value for the mean magnitude, $\max\left(\|\mu\|, \sim 100\right)$. The set of pixels belonging to each link and hence each colour group can be efficiently extracted from the occlusion map, which was shown in Figure 3.4(b).

This region consistency measure does not require the mean or covariance for the colour groups to be known apriori, and does not enforce consistency between consecutive frames. Its performance

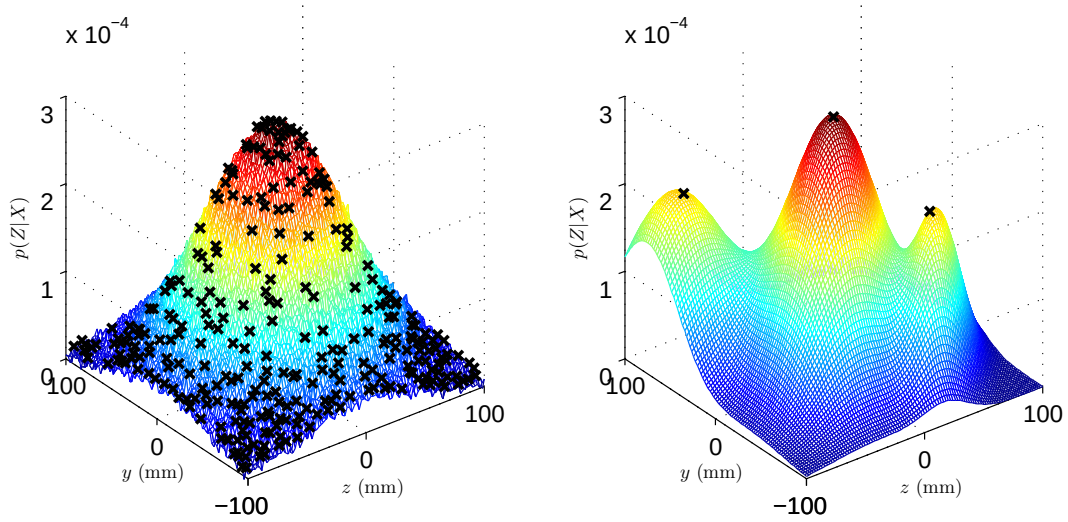
is evaluated later in Section 4.8.5, and was been found to provide a good measurement despite the issues discussed in Section 3.5.1. The uni-modal colour distribution assumption is limiting however, as it is not suitable for textured a colour group, such as occurs with a chequered shirt. In this case a spatial autocorrelation function for the colour group could be investigated.

4

Measurement Function Design

The desirable properties of edge measurement functions are discussed in this chapter. Graph based approaches commonly used in image segmentation problems are presented as an alternative to the edge measurement functions commonly used by other authors. This graph based method is proposed because it does not rely on extracting a set of edge features from the image, such as found by using edge detection; and also because it naturally uses a dense set of measured points, with consistency between measurements guaranteed by the edge directedness of the graph. Firstly, a method is given to formulate a graph to match a hypothesized edge position for a link in the kinematic chain. This method is then extended so that a single graph is formulated around the occluding contour(s) of an object. As self occlusions are not generally modelled in image segmentation problems, methods are given for calculating the shortest path in the presence of occluded graph vertices. Experimental evaluation shows that the proposed graph based approach has more desirable properties than other edge measurement methods, in terms of both its accuracy, and the number and size of modes in the observational likelihood. A treatise on using the graph based approach to model deformable surfaces, such as caused by loose fitting clothing, is also given.

4.1 What Constitutes a Good Measurement Function?



(a) Gaussian with random noise, 586 Modes (b) Mixture of three Gaussians, 3 Modes

FIGURE 4.1: Different optimization surfaces

Figure 4.1 reexamines two synthetically generated surfaces, where the first is a Gaussian perturbed by random noise, and the second is a mixture of three Gaussians. From visual inspection, the surface shown in Figure 4.1(b) would be suitably searched by a smooth transition state approach such as was described in Section 2.4.6. A smooth transition state approach would not be perform well for the surface shown in Figure 4.1(a) however, which would be most suited to an annealing style optimization such as was described in Section 2.4.4.

Examining both cost functions shown in Figure 4.1, it is difficult to judge if one is inherently more difficult to search than the other. While the Gaussian mixture has fewer local maxima, they are dispersed widely across the search region. A search algorithm could exploit the knowledge that the random noise cost function can be made smoother by smoothing with a suitable moving average filter. In essence the weighting function $\mathbf{w}(.)$ used in the annealed particle filter from Section 2.4.4 performs this smoothing. Ideally a quantitative measure of the *hardness* of the each surface could be determined, a continuation of the work of MacReady and Wolpert [67]. What is clear however is that the properties of the surface to be searched determine the optimal search strategy. From this it follows that an ideal measurement function will satisfy the following conditions:

1. The measurement function provides a close representation of the true model state (a small Kullback-Leibler divergence [27]).

2. It consistently produces likelihoods of the same class.

3. The likelihoods are as ‘easily’ searched as feasible.

Two likelihoods are defined as belonging to the same class if the same search algorithm performs "well" on both of them. The combination of these three conditions allows the problem to be solved robustly.

Implementing this practically means combining measurement methods to produce a likelihood which accurately reflects the true model state, and has predetermined topological characteristics. The measurement methods chosen must be reliable so that in all real environments they will neither fail in such a way as to change the class of the observational likelihood, nor cause the likelihood to no longer be representative of the true model state. In the authors opinion smoother surfaces are more readily searched, and so the proposed measurement function is aimed at producing likelihoods with few clustered local modes.

4.2 Graph Based Methods for Edge Measurement

The graph based edge measurement methodology developed in this chapter constructs graphs around hypothesized edge locations, with the edge directedness of the graphs matching the hypothesized edge directions. The measure of match between the image and the hypothesized edge location is a function of the minimum cost of traversing the graph, *i.e.* the *shortest path*.

As mentioned in Section 2.1.3, graph based methods have been used successfully for a variety of image segmentation tasks [4–7, 11, 19, 20, 41]. In these applications, the object to be segmented is typically much simpler than the highly articulated models used for human body tracking, and so a single graph can be constructed such that the set of possible paths across the graph is equivalent to the set possible object appearances. These approaches are then deterministic, in that they the graph vertex locations are not dependent upon a hypothesized object configuration.

Due to the complexity of the articulated models used for human motion tracking, formulating a graph such that each potential path across it corresponds to a (or space of) potential human postures is not tractable. Tracking algorithms cannot assume good object/image contrast in all areas, and so the possible paths across the graph must be restricted to paths which match the space of the object's image appearance. In the formulation given in this chapter, the graph's vertex locations are functions of a hypothesized object state, which allows the potential paths across the graph to be constrained to match the object's hypothesized appearance. This approach is then suitable for generative trackers.

In principle it may be possible to solve for the shortest path across a more general graph, but then to discard this path if it does not match the space of the object's image appearance, such as learnt via techniques such as discussed in Section 2.5.1. The next shortest path could then be calculated if the shortest path was discarded, until a path matching the object's image appearance was found. In this

way distinct object state hypotheses could be generated directly from an image.

4.2.1 Motivation

Edge measurement methods based on line searches or nearest edge approaches, discussed in Sections 2.1.2 and 2.1.2, are inherently limited by their reliance on edge detection as a preprocessing step. This is highly problematic in images where the object of interest does not contrast well with the background. Any robust tracking algorithm should be able to successfully track in low contrast situations. Sminchisescu and Triggs [93] comment of the disturbingly large number of local modes in their representation of the observational likelihood, which uses a line search edge measurement. It is hypothesized that an edge measurement scheme which does not rely on success edge extraction could significantly reduce the observational likelihood's modality.

Graph based approaches are not reliant upon the classification of each pixel in the image as either edge or not edge. The cost of traversing each vertex is given by a smooth function of the image gradient at the vertex's location. This is analogous to considering the probability that an image pixel is an edge. It is expected that the graph based path approach should perform significantly better in the presence of low object/background contrast. Furthermore, the vertex placement can be used to densely sample the region around the object's occluding contour. This makes it invariant to the problems associated with sample positions and missed edge features discussed in Section 2.3.4. The edge directedness of the graph also ensures the association between edge features in local regions. This alleviates the need for probabilistic interpretation of the feature distribution between different measurement lines, such as the Markov Random Walk approach [64].

The graph based approach is computationally more expensive than line searches or nearest edge approaches. Calculating the shortest cost path across a graph by dynamic programming has complexity $O(uv)$ [22], where u and v are the number of rows and columns of the trellis. Additionally, storing graph vertex location can be somewhat memory intensive. However, the most difficult aspect of tracking problems is the global search and optimization, and improving the conditioning of the underlying likelihoods can greatly decrease the number of measurement function evaluations required during this search.

The cost of the shortest path also does not have a direct probabilistic interpretation. While a distribution of shortest path costs could be learnt from training data, it would not be applicable to images where the object/background contrast is significantly different from the training data. However, most tracking approaches perform the global search and optimization on a cost surface representative of the likelihoods, and most resampling operations are also performed upon a representation of the

likelihood, are so direct probabilistic interpretation is often unnecessary.

4.3 Trellis Formulation

In the approach presented in this Chapter, trellises are formed around the hypothesized edge location of each link in the kinematic chain. Using trellises rather than more general graphs allows for the efficient computation of the shortest path by dynamic programming, as given in Algorithm 1.

4.3.1 Link Edge (rectangular) Trellis Formulation

From the link surface model described in Section 3.1.2, each link's sides (edges) are straight lines in the image. Each row i of the corresponding trellis runs parallel to the edge's direction, and each column j is orthogonal to the edge's direction. Using θ_s^E to denote an edge theta value for the start ellipse (as per Section 3.1.2), the normalized row direction is given by:

$$dV = \varsigma \frac{\frac{d\mathbf{r}_s(\theta_s^E)}{d\theta}}{\| \frac{d\mathbf{r}_s(\theta_s^E)}{d\theta} \|} \tag{4.1}$$

where $\frac{d\mathbf{r}_s(\theta)}{d\theta}$ was defined in Equation 3.3, and $\varsigma \in \{-1, 1\}$ is chosen to ensure the rows runs in the same direction as the corresponding edge,

$$\Delta\mathbf{r} = \mathbf{r}_e(\theta_e^E) - \mathbf{r}_s(\theta_s^E) \tag{4.2}$$

$$0 < dV \cdot \Delta\mathbf{r} \tag{4.3}$$

where $\mathbf{r}_s(\theta)$ was defined in Equation 3.2, and $\cdot$ denotes the dot product. Ideally the trellis vertices are spaced 1 pixel apart on the rows and columns, however due to non-integer edge lengths the number of trellis columns, v, is set to $v = 1 + \text{round}\,(\|\Delta\mathbf{r}\|)$. The distance between trellis columns is then $v_{gap} = \frac{v-1}{\|\Delta\mathbf{r}\|}$. Using the edge normal direction, $\mathbf{n}_s(\theta_s^E)$ defined in Equation 3.9, the unitized trellis column direction is given by:

$$nV = \frac{\mathbf{n}(\theta_s^E)}{\|\mathbf{n}(\theta_s^E)\|} \tag{4.4}$$

The image location of the jth vertex on the ith row of the trellis is then given by:

$$V(i,j) = \mathbf{r}(\theta_s^E) + i \times nV + (j-1) \times v_{gap} \times dV \tag{4.5}$$

assuming the edge row is row $i = 0$ for simplicity. This rectangular trellis formulation allows for efficient computation of the image locations of the trellis's vertices.

Figure 4.2 shows an example of a trellis formed around the interior edge of a golfer's lower arm. Using this trellis formulation, the trellis's vertex locations are dependent upon the hypothesized model

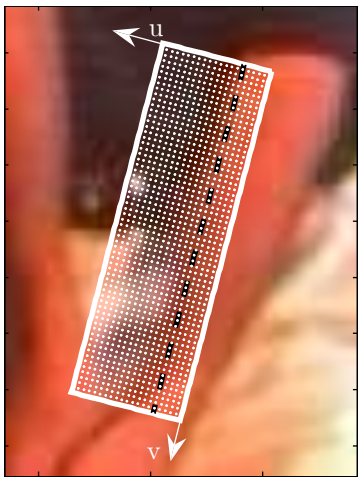


FIGURE 4.2: Example rectangular trellis

state[1]. This dependence ensures that the set of possible paths for a hypothesized object state reflects the geometric properties of the object model. This separates this approach from the minimal surface segmentation approach proposed by Appleton and Talbot [7], which can not enforce these geometric properties. Forcing the set of possible paths to match object's geometric properties is essential to resolving ambiguous image data (low object-background contrast).

4.3.2 Link End (curved) Trellis Formulation

For links at the end of a branch of the kinematic chain, a trellis is formed around the projection of the ellipse describing each link's end. The importance of performing measurements around these links ends can been seen by referring back to the experiment performed in Section 2.3.4. The edge row of each link's end trellis follows the ellipse's projection from $\theta_1 = \theta^{E1}$ to $\theta_v = \theta^{E2}$, where θ^{E1} and θ^{E2} are the ellipse's edge theta values as discussed in Section 3.1.2. Vertices are placed on the edge row such that the arc length, $\mathcal{A}$, between vertices is approximately one pixel. Equation 3.2 showed the image coordinates of a projected ellipse, $\mathbf{r}(\theta) = \{\mathrm{x}(\theta), \mathrm{y}(\theta)\}^T$, expressed as a function of θ. The jth vertex on the edge row is placed such that:

$$\mathcal{A}(\theta_j, \theta_{j-1}) = \int_{\theta_{j-1}}^{\theta_j} \sqrt{\frac{d\mathrm{x}(\theta)}{d\theta}^2 + \frac{d\mathrm{y}(\theta)}{d\theta}^2}\, d\theta \approx 1 \tag{4.6}$$

where given the rather daunting form of $\frac{d\mathrm{x}(\theta)}{d\theta}$ and $\frac{d\mathrm{y}(\theta)}{d\theta}$ given in Equations 3.4 and Equations 3.5[2], θ_j is found by modifying a numeric integration method. There is little need for accuracy when calculating the vertex locations, and in the implementation used to generate later results, θ_j is solved such that

[1] Hence the measurement function represents $p(Z|X)$ rather than $p(X|Z)$

[2] $\mathcal{A}(\theta_j, \theta_{j-1})$ reduces to an elliptic integral, $\int \frac{p^2(s)}{p^4(s)} \sqrt{p^4(s)} ds$, where $p^i(s)$ is an ith order polynomial.

the arc length between θ_j and θ_{j-1} is within $1 \pm \frac{1}{2}$ pixels. When trellis joining, discussed later in Section 4.4.7, is performed, the vertex locations are recalculated using a linear interpolation. Trellis columns are formed by casting curve normals, $\mathbf{n}(\theta)$ from Equation 3.9, to the curve at the points $\{\theta_1, \theta_2, ..., \theta_v\}$, where each row is separated by 1 pixel along these normals. The column direction corresponding to θ_j is then:

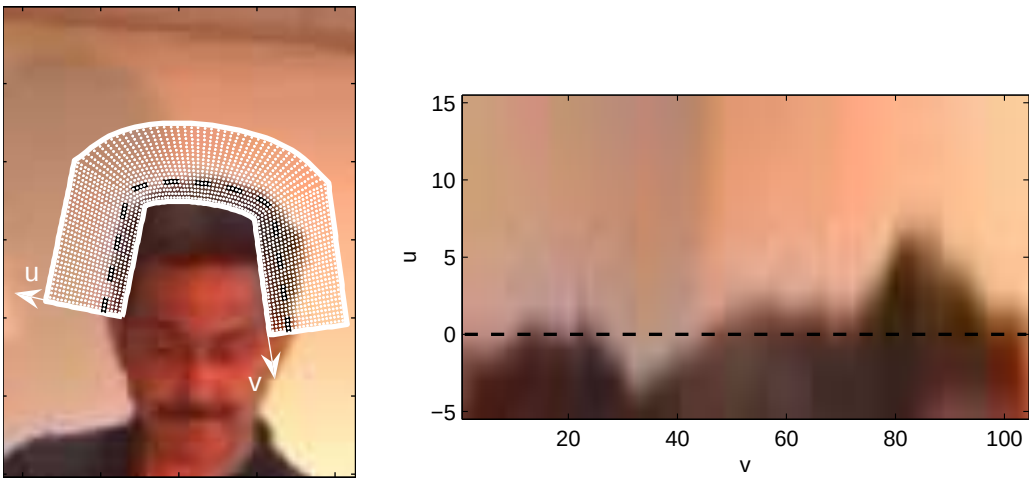
$$nV(\theta_j) = \frac{\mathbf{n}(\theta_j)}{\|\mathbf{n}(\theta_j)\|} \tag{4.7}$$

Assuming row $i = 0$ is the edge row for simplicity, the image location of the jth vertex on the ith trellis row is:

$$V(i, j) = \mathbf{r}(\theta_j) + i \times nV(\theta_j) \tag{4.8}$$

In high curvature regions consecutive trellis columns may intersect, in which case the trellis column closest to the trellis boundary is discarded. This is not an ideal treatment however, and limits the number of trellis rows interior to the edge row that can be used.

In cases where the link has no surface limits, as discussed in Section 3.1.2, the link's end is defined between $0 \leq \theta \leq 2\pi$, and the vertex locations of the first and last columns of this trellis match the first and last columns of the trellises formed around the link's projected sides. This follows Equation 3.6, where the ellipse tangents at θ_1 and θ_v match the edge directions (ignoring discarded columns). In cases where the link has surface limits, the link's end ellipse is only defined between $\theta_{l1} \leq \theta \leq \theta_{l2}$, and so up to three separate trellises may be required to measure the end of the link. The first a curved trellis from θ^{E1} to θ_{l2}, the second a rectangular trellis formed around the necessarily straight surface limit edge, and the third a curved trellis from θ_{l1} to θ^{E2}.



(a) Curved Trellis (b) Head sub-image in trellis space

FIGURE 4.3: Example curved trellis

Figure 4.3(a) shows the curved trellis around a link's end joined to the rectangular trellises formed around the link's projected sides, and Figure 4.3(b) shows the image in this trellis's space. This

transform between image and trellis space is similar to a polar unwrapping [4].

4.4 Occluding Contour Trellis Formulation

While the previously described graph based approach enforces edge consistency along a trellis, it is also desirable that the edge is consistent between the trellises formed around matching sides of consecutive links in the kinematic chain. In this *joined cost path* (JCP) approach, trellises are joined together to form a single trellis for each occluding contour of the object. In the approach presented here, an articulated object model may have multiple occluding contours due to branches in the kinematic chain.

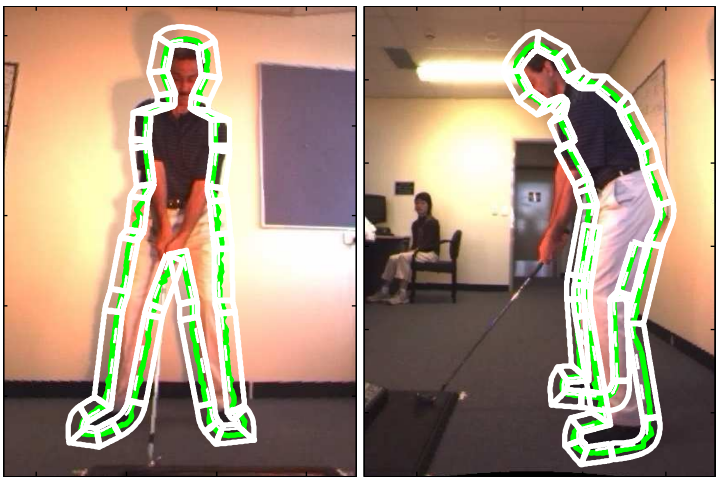


FIGURE 4.4: Chain of Joined Trellises

Figure 4.4 shows the boundaries of an occluding contour's trellis, formed by joining the edge trellises from the same links in both views, and the shortest path across this joined trellis. The divisions between the individual trellises used to form this single trellis are also shown. A method to build an occluding contour's trellis is discussed in the following sections.

4.4.1 Trellis Chains

To form each occluding contour trellis, a chain of trellises is created using a predefined order to determine which links' trellises are joined together. This order is based upon the structure of the kinematic chain. In the formulation used in later experiments, three trellis chains are used to determine the three occluding contours for the object in each view, where the three chains are formed around the following links:

1. Both edges of the hip, torso, neck, head, legs and feet,

2. The left shoulder and arm,

3. The right shoulder and arm.

The first chain does not include the arms because the upper back's link has three branches: the two shoulders and the neck. In the author's opinion there is no static order to follow around these three branches for an arbitrary human pose. While during a golf swing the golfer's hands are effectively joined together, it may be preferential to form separate chains for the interior and exterior of both arms (so both chains have a 'V' shape from a front on view). This is not done however to keep the measurement function more general, and because the state space is not restricted to poses where the hands are joined together, and attempting to join hand trellis's for arbitrary hand positions is undesirable.

4.4.2 Rectangular Trellis Joining Function

To join corresponding rows of two rectangular trellises, parametric cubic polynomials are used to describe the joins between the last column of the 'left' trellis, and the first column of the 'right' trellis. Each of these polynomials has the general form:

$$x_i(s) = a_i^x s^3 + b_i^x s^2 + c_i^x s + d_i^x \tag{4.9}$$

$$y_i(s) = a_i^y s^3 + b_i^y s^2 + c_i^y s + d_i^y \tag{4.10}$$

$$0 \leq s \leq 1$$

for each trellis row i.

The first constraint on the polynomial coefficients is that the polynomial should intersect the last vertex of the 'left' trellis, and the first vertex of the 'right' trellis. Using $V^L(i,v) = \left\{V_x^L(i,v), V_y^L(i,v)\right\}^T$ and $V^R(i,1) = \left\{V_x^R(i,1), V_y^R(i,1)\right\}^T$ to denote the image coordinates of the last vertex on row i of the 'left' trellis, and the first vertex on row i of the 'right' trellis, these constraints can be formulated as:

$$\{x_i(s=0), y_i(s=0)\}^T = V^L(i,v) \tag{4.11}$$

$$\{x_i(s=1), y_i(s=1)\}^T = V^R(i,1) \tag{4.12}$$

To ensure a smooth joining process, such that perturbing either trellis's location or orientation has limited effect on the joined portion (which will in turn improve the smoothness of the measurement function), the derivatives of each polynomial at $s=0$ and $s=1$ are chosen to match the left and right trellis's row directions, which were calculated using Equation 4.1. Using $dV^L = \left\{dV_x^L, dV_y^L\right\}^T$ and

$dV^R = \left\{dV_x^R, dV_y^R\right\}^T$ to denote the normalized row directions of the left and right trellises, these constraints on the polynomial coefficients can be formulated as:

$$\left\{\frac{dx_i(s=0)}{ds}, \frac{dy_i(s=0)}{ds}\right\}^T = \lambda dV^L \tag{4.13}$$

$$\left\{\frac{dx_i(s=1)}{ds}, \frac{dy_i(s=1)}{ds}\right\}^T = \lambda dV^R \tag{4.14}$$

with

$$\lambda = \left\|V^R(i,1) - V^L(i,v)\right\| \tag{4.15}$$

the image distance between the two vertices. Equations 4.11–4.14 form a linear system which uniquely determines the polynomial coefficients.

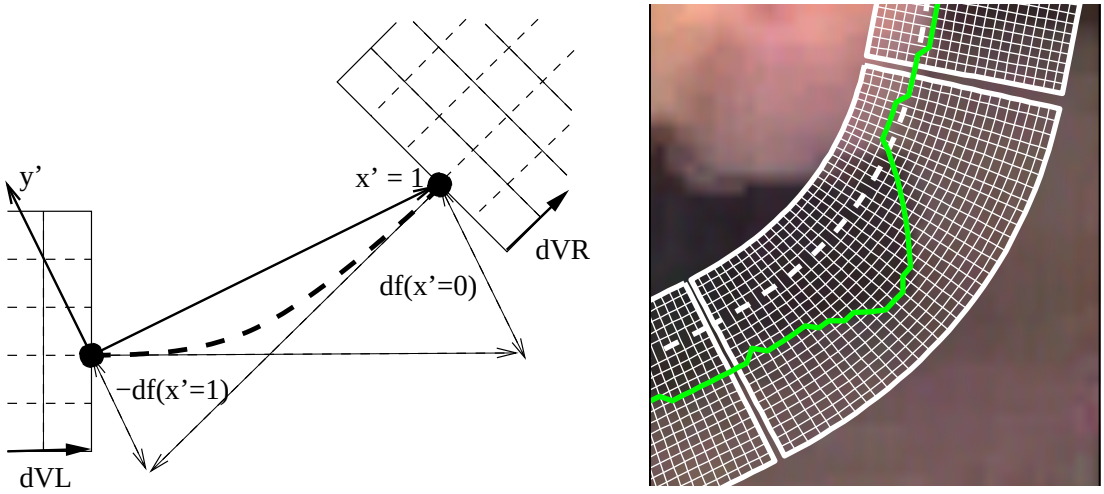


FIGURE 4.5: Trellis Joining Illustration

Figure 4.5 shows an example of joined trellises with the shortest path overlayed.

4.4.3 Derivative Limiting

In Section 4.4.2, separate polynomials were used for each row as opposed to using trellis columns directed along a single polynomials normals, because these normals often intersect at small distances along them. Problematic cases can still occur using separate polynomials however. Considering transforming the polynomials $x(s)$ and $y(s)$ to a function $y' = f(x')$ in a space defined by the basis $\{x', y'\}$ shown in Figure 4.5[3]. Using $df(x') = \frac{df(x')}{dx'}$, problematic cases occur when $|df(x'=0) + df(x'=1)|$ is large. In this situation the trellis rows can lose their desired one pixel apart spacing. To compensate for this, a function is used to limit the values of $df(x'=0)$ and $df(x'=1)$. In keeping with the

[3] $f(x')$ can not always be strictly expressed as a function. For some combinations of polynomials $x(s)$ and $y(s)$, there are two solutions for $f(x')$ for some x', hence the use of the parametric functions $x(s)$ and $y(s)$. The basis $\{x', y'\}$ is not used computationally, but is described here as it is conceptually simpler than the parametric space.

theme of ensuring continuity, a smooth scaling function is used rather than imposing hard limits. This scaling function s_c is given by:

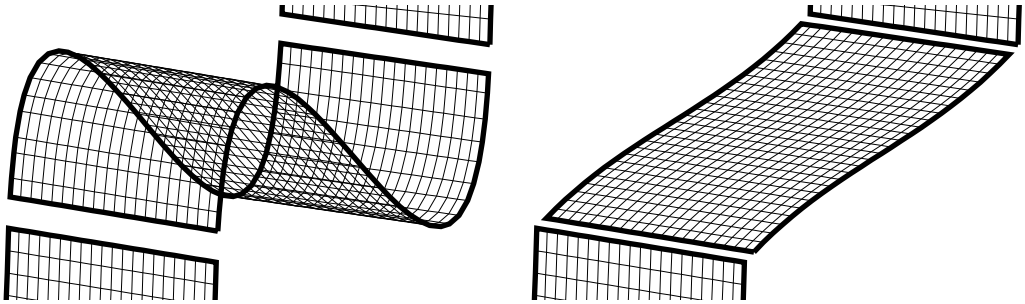
$$s_c = \min\left(\left|\frac{dx_c + dx_m\theta}{df(x'=0) + df(x'=1)}\right|, 1\right) \tag{4.16}$$

where $\theta = \arccos(dV^L \cdot dV^R)$ is the angle between the row directions dV^L and dV^R, and dx_c, dx_m, are tunable smoothing parameters[4]. Equations 4.13 and 4.14 are then replaced by:

$$\left\{\frac{dx_i(s=0)}{ds}, \frac{dy_i(s=0)}{ds}\right\} = s_c \lambda dV^L \tag{4.17}$$

$$\left\{\frac{dx_i(s=1)}{ds}, \frac{dy_i(s=1)}{ds}\right\} = s_c \lambda dV^R \tag{4.18}$$

Figure 4.6 shows an example where both $df(x'=0)$ and $df(x'=1)$ are large and have the same sign. The derivative limiting dramatically reduces the size of the local maximum and local minimum of the polynomials, ensuring trellis rows do not intersect. In this example and in the experiments performed throughout this thesis, $dx_c = 0.25$, and $dx_m = 0.7$ were chosen empirically to always maintain a reasonable spacing between the rows in the join trellis.



(a) No Derivative Limits (b) Derivative Limits

FIGURE 4.6: Trellis Joining Derivative Limiting

4.4.4 Joining Intersecting Rectangular Trellises

In some cases the two trellises to be joined may intersect each other. In this case, the rows of each trellis are shortened such that no intersection occurs. The intercept distances $d^L(i)$ and $d^R(i)$ for the ith row of the left and right trellises are calculated, ensuring a one pixel gap is left between the end points (following the row directions). Again using $V^L(i,v)$ and $V^R(i,1)$ to denote the image coordinates of the last and first first vertices on the ith row of the left and right trellises respectively, and dV^L and dV^R to denote the normalized row direction for these vertices, the row intercept distances

[4]When $f(x')$ cannot be expressed as a function, either $-df(x'=0)$ or $-df(x'=1)$ is used, to match the quadrants that dV^L and dV^R lie in.

$d^L_{int}(i)$ and $d^R_{int}(i)$ and for the ith row of each trellis are calculated:

$$d^L_{int}(i) = \frac{\langle \Delta V \times dV^R \rangle_z}{\langle dV^L \times dV^R \rangle_z} - d_{gap} \tag{4.19}$$

$$d^R_{int}(i) = \frac{\langle \Delta V \times dV^L \rangle_z}{\langle dV^L \times dV^R \rangle_z} + d_{gap} \tag{4.20}$$

with:

$$\Delta V = V^R(i,1) - V^L(i,v)$$
$$d_{gap} = \frac{1}{2}\sin^{-1}\left(\frac{\arccos\left(dV^L \cdot dV^R\right)}{2}\right)$$

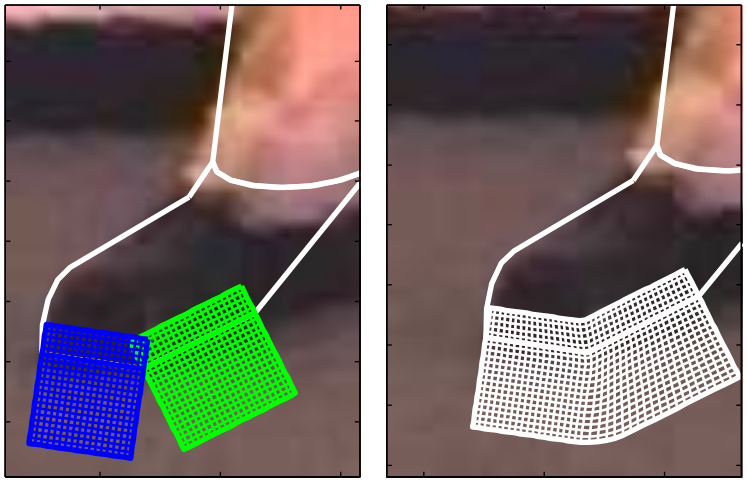
where $\cdot$ denotes the dot product and $\langle \times \rangle_z$ denotes the z component of the cross product, *i.e.* $\langle \Delta V \times dV^R \rangle_z = \Delta V_x dV^R_y - \Delta V_y dV^R_x$. Due to the trellis formulation given in Section 4.3.1, $d^L_{int}(i)$ and $d^R_{int}(i)$ are linear functions of i.

It is generally not desirable to extend the trellis rows, $d^L_{int}(i) > 0$ or $d^R_{int}(i) < 0$, and so in cases where only one trellis's row must be shortened, its row is shortened to the point where the other trellis's vertex projects onto the first trellis's row (again leaving a 1 pixel gap):

$$d^L_{int}(i) = \Delta V \cdot dV^L - 1, \quad \textbf{if } d^R_{int}(i) < 0 \tag{4.21}$$

$$d^R_{int}(i) = -\Delta V \cdot dV^R + 1, \textbf{ if } d^L_{int}(i) > 0 \tag{4.22}$$

and the other trellis's join vertex is left in its original location. Again, $d^L_{int}(i)$ and $d^R_{int}(i)$ are linear functions of i. In some cases these projection distances can extend the trellis row, which means that the trellis row does not need to be shortened.



(a) Original Trellises (b) Joined Trellis

FIGURE 4.7: Rectangular Trellis Joining

Figure 4.7 shows an example of joining intersecting rectangular trellises, where the green trellis was formed around the edge of the foot, and the blue trellis was formed around the surface limit edge

of the end of the foot. The rows interior to the edge row were shortened as described above, and the rows exterior to the edge were joined using the polynomial joining method discussed in the previous section. There are no intersecting trellis rows or columns in the joined trellises.

Using the above equations the positions of the first and last vertices for each row of each trellis in the current occluding contour can be calculated. In some model states, however, there exists consecutive trellises which can't or shouldn't be joined. These cases occur when the intercept distances are larger than the length of the projected edge the trellis is formed around, $-d^L_{int}(i) > \|\Delta\mathbf{r}^L\|$, or, $d^R_{int}(i) > \|\Delta\mathbf{r}^R\|$, with $\|\Delta\mathbf{r}\|$ the length of the link's edge given in Equation 4.2. For the model formulation used in this thesis, this occurs most often when joining a trellised formed around a kinematic chain's end link's edge trellis and its surface limit trellis. Because of the generally convex nature of each occluding contour, the number of trellises which should not be joined increases as the number of trellis rows interior to the edge row increases. This is the reason a different number of interior and exterior trellis rows are used in later experiments, where 5 rows are used on the interior of the link's projected edge, and 15 rows are used on the exterior of this edge.

4.4.5 Joining a Rectangular Trellis to a Curved Trellis

Curved trellises are formed around the end ellipses of links which at the end of a branch of the kinematic chain. As previously discussed in Section 4.3.2, the formulation for the link's projected sides means the vertex locations for the first and last curved trellis columns, $V^C(:,1)$ and $V^C(:,v_C)$, generally match the last and first trellis columns of the neighbouring trellis formed around this link's sides, $V^L(:,v_L)$ and $V^R(:,1)$, *i.e.* $V^C(:,1) = V^L(:,v_L)$ and $V^C(:,v_C) = V^R(:,1)$, with v_L the number of trellis columns of V^L. In this case joining is trivial, and the column $V^L(:,v_L)$ or $V^R(:,1)$ is simply removed from V^L or V^R.

When the link has surface limits, discussed in Section 3.1.2, or trellis columns have been discarded as per Section 4.3.2, a more rigorous joining method must be employed. When joining a rectangular trellis V^L to a curved trellis V^R, the intersection distances, $d^L_{int}(i)$ and $d^R_{int}(i)$, for the ith row of V^L and V^R are calculated using Equations 4.19 and 4.20, with dV^R the curve tangent direction at $V^R(i,1)$. When $d^R_{int}(i) > 0$, the ith row of V^L intersects the ith row of V^R in the region defined by the curved trellis, and so the number of columns of V^R must be reduced.

As discussed in Section 4.3.2, the jth column $V^R(:,j)$ has a corresponding θ_j, such that $V^R(i+1,j) - V^R(i,j)$ is orthogonal to the projected ellipse's tangent direction at θ_j for all rows i. Each row of V^R is then defined in the range $\theta_1 \leq \theta \leq \theta_{v_R}$ (for simplicity it is assumed that $\theta_1 < \theta_{v_R}$), where v_R is the number of columns of V^R. To reduce the number of columns of V^R, $\theta_{int}(i)$ corresponding

to the intersection of rows $V^L(i,:)$ and $V^R(i,:)$ is calculated. Examining the vertex placement equation for curved trellises given in Equation 4.8, there is no simple expression for the ith row's vertex locations as a function of θ. Instead an approximate solution to $\theta_{int}(i)$ is found by finding using the 'spans' of $V^R(i,:)$ to determine the column j, such that $V^L(i,:)$ intersects the span (line) between $V^R(i,j-1)$ and $V^R(i,j)$. The accuracy of this estimate is improved by using a localized quadratic model of θ as a function of the distance, s, between vertices: ost

$$\begin{aligned} \mathbf{s} &= \left\{-\|V^R(i,j)-V^R(i,j-1)\|, 0, \|V^R(i,j+1)-V^R(i,j)\|\right\}^T \\ f(s=\mathbf{s}) &= \{\theta_{j-1},\theta_j,\theta_j\}^T \end{aligned} \tag{4.23}$$

and generating n uniformly spaced points in the range $\mathbf{s}_1 \leq s \leq 0$, giving the trial points $\{\theta_1^{trial},\ldots,\theta_n^{trial}\}$. The image coordinates for these trial points, $\mathbf{r}^{trial}$, are then calculated using Equation 4.8. Again the appropriate "span" (and hence trial point k) is selected such that $V^L(i,:)$ intersects the span between $\mathbf{r}(\theta_{k-1}^{trial})$ and $\mathbf{r}(\theta_k^{trial})$. The error of the approximation of $\theta_{int}(i)$ is approximately $\frac{\|V^R(i,j)-V^R(i,j-1)\|}{n}$ pixels, and so n can be dynamically chosen to achieve a desired accuracy. Because of the generally convex shape of the occluding contour, and the necessarily convex curved trellis, these intersection generally occur on interior trellis rows, where $\|V^R(i,j)-V^R(i,j-1)\|<1$. In the experiments performed later, this error was set to ≈ 0.1 pixels, where a small error is used so that perturbing the model state slightly only slightly changes the vertex locations, which helps ensure a smooth measurement function.

The intercept values for the trellises can then be set:

$$\theta_{int}(i) = \theta_k^{trial} \tag{4.24}$$

$$d_{int}^R(i) = 0 \tag{4.25}$$

$$d_{int}^L(i) = p_{dist} - \sqrt{1-p_{err}^2} \tag{4.26}$$

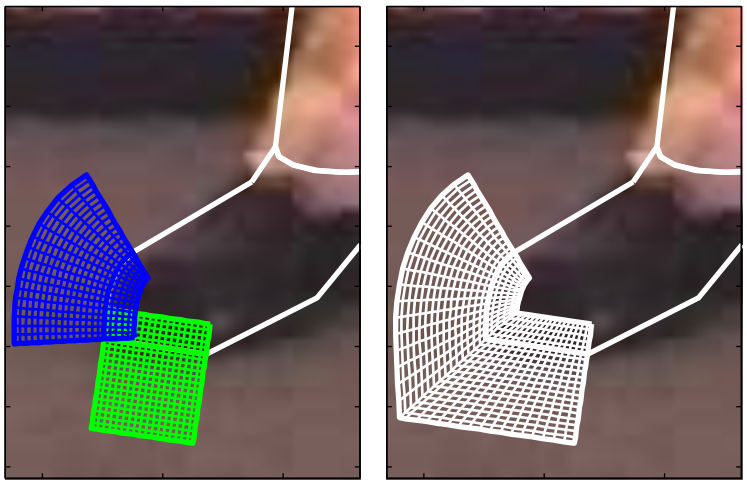
with

$$p_{dist} = \left(\mathbf{r}_k^{trial} - V^L(i,v_L)\right)\cdot dV^L \tag{4.27}$$

$$p_{err} = \|\mathbf{r}_k^{trial} - V^L(i,v_L) - p_{dist}dV^L\| \tag{4.28}$$

where p_{dist} is the projection distance of $\mathbf{r}_k^{trial}$ along $V^L(i,:)$, and p_{err} is the smallest distance between $\mathbf{r}_k^{trial}$ and $V^L(i,:)$. Again $d_{int}^L(i)$ was formulated to give a one pixel spacing between consecutive vertices in the joined configuration.

Figure 4.8 shows an example of joining a rectangular trellis (green) to a curved trellis (blue). The joined trellis has no row or column intersections. Unlike when joining two rectangular trellises, in this case join distances which extend the trellis rows, $d_{int}^L(i)>0$, are considered acceptable. If $\theta_{int}(i)>\theta_{v_R}$, the curved trellis can not be joined to this rectangular trellis.



(a) Original Trellises (b) Joined Trellis

FIGURE 4.8: Curved Trellis Joining

4.4.6 Vertex Placement for Rectangular Trellis Joins

Once the join distances and join polynomials for either side of a trellis V have been calculated, the vertex locations in the occluding contour's trellis corresponding to trellis V can be calculated. In the approach presented here, the join portion between trellises is determined to be part of the 'left' trellis in the joining process for the calculation of vertex placements.

Given the join distances $d_i = \{d_i^L, d_i^R\}$, for each row i, $1 \leq i \leq u$, of a trellis, where d_i is the distance along the row from the original vertex $V(i,1)$ (assume $d_i^R > d_i^L, \forall i$, *i.e.* a successful join), the link's edge length $\|\Delta\mathbf{r}\|$ given in Equation 4.2, and the join polynomials for each row, $x_i(s)$ and $y_i(s)$, the total image distance of each row can be calculated:

$$D_i = d_i^R - d_i^L + \mathcal{A}_i \tag{4.29}$$

where $\mathcal{A}_i$ is the arc length of the polynomial join section between $0 \leq s < 1$. For computational efficiency, the arc length is approximated by first finding the middle row, *mid*, which requires a join section (Figure 4.7 showed an example where not all rows require a join section).

$$low = \min(i|\, d_i^R = \|\Delta\mathbf{r}\|) \tag{4.30}$$

$$upp = \max(i|\, d_i^R = \|\Delta\mathbf{r}\|) \tag{4.31}$$

$$mid = \frac{upp - low}{2} + low \tag{4.32}$$

where *low* and *upp* are the first and last rows which require a join section, and *mid* is rounded towards the edge row. The arc length for this middle row is calculated numerically (again precession is not essential), and the other rows' arc lengths are estimated from this:

$$\mathcal{A}_{mid} = \int_0^{1-\lambda_{mid}^{-1}} \sqrt{\frac{dx_{mid}(s)}{ds}^2 + \frac{dy_{mid}(s)}{ds}^2}\, ds \tag{4.33}$$

$$\mathcal{A}_i = \begin{cases} = \frac{\lambda_i}{\lambda_{mid}} \mathcal{A}_{mid}, & \text{if } low \leq i \leq upp \\ = 0, & \text{otherwise} \end{cases} \tag{4.34}$$

where λ_i is the distance between vertices to be joined as defined in Equation 4.15, and the integral is taken between $0 \leq s \leq 1 - {\lambda_{mid}}^{-1}$ to preserve the 1 pixel spacing between the last vertex of the join section and the first vertex of the trellis row it is joined to. This approximation is exact when the shape of the polynomials does not change between rows, *i.e.* the join polynomials are scaled version of each other.

The number of vertices used is chosen so the vertices are spaced approximately one pixel apart on the edge row, $v = \text{round}(D_{edge}) + 1$. The vertex spacing along each row is then $v_i^{gap} = \frac{D_i}{v-1}$. Vertex locations in the region between d_i^L and d_i^R are calculated by modifying Equation 4.5:

$$V(i,j) = \mathbf{r}(\theta_s^E) + i \times nV + ((j-1) \times v_i^{gap} + d_i^L) \times dV \tag{4.35}$$

Placing vertices along the join section is more difficult however. In the implementation used here, when $\mathcal{A}_{mid}$ is approximated a 'lookup' table of the arc length as a function of s is created. When calculating the position of vertex j on row i, the desired distance along the polynomial is found:

$$d_{poly}^j = (j-1) \times v_i^{gap} - (d_i^R - d_i^L) \tag{4.36}$$

This is then transformed into an equivalent distance along the middle row's polynomial:

$$d_{poly}^{mid} = \frac{\lambda_{mid}}{\lambda_i} d_{poly}^j \tag{4.37}$$

and a linear interpolation of the lookup table is used to determine the appropriate parametric coordinate, s_j^i, for the desired vertex location. Once s_j^i has been calculated, the vertex location is simply:

$$V(i,j) = \{x_i(s_j^i), y_i(s_j^i)\}^T \tag{4.38}$$

As mentioned above, the accuracy of the arc length approximation and vertex placements is not particularly important. It is important however to ensure that there is only a small change in the vertex placements when the model state is perturbed slightly, *i.e.* the vertex locations are stable. This will in turn ensure that the measurement function is smooth, and so does not contain points with undesirably large derivatives.

4.4.7 Vertex Placement for Curved Trellis Joins

Calculating the vertex placement for a joined curved trellis V is similar to the problem of calculating the vertex placement for rectangular trellis joins. In the joined rectangular trellis case, each row

consisted of a straight section followed by a curved join section. The join curved trellis consists of a straight join section, followed by a curved section, followed by another straight join section.

Given the join distances $d_i = \{d_i^L, d_i^R\}$, for each row i, $1 \leq i \leq u$, of the curved trellis, where d_i^L and d_i^R are the distances along the curve tangents from the original vertices $V(i,1)$ and $V(i,v)$ respectively, and the (intersection) theta values $\theta_i = \{\theta_i^L, \theta_i^R\}$ for each side of the curve trellis. In this section it is assumed for simplicity that $\theta_i^L < \theta_i^R$, and so $d_i^L \leq 0$ and $d_i^R \geq 0$. The total image distance of each row can be calculated:

$$D_i = -d_i^L + \mathcal{A}_i(\theta_i^L, \theta_i^R) + d_i^R \tag{4.39}$$

where $\mathcal{A}_i(\theta_i^L, \theta_i^R)$ is the arc length of the curved section. The arc length for the edge row was previously calculated to determine the original vertex placements. For the other rows, the arc length is approximated by simply using the distances between the vertices of each row. If a row intersection occurred, the intersection distances (calculated when finding θ_{int} as per Section 4.4.5) are simply subtracted from the arc length. Row intersections along the edge row are rare, as the occluding contour doesn't intersect itself in a local region, however may occur as a consequence of joining failures (Section 4.4.8).

Again the number of vertices used is chosen so the vertices are spaced approximately one pixel apart on the edge row, $v = \text{round}(D_{edge}) + 1$. The vertex spacing along each row is then $v_i^{gap} = \frac{D_i}{v-1}$. Placing vertices in the left curve tangent extension is done by modifying Equation 4.5:

$$V(i,j) = \mathbf{r}(\theta_i^L) + i \times nV(\theta_i^L) + \left(d_i^L + (j-1) \times v_i^{gap}\right) \times dV^L \tag{4.40}$$

and similarly for the right curve tangent extension:

$$V(i,j) = \mathbf{r}(\theta_i^R) + i \times nV(\theta_i^R) + \left(d_i^L + (j-1) \times v_i^{gap} - \mathcal{A}_i(\theta_i^L, \theta_i^R)\right) \times dV^R \tag{4.41}$$

with dV^L and dV^R the normalized curve tangent directions at θ_i^L and θ_i^R.

Selecting θ values for placing vertices along the curved section is more difficult however. Again a lookup table style approach is used, where for each row a lookup table of the arc length as a function of θ is created during the approximation of $\mathcal{A}_i(\theta_i^L, \theta_i^R)$. When calculating the position of vertex j on row i, the desired distance along the curve is found:

$$d_{curve}^j = (j-1) \times v_i^{gap} + d_i^L \tag{4.42}$$

A linear interpolation of the lookup table is used to determine the appropriate value, θ_j^i, for this vertex. The vertex's image coordinates are then found using Equation 4.8.

4.4.8 Building an Occluding Contour's Trellis

The previous sections have discussed how to join trellises together, and how to calculate the vertex placements for the occluding contour once these trellises have been joined. It has been noted however that in some cases consecutive trellises can not be effectively joined. When this occurs, the particular segment which can not be joined is identified, and is removed from the occluding contour's trellis, with the trellises before and after it then joined together. When the trellises on either side of it can't be joined together either, a *break* is recorded in the occluding contour's trellis. Edge measurement consistency is not enforced over this break, and so when a break occurs at trellis column j, vertices

1) Starting with the set of trellises $\mathcal{V} = \{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ for an occluding contour:
  1a) Initialize the join distances, d_i, and theta values, θ_i, $\forall\, \mathcal{V}_i \in \mathcal{V}$, as per Section 4.4.8.
  1b) Initialize the *left joined trellises*, $\mathcal{V}^L = \{\mathcal{V}_1\}$, and *right joined trellises*, $\mathcal{V}^R = \emptyset$.
  1c) Set the current left and right trellis indices, $c = 1$, $i = 2$.
  1f) Set the current path breaks $\mathcal{B} = \emptyset$, and last failure index *last fail* $= \emptyset$.

while $i \leq n$ **do**
  2) Calculate the join distances, d_c^R and d_i^L, for joining $\mathcal{V}_c^L$ to $\mathcal{V}_i$ (Sections 4.4.4, 4.4.5).
  3) Evaluate the join success as per step 1 of Algorithm 6
  4) Evaluate Cases:
  if *join success* **then**
    Perform step 2 of Algorithm 6
  else if *last fail* $\neq \emptyset$ **then**
    Indicate a path break is needed, set: *total failure* $=$ *true*. Go back to the last failed trellis $i =$ *last fail*.
  if *total failure* **then**
    Perform step 3 of Algorithm 6
  else if *left fail* **then**
    Perform step 4 of Algorithm 6
  else if *right fail* **then**
    Perform step 5 of Algorithm 6
  5) Increment i, $i = i + 1$.
end
6) Append the path breaks $\mathcal{B} = \{1, \mathcal{B}, n + 1\}$
7) Calculate the vertex placement for each section of the occluding contour:
for *each section* i *of* $\mathcal{B}$ **do**
  Calculate the vertex placements for each $\{\mathcal{V}^R_{\mathcal{B}_{(i-1)}}, \ldots, \mathcal{V}^R_{\mathcal{B}_i - 1}\}$, as per Sections 4.4.6 and 4.4.7, and add them to the occluding contour trellis $\mathcal{V}_i^{OC}$.
end
8) **return** $\mathcal{V}^{OC}$

Algorithm 5: Generating an Occluding Contour's Trellis

$V(:,j)$ can connect to any vertex $V(:,j+1)$. This is equivalent to breaking the occluding contour's trellis into multiple parts.

Given a set of trellises $\mathcal{V} = \{\mathcal{V}_1, \ldots, \mathcal{V}_n\}$ which are formed around consecutive regions of an occluding contour, the join distances for each row j of each rectangular trellis i are initialized $d_i[j] = \{d_i^L[j], d_i^R[j]\} = \{0, \|\Delta\mathbf{r}_i\|\}\ \forall\, j \in \{1,\ldots,u\}, i \in \{1,\ldots,n\}$, where $\|\Delta\mathbf{r}_i\|$ is the row length of trellis i in the image, and u is the number of rows of each trellis. For any curved trellises $\mathcal{V}_i \in \mathcal{V}$, the ellipse θ values for the first and last columns are also recorded, $\theta_i[j] = \{\theta_{V_i}(:,1), \theta_{V_i}(:,v_i)\}$, where v_i is the number of columns in trellis V_i, and the join distances are initialized $d_i[j] = \{0,0\}$. With this initialization, Algorithm 5 can be used to determine the trellis vertex locations for each section of an occluding contour, $\mathcal{V}^{OC}$. When the join success is evaluated for a curved trellis (step 1 of Algorithm 6), for simplicity it has been assumed that $\theta_{V_i}(:,1) < \theta_{V_i}(:,v_i)$.

1) **Evaluate Success**

1a) Set: *left fail* $= \min_j \left(d_c^R[j] - d_c^L[j]\right) < 1$ **or** *left fail* $= \min_j \left(\theta_c^R[j] - \theta_c^L[j]\right) \leq 0$.

1b) Set: *right fail* $= \min_j \left(d_i^R[j] - d_i^L[j]\right) < 1$ **or** *right fail* $= \min_j \left(\theta_i^R[j] - \theta_i^L[j]\right) \leq 0$.

1c) Set: *join success* = $\neg$ *left fail* **and** $\neg$ *right fail*.

1d) Set: *total failure* = *left fail* **and** *right fail*.

2) **Join Success**

2a) Add $\mathcal{V}_c^L$ to the set of right joined trellises, $\mathcal{V}^R = \{\mathcal{V}^R, \mathcal{V}_c^L\}$.

2b) Add $\mathcal{V}_i$ to the set of left joined trellises, $\mathcal{V}^L = \{\mathcal{V}^L, \mathcal{V}_i\}$, $c = c+1$.

2c) Reset the last failure, *last fail* $= \emptyset$.

3) **Total Failure**

3a) Add the failure to the set of path breaks, $\mathcal{B} = \{\mathcal{B}, c\}$.

3b) Re-initialize the join distances d_c^R and d_i^L.

3c) Perform step 2.

4) **Left Failure**

Attempt to join trellis i to the trellis preceding $\mathcal{V}_c^L$.

4a) Calculate the join distances, d_{c-1}^{R*}, d_i^{L*} for joining $\mathcal{V}_{c-1}^L$ to $\mathcal{V}_i$.

4b) Evaluate the success of d_{c-1}^{R*} and d_i^{L*} as per step 1.

if *join success* **then**

4c) Record these join distances, set $d_{c-1}^R = d_{c-1}^{R*}$ and $d_i^L = d_i^{L*}$.

4d) Replace $\mathcal{V}_c^L$ with $\mathcal{V}_i$.

4e) Reset the last failure, *last fail* $= \emptyset$.

else

4f) Set a path break by performing step 3.

5) **Right Failure**

5a) Record this failure, set *last fail* $= i$.

Algorithm 6: Trellis Join Cases

The effect that non-joinable trellises have on the measurement function is discussed later in Section 4.11. Also, following the vertex placement method described in Sections 4.4.6 and 4.4.7, the vertex spacing for each row of an occluding contour's trellis varies across different regions. While not investigated, it is envisaged that could lead to a situation where a configuration has an undesirably high measurement score, because vertices are densely packed around a 'strong' edge, are sparsely situated upon a 'weak' edge.

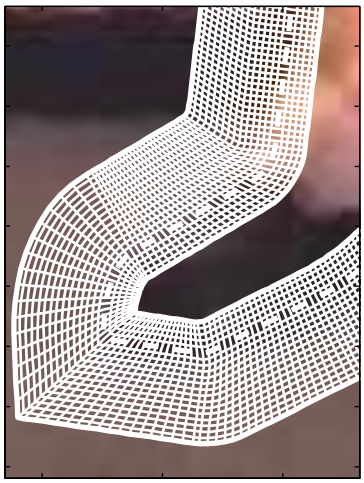


FIGURE 4.9: Occluding contour trellis around the left foot

Figure 4.9 shows the occluding contour trellis around the left foot, which was used for illustrating the curved and rectangular trellis joinings. While the ends of the occluding contour trellis could be joined together, the *circular* shortest path problem has complexity $O(u^{1.6}v)$ [4] as opposed to $O(uv)$ for the standard shortest path problem, with $u = 21$ for the case illustrated here. It was judged that the improvement in measurement consistency resulting from joining the ends of the shortest path would not justify the increase in computation expense.

4.5 Cost Metric

To determine the vertex weights ($M(i,j)$ in Algorithm 1), an image gradient based cost metric is used. Following the shortest path formulation, image regions with high gradients are given low costs. The formulation used by Appleton and Sun [4] is adopted here, where the cost metric $M_G(x,y)$ is given by:

$$M_G(x,y) = \frac{1}{(\tau + \nabla^s_\sigma I(x,y))^p} \tag{4.43}$$

with $\tau \geq 1$, and p typically equal to 2. The expression $\nabla^s_\sigma I(x,y) \geq 0, \forall(x,y)$ is used to denote an appropriate spatial gradient performed on an image $I(x,y)$ smoothed by a symmetric Gaussian filter

with standard deviation σ. In the experiments performed in this thesis the spatial gradient $\nabla^s_\sigma I(x,y)$ is taken to be the magnitude of combined colour and intensity gradient method discussed in Section 3.3.

As no edge feature discretization is required for this approach, it is simple to incorporate temporal information into the cost metric. Temporal information is used to strengthen edges that have an associated temporal difference. Equation 4.43 can then be rewritten:

$$M_G(x,y,t) = \frac{1}{(\tau + \nabla^s_\sigma I(x,y,t) \times (1 + \nabla^t_\sigma I(x,y,t)))^p} \tag{4.44}$$

Here $\nabla^t_\sigma I(x,y,t) \geq 0, \forall (x,y,t)$, is an appropriate temporal differencing function. In the experiments performed in this thesis, $\nabla^t_\sigma I(x,y,t)$ is a temporal gradient function, where the temporal gradient uses a similar combined intensity and colour formulation as the spatial gradient. If available, optical flow data could form the basis of the temporal differencing function. From the form of Equation 4.44 it follows that temporal information can not increase the cost metric's value at any point.

While the above formulation and a shortest path algorithm provides a good basis for determining the likelihood an edge runs in the trellis's v direction, for the tracking problem it is also desirable to incorporate the distance between the shortest path and a hypothesized edge location. One approach to this would be to directly measure this distance in image space, which is analogous to a line search approach where the shortest path is used as the set of edge features. This has the disadvantage that another path may exists which is significantly closer to the hypothesized edge location, but with slightly greater cost to the shortest path. Also, due to the finite number of rows in the trellis, perturbing the trellis location in the u direction may significantly change the shortest path, as a new potential path is added. This has the potential to increase the number of modes in the observational likelihood.

To penalize trellises where the shortest path would otherwise be distant from the hypothesized edge location, a penalty term is added to each vertex based on the distance of the trellis row from the hypothesized edge. Assuming the trellis is parameterized such that the row $u = 0$ is the hypothesized edge row, with n_{ext} rows external to the edge and n_{int} rows internal to the edge, the penalty function $M_P(u)$ should observe the following five properties:

1. $M_P(0) = 0$
2. $\frac{dM_P(u=0)}{du} = 0$
3. $M_P(u) < M_P(u+\eta), \forall |u| < |u+\eta|$
4. $M_P(u) < \delta M_G, \forall \ -n_{\text{int}} \leq u \leq n_{\text{ext}}$
5. $M_P(u) \geq \delta M_G, \forall\ u < -n_{\text{int}}$ or $u > n_{\text{ext}}$

where $\delta M_G = \max_{\{x,y,t\}} (M_G(x,y,t)) - \min_{\{x,y,t\}} (M_G(x,y,t))$. Properties 1 – 3 describe a strictly convex function with a unique minimum at $u = 0$. Property 4 ensures that there are no wasted rows, *i.e.*

rows that the shortest path can never use. Property 5 ensures that increasing the number of rows in the trellis will not change the shortest path, which in turn means that perturbing the edge location one pixel in the u direction (analogous to making a new path available) will not alter the shortest path by more than one vertex (pixel) at any point. This minimizes the number of modes in the observational likelihood. Using the cost metric formulation given in Equation 4.44, $\delta M_G \leq \tau^{-2}$. Algorithm 1 is optimal in that it is guaranteed to find the smallest cost path across a trellis, however there is no guarantee that there is a unique shortest path. Where necessary trivial modifications can be made to Algorithm 1 to ensure the shortest path closest to the edge row is found.

In the experiments performed in this thesis, a non symmetric penalty function of the form:

$$M_P(u) = \begin{cases} 1 - e^{\frac{-u^2}{\sigma_{\text{int}}^2}}, & \text{if } u \leq 0 \\ 1 - e^{\frac{-u^2}{\sigma_{\text{ext}}^2}}, & \text{if } u > 0 \end{cases} \tag{4.45}$$

is used, where σ_{int} and σ_{ext} are chosen such that $M_P(n_{\text{ext}}+1) = M_P(-n_{\text{int}}-1) = \tau^{-p}$, where τ was defined in Equation 4.44. Anecdotal evidence suggests that increasing σ_{ext} and σ_{int} can increase the zone of attraction in some regions, but introduces new local modes in other regions, which reduces the zone of attraction in these regions. Since $\frac{dM_P(u=0)}{du} = 0$, the discontinuity in the second derivative $\frac{d^2M_P(u=0)}{du^2}$ will not adversely effect any damped Newton like optimization.

Using $\mathbf{T}(i,j) : \mathbb{Z}^2 \rightarrow \mathbb{R}^2$ to denote the transformation from trellis space to image space, *i.e.* the image location of a vertex, the total cost $M(i,j)$ of vertex $V(i,j)$ is given by:

$$M(i,j) = M_G([\mathbf{T}(i,j), t]) + M_P(i) \tag{4.46}$$

4.6 Measurement Score

The measurement score used is a function of the cost of the shortest path and the number of columns in the trellis. Using $P_i[j], j \in \{1, ..., v_i\}$ to denote the shortest path across trellis i with v_i columns, the shortest path cost C_i and is associated length D_i are then:

$$C_i = \sum_{j=1}^{v_i} \begin{cases} 0, & \text{if } V(P_i[j], j) \text{ is occluded} \\ M\left(P_i[j], j\right), & \text{otherwise} \end{cases} \tag{4.47}$$

$$D_i = \sum_{j=1}^{v_i} \begin{cases} 0, & \text{if } V(P_i[j], j) \text{ is occluded} \\ 1, & \text{otherwise} \end{cases} \tag{4.48}$$

where $M(P_i[j], j)$ are the vertex costs described in Section 4.5. The probability of the edge measurements Z for a model state X is then given by:

$$\log\left(p(Z|X)\right) \propto -\frac{\sum_{i=1}^{n} C_i}{\sum_{i=1}^{n} D_i} \tag{4.49}$$

where n is the number of constructed trellises in all views.

4.7 Occlusions and Shortest Paths

It is not immediately obvious how to treat self occlusions a graph based approach is used. The graph based approach is adapted from segmentation problems which typically do not use object models, and hence do not model self occlusions. As such five potential treatments are examined:

1. Ignoring occlusions (NO)

In this case self occlusions are ignored. Letting $\mathcal{P}$ denote the set of all possible paths across a trellis, the shortest path $P^* \in \mathcal{P}$ is given by:

$$P^* = \underset{P_i \in \mathcal{P}}{\arg\min} \sum_{j=1}^{v} M\left(P_i[j], j\right) \tag{4.50}$$

2. Constant occlusion cost (CO)

The weight for each trellis vertex designated as occluded is given a constant value, M_{occ}. The shortest path P^* is then the path minimizing:

$$P^* = \underset{P_i \in \mathcal{P}}{\arg\min} \sum_{j=1}^{v} \begin{cases} M_{occ} + M_P(P_i[j]), & \text{if } V(P_i[j], j) \text{ is occluded} \\ M\left(P_i[j], j\right), & \text{otherwise} \end{cases} \tag{4.51}$$

where M_{occ} is chosen to be slightly more than the expected un-penalized vertex weight, M_G, for a vertex on the shortest path in a trellis where an edge is present. This penalizes an occlusion more than if an edge is present, but less than if no edge were present. This is similar to the occlusion penalty measurement used by Sminchisescu [86] for line searches. In this case the constant occlusion cost is used when evaluating the edge likelihood cost, and so $D_i = v_i$ from Equation 4.48. In later experiments M_{occ} is set to be 20% greater than the minimum value of M_G.

3. Zero occlusion cost and distance (ZO)

In this formulation an occluded vertex does not contribute to the shortest path's cost or length. As such, different paths across a trellis can have different lengths, and so the shortest path across

the trellis is redefined to be the path which minimizes the ratio of cost to distance. The shortest path $P^* \in \mathcal{P}$ is the given by:

$$P^* = \arg\min_{P_i \in \mathcal{P}} \frac{C_i}{D_i} \tag{4.52}$$

where:

$$\begin{aligned} C_i &= \sum_{j=1}^{v} \begin{cases} M_{occ} = 0, & \text{if } V(P_i[j], j) \text{ is occluded} \\ M\left(P_i[j], j\right), & \text{otherwise} \end{cases} \\ D_i &= \sum_{j=1}^{v} \begin{cases} D_{occ} = 0, & \text{if } V(P_i[j], j) \text{ is occluded} \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{4.53}$$

This is the same metric as is used to calculate the edge likelihood once the shortest path has been calculated, given in Section 4.6.

4. Zero occlusion cost and distance, possible occlusions (PO)

This method is similar to the ZO approach, however each path through the trellis can use an occluded vertex as either occluded or unoccluded . There are then more possible paths using this approach as opposed to the ZO method, as two distinct paths may contain the same set of vertices, however use different hypotheses about whether each vertex is occluded. The occlusion uncertainty is introduced to compensate for inaccuracies in the self occlusion model.

5. Biased occlusions (BO)

In this approach the shortest paths are intentionally biased towards occluded vertices. This represents a concession that each link's surface may not model the object accurately, because of deformation characteristics such as loose fitting clothing. As such the self occlusion map itself will not be accurate. Biasing the shortest paths towards occluded pixels effectively limits the possible paths near occluded vertices, and is aimed at reducing the variability of the measurement function due to inaccurate self occlusion modelling.

While this bias may be introduced by giving occluded vertices a negative vertex weight, the bias is instead introduced by re-weighting the distance score for an occluded pixel. Modifying the distance score is advantageous as the distance score is not a function of the image data, and so the same value is suitable for all image sequences. The re-weighted distance score is consistent with the redefinition of the shortest path as the path which minimizes the ratio of cost to distance as used by the ZO and PO methods, given in Equation 4.52. The cost of a path P_i remains the same as in Equation 4.53, while the distance of a path used to determine the

shortest path is given by:

$$D_i = \sum_{j=1}^{v} \begin{cases} D_{occ}, & \text{if } V(P_i[j], j) \text{ is occluded} \\ 1, & \text{otherwise} \end{cases} \quad l \tag{4.54}$$

where $D_{occ} > 1$ controls the strength of the bias. In later experiments, $D_{occ} = 1.2$ has been used.

1) Insert occlusions values into the trellis:

foreach *vertex* $i = 1 : u \times v$ **do**
  if $V(i,1)$ *occluded* **then** $M(i) = M_{occ}$; $D(i) = D_{occ}$;
  else $D(i) = 1$;
end

2) Initialize the first column enforcing start locations:

foreach *row* $i = 1 : u$ **do**
  if $V(i,1) \in S$ **then** $C^*(i,1) = M(i,1)$; $D^*(i,1) = D(i,1)$;
  else $C^*(i,1) = \infty$; $D^*(i,1) \ll 1$;
end

3) Find the shortest path to each point in the trellis (i,j):

foreach *column* $j = 2 : v$ **do**
  foreach *row* $i = 1 : u$ **do**
    3a) Calculate the backpointer ratios:
    $R_k = \frac{M(i,j)+C^*(k,j-1)}{D(i,j)+D^*(k,j-1)}, k \in \{i-1, i, i+1\}$
    3b) Calculate the backpointer with the smallest ratio:
    $b(i,j) = \arg\min_i (R_{i-1}, R_i, R_{i+1})$
    3c) Update the path cost and distance:
    $C^*(i,j) = M(i,j) + C^*(b(i,j), j-1)$;
    $D^*(i,j) = D(i,j) + D^*(b(i,j), j-1)$;
  end
end

4) Locate the end point with shortest path ratio:

$(P[v], v) = \arg\min_{(i,v)\in T} \left(\frac{C^*(i,v)}{D^*(i,v)} \right)$

5) Follow the back pointers:

foreach *column* $j = v - 1 : 1$ **do**
  $P[j] = b(P[j+1], j+1)$;
end

Algorithm 7: Non-optimal Shortest Path by Dynamic Programming

The ZO, BO, and PO methods described above all redefine the shortest path across a trellis as the path which minimizes the ratio of cost to distance. As such, Algorithm 1 can be modified to return a path based on this metric. This is given in Algorithm 7, where M_{occ} and D_{occ} are determined by the occlusion treatment.

Algorithm 7 is not optimal, in that it is not guaranteed to return the path with the smallest cost to distance ratio. Considering a trellis with vertex $V(i,j), 1 < j < v$, with back connected vertices $V(short, j-1)$ and $V(long, j-1)$ with distance scores $D^*(short, j-1) < D^*(long, j-1)$, such that:

$$R_B = \frac{M(i,j) + C^*(short, j-1)}{D(i,j) + D^*(short, j-1)} = \frac{M(i,j) + C^*(long, j-1)}{D(i,j) + D^*(long, j-1)} \tag{4.55}$$

The back pointer $b(i,j)$ will be chosen arbitrarily as the two potential paths to the vertex have the same cost to distance ratio. Now considering the shortest path from the vertex $V(i,j)$ to the acceptable end locations (sinks), which has cost C_A, distance D_A, and ratio R_A. When $R_A \neq R_B$, two cases arise:

$$\frac{C_A + C^*(short, j-1)}{D_A + D^*(short, j-1)} < \frac{C_A + C^*(long, j-1)}{D_A + D^*(long, j-1)}, \quad \text{if } R_A < R_B \tag{4.56}$$

$$\frac{C_A + C^*(short, j-1)}{D_A + D^*(short, j-1)} > \frac{C_A + C^*(long, j-1)}{D_A + D^*(long, j-1)}, \quad \text{if } R_A > R_B \tag{4.57}$$

As such, this arbitrary path selection does not guarantee the shortest path across the trellis. In the experiments performed in this chapter, the biased occlusion approach is evaluated using Algorithm 7, as this non optimality problem arises vary rarely when the shortest path is biased towards the occlusions.

To find the shortest path where the shortest path is defined as the path minimizing the ratio of cost to distance, paths to each vertex using a given number of unoccluded pixels are built, so a dimension is added to the minimum costs $C^*(i,j)$ from Algorithm 1, such that $C^*(i,j,k)$ is the minimum cost of traversing from a source point to vertex $V(i,j)$ using k unoccluded vertices. A branch and bound [60, 105] style algorithm is adopted, where paths are prioritized assuming zero cost and all unoccluded vertices from $V(i,j)$ to the sink vertices, *i.e.* the lower bound estimate of the cost remaining is zero, and upper bound of the estimate of the distance remaining is $v - j$.

The branch and bound style also allows the inclusion of the properties given in Equation 4.57. A path with cost $C^*(i,j,k)$ can be discarded if two paths to $V(i,j)$ exist which both have a better cost ratios, with more and less unoccluded vertices respectively. Algorithm 8 shows the optimal shortest path procedure. This algorithm is guaranteed optimal by the branch and bound formulation while the lower bound estimate is lower than the true lower bound, and the properties given in Equation 4.57 hold. Trivial modifications to Algorithm 8 allow the shortest path to be calculated in the presence of possible occlusions.

4.8 Experimental Evaluation

To evaluate the proposed edge measurement methods, the observational likelihood was exhaustively calculated for some simple experiments using a discrete two dimensional state space. The following quantitative properties of the observational likelihood for the different measurement schemes are

1) Insert occlusions values into the trellis, as per algorithm 7

2) Add start locations to the queue Q:

foreach *row* $i = 1 : u$

if $V(i,1) \in S$ **then**

Add node $\sigma = \{\sigma_r, \sigma_c, \sigma_d\} = \{i, 1, D(i,1)\}$ with cost $C^*(\sigma) = M(i,1)$ to Q

3) Priority search:

Loop

3a) Retrieve the unsearched node with minimum lower bound:

$\sigma = \underset{\sigma' \in Q}{\arg\min} \left(\frac{C^*(\sigma')}{\sigma'_d + v - \sigma'_c} \right)$

3b) Check for convergence:

if $\sigma_c = v$ **then**

break

3c) Examine other nodes at the same vertex, $Q' \subset Q$ with:

$\sigma'_r = \sigma_r, \sigma'_c = \sigma_c, \forall\, \sigma' \in Q'$

3d) Check for shorter and longer paths with better ratios:

if $\nexists \{\sigma', \sigma^\dagger\} \subset Q'$: $\max \left(\frac{C^*(\sigma')}{\sigma'_d}, \frac{C^*(\sigma^\dagger)}{\sigma^\dagger_d} \right) \leq \frac{C^*(\sigma)}{\sigma_d}, \sigma'_d < \sigma_d < \sigma^\dagger_d$ **then**

3e) Add neighbours to the queue:

foreach *connected vertex* $V(i,j)$ *of* $V(\sigma_r, \sigma_c)$

if $\sigma' = \{i, j, \sigma_d + D(i,j)\} \ni Q$ **then**

Add node σ' with cost $C^*(\sigma') = C^*(\sigma) + M(i,j)$ to Q

Store the backpointer $b(\sigma') = \sigma$

3f) Flag σ as searched

4) Follow the back pointers from the last retrieved node σ

Algorithm 8: Optimal Shortest Path by Branch and Bound

compared:

1) Global maximum error: The distance in the state space between the location of the global maximum and the author's interpretation of the true object state. This is used to judge the accuracy of the measurement approach. When the global maxima is not the largest maxima by area, the error between the largest maxima by area and the correct state is also given, denoted by an *.

2) Number of local maxima: The total number of local maxima in the likelihood. This gives a measure of the smoothness of the likelihood.

3) Expected Maxima Value: The value of the likelihood at the state corresponding to the author's interpretation of the true object state. This is expressed as a percentage of the difference between the global maxima's value and the global minima's value. This measure is also used to assess the accuracy of each method.

4) Maxima area: The region of the state space where there is a monotonically non-decreasing path from any point to the applicable local maxima. Any point in the state space is "assigned" to only the single (the largest) local maxima where this non-decreasing path exists. As such, any point not assigned to the global maxima is guaranteed not to converge to the global maxima using a truly local maximizer, however points assigned to the global maxima are not guaranteed to converge to the global maxima. This measure is indicative of the region of convergence of each maxima.

5) Maxima value: The difference between the mode's value and the largest value that the point could take and be part of another mode. This is given as a percentage of the difference between the global maxima's value and the global minima's value. Along with the area, this is indicative of how difficult a mode is to escape.

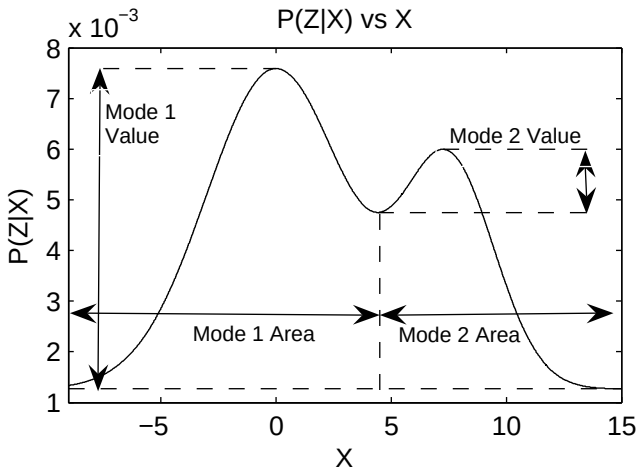


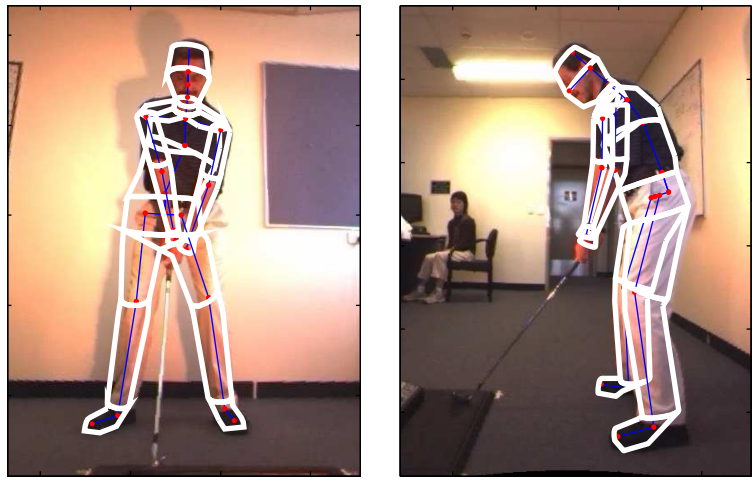
FIGURE 4.10: Mode Properties Illustration

These properties, illustrated in Figure 4.10, were calculated using a flow based approach, where the flow is initialized at the global maxima and expanded outwards such that the flow at a point is the minimum of the point's value and the maximum flow of any of its neighbours.

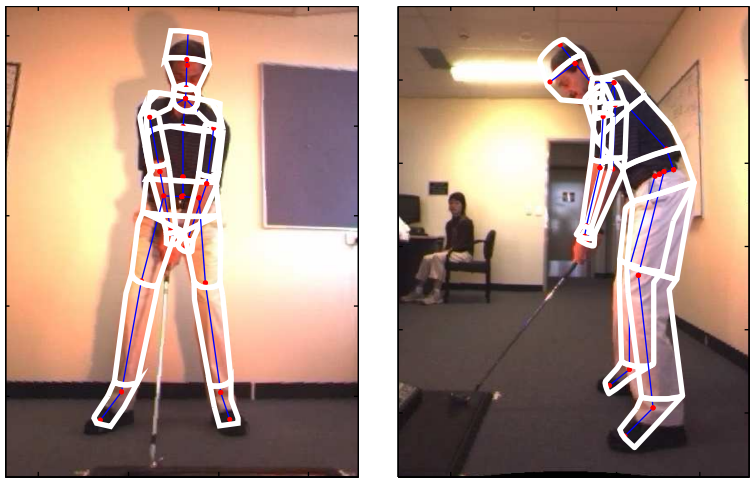
This was implemented via a priority queue approach. If a point is saturated, *i.e.* the flow at the point is equal to the point's value, and has a neighbour point assigned to the first mode, then the point is itself 'assigned' to the first mode. A new flow is then initialized at the largest unsaturated point, the second largest local maxima. The 'maxima value' measure at this point is then the difference between the point's value and the previous flow at the point. Again the new flow is expanded outwards, and any previously unsaturated point which is now saturated is 'assigned' to this mode. This process is repeated until all points are saturated. As such modes are measured to numeric precision, however

the point spacing used in the later experiments is designed to eliminate spurious modes caused by rounding errors.

4.8.1 Proposed Experiment 1



(a) X = (100,0), View 1 (b) X = (100,0), View 2



(c) X = (0,100), View 1 (d) X = (0,100), View 2

FIGURE 4.11: Adapting to Base Translation

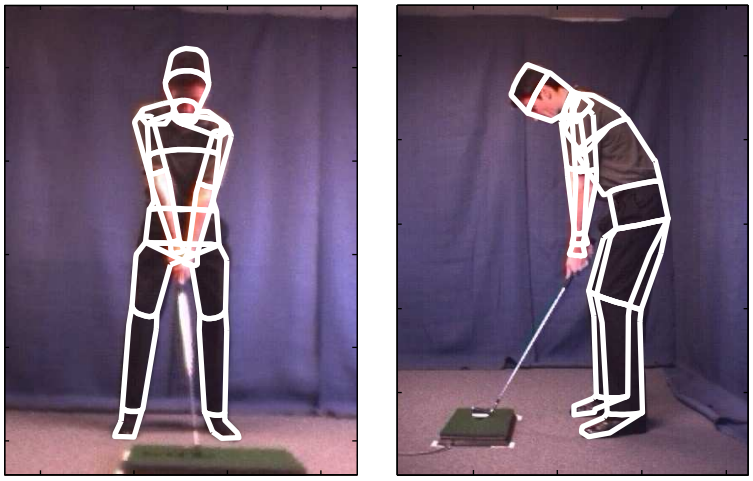
To test the various measurement approaches, a two dimensional state space was considered, where the two dimensions corresponded to the base link translations in the y and z directions of the real world coordinate system. In the articulated model formulation used here, the base link models the hips. The joint angles were automatically adjusted to leave all of the links further down the kinematic chain in the same Cartesian position (maintaining the model's geometric constraints). This adjustment process is described later in Section 6.4. In the experiments performed, the observational likelihood was sampled between ± 100mm at a 2mm resolution (10201 points) in both the y and z directions. Figure 4.11 shows how the joint angles adapted to a base link translation of 100mm in the y and z

directions, leaving the subsequent links in approximately the same position.

This state space was selected for the experiment as it seems intuitive that with noiseless data and a good measurement function, that the likelihood would be uni-modal in the test range. Furthermore, the adjustment process means the entire model was perturbed for any change in the state vector.

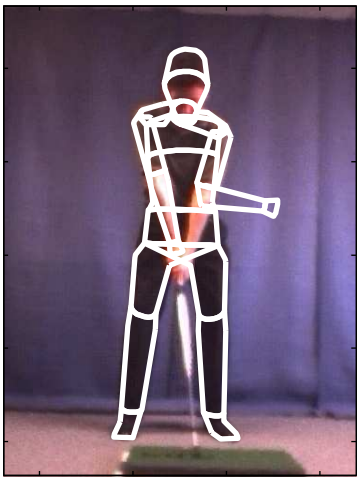
4.8.2 Proposed Experiment 2

A two dimensional state space was again considered in this experiment, where the two dimensions corresponded to an about axis z rotation of the left upper arm, and the left elbow y rotation. This z rotation was varied between $-90°$and $25°$, while the y rotation was varied between $0°$and $90°$, with both sampled at a $1°$resolution (10556 points). This range was chosen to limit body part interpenetration. The rest of the model was left in the default pose of the golfer addressing the ball. Figure 4.12 shows example states for this experiment.



(a) X = (45,11), View 1

(b) X = (45,11), View 2



(c) X = (-90,90), View 1

(d) X = (-90,90), View 2

FIGURE 4.12: PDF experiment 2 example state space points

In this experiment only a small part of the articulated model was varied over a large range. As such it was designed to test the modality of the observational likelihood when multiple local maxima are expected in the likelihood. Unlike in experiment 1, there was no reasonable expectation that the edge likelihood should be uni-modal. In particular strong local maxima were expected where the hypothesized edge location matches the 'other' true edge, *i.e.* a link's hypothesized interior edge matches the link's true exterior edge. There is also greater variation in the self occlusions in this experiment than in Experiment 1.

4.8.3 Data Sets

The above experiments were performed upon two data sets. The first, shown in Figure 4.11, has good object/background contrast for the top half of golfer, but very poor object/background contrast for the legs. The second data set, shown in Figure 4.12, has good object/background contrast for the bottom half of golfer, but poorer contrast around the head (despite the artificial background).

The same cost metric parameters from Section 4.5 were use on both images for both data sets ($\tau = 2$, $p = 2$). For methods reliant on discrete edge features however, the "edge threshold" was manually selected for each image.

4.8.4 Occlusion Treatment Experiments

The results of evaluating the different self occlusion treatments, described in Section 4.7, using the afore described experiments are given in this section. The joined cost path measurement approach, described in Section 4.4, was used because the impact of the occlusions on the shortest path will increase as the number of trellis columns (and hence occluded vertices) increases. While the method used to calculate the shortest path differs between treatments, once the shortest path has been calculated the same cost and distance functions, given in equations 4.47, are used to evaluate the edge likelihood. The results for each occlusion treatment for Experiment 1 are given in Tables 4.3–4.6. The results for each occlusion treatment for Experiment 2 are given in Tables 4.7–4.10. Summaries of these results for each experiment are given in Tables 4.1 and 4.2.

These results show that the possible occlusion (PO) treatment performed almost identically to the zero occlusion (ZO) treatment[5]. This is indicative that there were no cases where an observable edge was present on a vertex marked as occluded. It is possible that the PO occlusion treatment could

[5]The computational time of both methods is reported as being almost identical here, however the ZO treatment should be slightly more efficient. This occurred because the same implementation was used for both, however the shortest path was selected from a subset of the calculated paths for the ZO treatment.

Method	Pos. Err. mm (global max)	Pos. Err. mm (max by area)	Number of Modes	Expected Max. Val %	M_1 Height % $\times 10^{-2}$	M_1 Area %
PO	18.08 ± 7.30	18.08 ± 7.30	13.50 ± 9.15	97.07 ± 1.00	0.13 ± 0.01	98.67 ± 1.61
ZO	18.08 ± 7.30	18.08 ± 7.30	13.50 ± 9.15	97.07 ± 1.00	0.13 ± 0.01	98.67 ± 1.61
NO	13.73 ± 3.36	13.73 ± 3.36	10.25 ± 3.86	97.37 ± 1.21	0.09 ± 0.01	93.15 ± 10.1
BO	17.65 ± 3.93	17.65 ± 3.93	9.75 ± 8.06	96.75 ± 1.80	0.14 ± 0.01	99.00 ± 1.12
CO	16.36 ± 3.74	16.36 ± 3.74	8.75 ± 6.85	97.22 ± 0.78	0.10 ± 0.01	97.71 ± 3.69

Table 4.1: Experiment 1 Translation, Occlusion Treatment Summary

Method	Pos. Err. ° (global max)	Pos. Err. ° (max by area)	Number of Modes	Expected Max. Val %	M_1 Height % $\times 10^{-2}$	M_1 Area %
PO	15.18 ± 18.0	6.54 ± 2.16	207.0 ± 51.8	98.82 ± 0.47	5.55 ± 1.97	35.46 ± 22.0
ZO	15.18 ± 18.0	6.54 ± 2.16	207.0 ± 51.8	98.82 ± 0.47	5.55 ± 1.97	35.46 ± 22.0
BO	19.70 ± 18.8	6.33 ± 3.25	166.3 ± 22.9	98.79 ± 0.49	5.48 ± 1.74	39.30 ± 14.9
CO	18.39 ± 14.3	8.69 ± 3.17	138.8 ± 16.5	97.79 ± 2.17	4.03 ± 1.43	39.37 ± 23.5
NO	17.51 ± 19.1	27.89 ± 22.9	83.8 ± 24.5	98.44 ± 0.78	3.65 ± 1.54	43.11 ± 28.8

Table 4.2: Experiment 2 Rotation, Occlusion Treatment Summary

be improved by considering a larger set of vertices as potentially occluded, as this method does not necessarily discard measurements from occluded vertices.

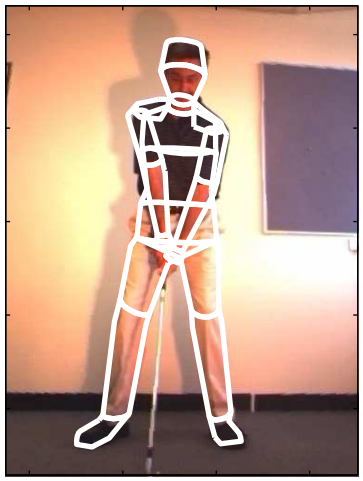


FIGURE 4.13: Global maximum state, X = (24, 12)

Upon examination of the summaries in Tables 4.1 and 4.2, it is not immediately obvious which treatment has performed the best. All treatments have large errors in the position of the global maximum, however these are the result of performing the experiment in the monocular case. Unrepresentative local maximum can perhaps be expected due to the depth ambiguity. As an example, the global maxima for the biased occlusion (BO) treatment using the first data set for the second experiment had a reported positional error of $30°$. Figure 4.13 shows the object state corresponding to this global maxima, which looks quite correct from the single view. All methods except the no occlusion (NO) treatment reported a small error in the position of the mode with the largest area in the monocular

case, which is indicative the joined cost path approach is accurate enough to resolve the depth ambiguity to some extent. This ability to resolve the ambiguous direction was a somewhat surprising result, and was part of the motivation for the local optimization method proposed in Chapter 6.

Of the treatments which minimized the ratio of energy to distance (ZO, PO, and BO), the BO treatment has the smallest number of modes while having an accuracy equivalent to the other methods. The performance of the NO method was relatively worse in experiment 2, where the self occlusions are more variable than in experiment 1. This treatment could be expected to perform most poorly where the true position was significantly occluded.

There is little difference between the performance of the constant occlusion (CO) and BO treatments, with the CO treatment performing marginally better in general. The CO treatment's free parameter is the occlusion cost, M_{occ}, while the BO treatment's free parameter is the occlusion distance, D_{occ}. In these experiments, the occlusion cost was set to 20% of the range between the image cost metric's values. The minimum of the cost metric may not occur on the target object's edge however, and as such this occlusion cost may be independent of the object appearance. Comparisons between object states with different numbers of occluded vertices may then be dependent upon unrepresentative background information using the CO treatment. Even if the minimum of the cost metric occurs on the object boundary, using the object/background contrast for a different part of the object may prove problematic.

In contrast the BO treatment's free parameter, the distance bias, is invariant to the image being analysed. As such, altering the image's background in an area away from the object will not effect the likelihood of the true object state, and so comparisons between different object states aren't effected by unrepresentative background information. The author feels this is sufficient reason to prefer the BO treatment to the CO treatment, and so the BO treatment has been used in experiments presented later in this thesis.

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
PO	21	24	95.57%	0.1154	98.4609	0.0842	0.3725	63.3
ZO	21	24	95.57%	0.1154	98.4609	0.0842	0.3725	63.4
BO	21	15	94.10%	0.1237	98.3433	3.9136	0.6862	13.2
CO	18	15	96.17%	0.0995	98.6374	0.4643	0.5294	12.6
NO	18	14	97.26%	0.0895	99.1667	0.1792	0.2843	12.7

Table 4.3: 2D translation occlusion experiment *YZ* – data set 1, 1 view

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
PO	27	18	97.58%	0.1362	96.4709	7.1717	1.4802	65.3
ZO	27	18	97.58%	0.1362	96.4709	7.1717	1.4802	66.5
BO	16	18	97.20%	0.1427	97.7551	7.0193	1.3528	15.7
CO	20	14	97.27%	0.1087	92.2557	0.7314	3.5487	14.5
NO	15	13	97.56%	0.1032	76.7278	3.2419	14.116	14.0

Table 4.4: 2D translation occlusion experiment *YZ* – data set 2, 1 view

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
PO	14	8	97.71%	0.1392	99.8824	0.0969	0.0294	194.2
ZO	14	8	97.71%	0.1392	99.8824	0.0969	0.0294	194.4
NO	10	6	98.80%	0.0835	97.3336	0.9920	2.5880	22.4
BO	20	1	97.90%	0.1474	100.0000	–	–	23.8
CO	14	1	98.06%	0.1060	100.0000	–	–	22.6

Table 4.5: 2D translation occlusion experiment *YZ* – data set 1, 2 views

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
NO	12	8	95.85%	0.0724	99.3726	0.3649	0.3235	24.5
BO	13	5	97.81%	0.1294	99.9020	0.0415	0.0490	26.9
CO	12	5	97.36%	0.0829	99.9314	0.3772	0.0294	25.4
PO	10	4	97.42%	0.1220	99.8726	0.2935	0.0588	213.5
ZO	10	4	97.42%	0.1220	99.8726	0.2935	0.0588	213.7

Table 4.6: 2D translation occlusion experiment *YZ* – data set 2, 2 views

Method	Maxima Err. (°)	Num. Modes	Expected Max. Val.	M_1 Height $\times 10^{-2}$	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
PO	5	207	98.22%	4.699	31.2334	7.6989	20.443	81.4
ZO	5	207	98.22%	4.699	31.2334	7.6989	20.443	81.6
BO	30, 6*	176	98.07%	4.785	4.1682	0.9515	34.985	14.7
CO	5	155	98.35%	3.879	30.8355	7.3335	21.845	14.5
NO	10, 49*	98	98.43%	3.594	8.6112	1.4412	23.986	13.9

Table 4.7: 2D rotation occlusion experiment *YZ* – data set 1, 1 view

Method	Maxima Err. (°)	Num. Modes	Expected Max. Val.	M_1 Height $\times 10^{-2}$	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
BO	41, 10*	194	98.88%	7.718	5.0303	0.0376	34.123	11.9
PO	42, 4*	156	98.77%	7.971	11.9553	0.2431	32.408	62.2
ZO	42, 4*	156	98.77%	7.971	11.9553	0.2431	32.408	64.4
CO	30, 11*	130	98.68%	5.928	7.0765	0.0929	34.331	12.8
NO	46	76	97.49%	5.694	51.5347	0.8317	6.707	12.3

Table 4.8: 2D rotation occlusion experiment *YZ* – data set 2, 1 view

Method	Maxima Err. (°)	Num. Modes	Expected Max. Val.	M_1 Height $\times 10^{-2}$	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
PO	4, 7*	278	98.92%	3.379	5.8261	3.7658	12.609	227.2
ZO	4, 7*	278	98.92%	3.379	5.8261	3.7658	12.609	229.0
CO	31, 12*	150	94.61%	2.441	5.8734	5.7332	19.098	25.0
BO	5, 7*	146	99.10%	3.621	8.0428	3.7333	27.027	26.3
NO	7, 9*	108	98.45%	1.970	15.1667	1.9615	16.853	25.1

Table 4.9: 2D rotation occlusion experiment *YZ* – data set 1, 2 views

Method	Maxima Err. (°)	Num. Modes	Expected Max. Val.	M_1 Height $\times 10^{-2}$	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
PO	9	187	99.36%	6.137	65.5930	1.9071	9.786	187.7
ZO	9	187	99.36%	6.137	65.5930	1.9071	9.786	197.1
BO	2	149	99.10%	5.803	61.0648	0.0111	11.889	22.3
CO	7	120	99.51%	3.867	73.2095	2.0980	8.346	22.4
NO	7	53	99.40%	3.344	80.0587	1.2158	5.599	22.6

Table 4.10: 2D rotation occlusion experiment *YZ* – data set 2, 2 views

4.8.5 Measurement Function Experiments

The proposed graph based measurement functions are compared with edge measurement functions commonly used by other authors in this section. The compared methods are designated:

Discrete Line Search (DLS)

The first method examined is a discrete line search (DLS) method, discussed in Section 2.1.2 and used by authors such as Sminchisescu and Triggs [90]. In this approach search lines are cast at normals to each link's projected edges. Edge features are points where these search lines cross image points designated as edges. The edge map used is based on the combined intensity and gradient method discussed in Section 3.3, which was shown in Figure 3.8(b). As in [90], the probability of each measurement line is based on a Leclerc distribution, with $\sigma = 7$, and non-detection rate 25%, chosen to give good results. These probabilities were combined using a sum of squared differences approach, as used by by Deutscher *et al.* [31].

Nearest Edge (NE)

The second method is a nearest edge distance (NE) method, discussed in Section 2.1.2 and used by authors such as Deutscher *et al.* [31]. In this method the "edge distance map" shown previously in Figure 2.3(b), was linearly interpolated to produce a likelihood for each sampled point. The same edge detector was used as was used for the DLS method. The "edge distance map" was generated using a Gaussian distribution, again with $\sigma = 7$, and non-detection rate 25%, chosen to give good results. These probabilities were combined using a sum of squared differences approach, as used by Deutscher *et al.* [31].

Cost Paths (CP)

This is the proposed graph based method described in Section 4.3, where consecutive trellises are not joined.

Joined Cost Paths (JCP)

This is the proposed graph based method described in Section 4.4, where consecutive trellises are joined to form a single trellis for each occluding contour.

Region Consistency (RC)

For thoroughness the region consistency measure, discussed in Section 3.5.2, is evaluated as it is used in later results. The performance of this scheme is largely dependant upon the consistency (case transition smoothness) of the link joining function described in Section 3.1.3.

No Edge Measurement (NONE)

The no edge measurement method is used to determine the underlying time taken by the pre-edge measurement steps: link projection, link joining, and occlusion map generation, so that they may be separated from the time taken to evaluate the edge measurement function itself.

Of note is that the same image gradient was implicity used by all methods. The results for each edge measurement method for Experiment 1 are given in Tables 4.13–4.16, while the results for each measurement method for Experiment 2 are given in Tables 4.17–4.20. These results are summarized in Tables 4.11 and 4.12.

Method	Pos. Err. mm (global max)	Pos. Err. mm (max by area)	Number of Modes	Expected Max. Val %	M_1 Height	M_1 Area %
DLS	48.09 ± 23	32.69 ± 14.7	445.8 ± 92.8	78.43 ± 4.58	0.36 ± 0.05	10.09 ± 2.53
NE	36.86 ± 16.8	36.86 ± 16.8	32.75 ± 14.7	72.21 ± 18.2	0.35 ± 0.37	85.25 ± 14.9
RC	38.83 ± 24.7	38.83 ± 24.7	30.25 ± 20.8	84.37 ± 15.2	0.16 ± 0.01	96.03 ± 4.08
CP	21.78 ± 3.42	21.78 ± 3.42	16.00 ± 11.0	96.48 ± 1.44	0.13 ± 0.01	98.36 ± 1.99
JCP	17.65 ± 3.93	17.65 ± 3.93	9.75 ± 8.06	96.75 ± 1.80	0.14 ± 0.01	99.00 ± 1.12

Table 4.11: Experiment 1 Translation, Measurement Function Summary

Method	Pos. Err. ° (global max)	Pos. Err. ° (max by area)	Number of Modes	Expected Max. Val %	M_1 Height	M_1 Area %
DLS	24.18 ± 41.4	15.67 ± 21.6	419.8 ± 62.1	96.39 ± 3.7	0.21 ± 0.08	24.00 ± 10.3
NE	25.21 ± 37.5	15.41 ± 18.1	202.0 ± 19.2	97.66 ± 2.8	0.27 ± 0.29	20.92 ± 12.1
CP	20.56 ± 17.9	5.65 ± 1.60	170.0 ± 13.9	98.71 ± 0.6	0.06 ± 0.02	37.61 ± 22.1
JCP	19.70 ± 18.8	6.33 ± 3.25	166.8 ± 22.8	98.79 ± 0.8	0.05 ± 0.02	39.30 ± 14.9
RC	26.18 ± 30.1	24.94 ± 27.7	52.25 ± 22.2	79.66 ± 19	0.14 ± 0.02	73.80 ± 19.6

Table 4.12: Experiment 2 Rotation, Measurement Function Summary

These results show that the proposed graph based measurement schemes significantly outperformed the other methods. While there was a large number of modes reported in Experiment 2 by all edge based measurements, the primary mode of the proposed CP and JCP methods is almost twice the size of the DLS and NE methods. This then also indicates there is little difference in the modality of the measurement schemes in regions where the object is not present. The region consistency (RC) measurement produced the fewest modes in experiment, where the search range was large, and had the largest primary mode area. The RC measurement also had in generally a larger error than the edge measurement methods. These results match the region measurement generalizations made in Section 2.1.4.

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	$50, 51^*$	518	72.39%	0.2886	4.1761	10.9691	6.5484	36.5
NE	42	50	50.67%	0.0174	93.8731	3.6072	1.8528	22.6
RC	16	34	96.14%	0.1493	98.4903	0.4984	0.3725	–
CP	20	25	94.93%	0.1186	97.9806	1.5923	0.8627	10.0
JCP	21	15	94.10%	0.1237	98.3433	3.9136	0.6862	13.3

Table 4.13: 2D translation measurement experiment *YZ* – data set 1, 1 view

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	$74, 24^*$	384	77.36%	0.4062	1.4312	0.3505	11.1656	39.8
RC	65	57	65.08%	0.1676	90.6578	4.5622	2.5096	–
NE	55	34	68.10%	0.7415	81.3548	39.9436	7.9208	24.5
CP	27	26	95.78%	0.1362	95.6867	10.7672	2.2351	10.6
JCP	16	18	97.20%	0.1427	97.7551	7.0193	1.3528	13.2

Table 4.14: 2D translation measurement experiment *YZ* – data set 2, 1 view

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	$50, 38^*$	532	81.70%	0.3606	7.4895	4.1712	10.1951	93.9
NE	36	14	75.41%	0.0531	99.6765	0.0369	0.1765	62.2
RC	20	8	96.78%	0.1753	99.8039	0.4621	0.1078	–
CP	21	7	98.23%	0.1453	99.9118	0.0446	0.0294	18.9
JCP	20	1	97.90%	0.1474	100.0000	–	–	25.8

Table 4.15: 2D translation measurement experiment *YZ* – data set 1, 2 views

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	18	349	82.26%	0.3767	12.4400	6.0428	10.0284	79.0
NE	14	33	94.68%	0.5821	66.1112	4.7910	31.6145	47.8
RC	55	22	79.48%	0.1435	95.1573	2.8994	2.9801	–
CP	20	6	96.98%	0.1281	99.8432	0.8871	0.0686	18.9
JCP	13	5	97.81%	0.1294	99.9020	0.0415	0.0490	24.6

Table 4.16: 2D translation measurement experiment *YZ* – data set 2, 2 views

Method	Maxima Err. (°)	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	4	441	92.67%	0.1482	25.1800	7.0314	5.1250	40.4
NE	2	209	99.74%	0.0082	15.2804	3.2447	12.8458	25.1
JCP	30, 6*	177	98.07%	0.0479	4.1682	0.9515	35.0133	14.6
CP	30, 7*	161	97.90%	0.0550	4.3672	3.4663	25.8336	11.0
RC	5	82	95.72%	0.1170	62.2584	1.0386	14.7878	–

Table 4.17: 2D rotation measurement experiment *YZ* – data set 1, 1 view

Method	Maxima Err. (°)	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	4, 48*	410	99.62%	0.3020	11.7374	2.1032	15.2236	40.2
NE	81, 42*	220	97.44%	0.5150	0.5873	4.7133	17.4214	24.4
JCP	41, 10*	194	98.88%	0.0772	5.0303	0.0376	34.1228	12.9
CP	41, 5*	177	98.62%	0.0856	13.8310	0.0659	32.8912	10.8
RC	69, 64*	45	55.05%	0.1663	24.5548	0.1365	52.3115	–

Table 4.18: 2D rotation occlusion experiment *YZ* – data set 2, 1 view

Method	Maxima Err. (°)	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	86, 8*	488	93.81%	0.1378	0.1516	1.6970	17.5256	92.6
NE	11	204	93.83%	0.0354	12.1826	23.0950	9.7196	54.8
CP	7, 7*	156	99.20%	0.0442	7.0955	3.6314	21.7412	22.2
JCP	5, 7*	146	99.10%	0.0362	8.0428	3.7333	27.0273	29.7
RC	7	29	94.46%	0.1408	92.7435	17.7551	3.3441	–

Table 4.19: 2D rotation measurement experiment *YZ* – data set 1, 2 views

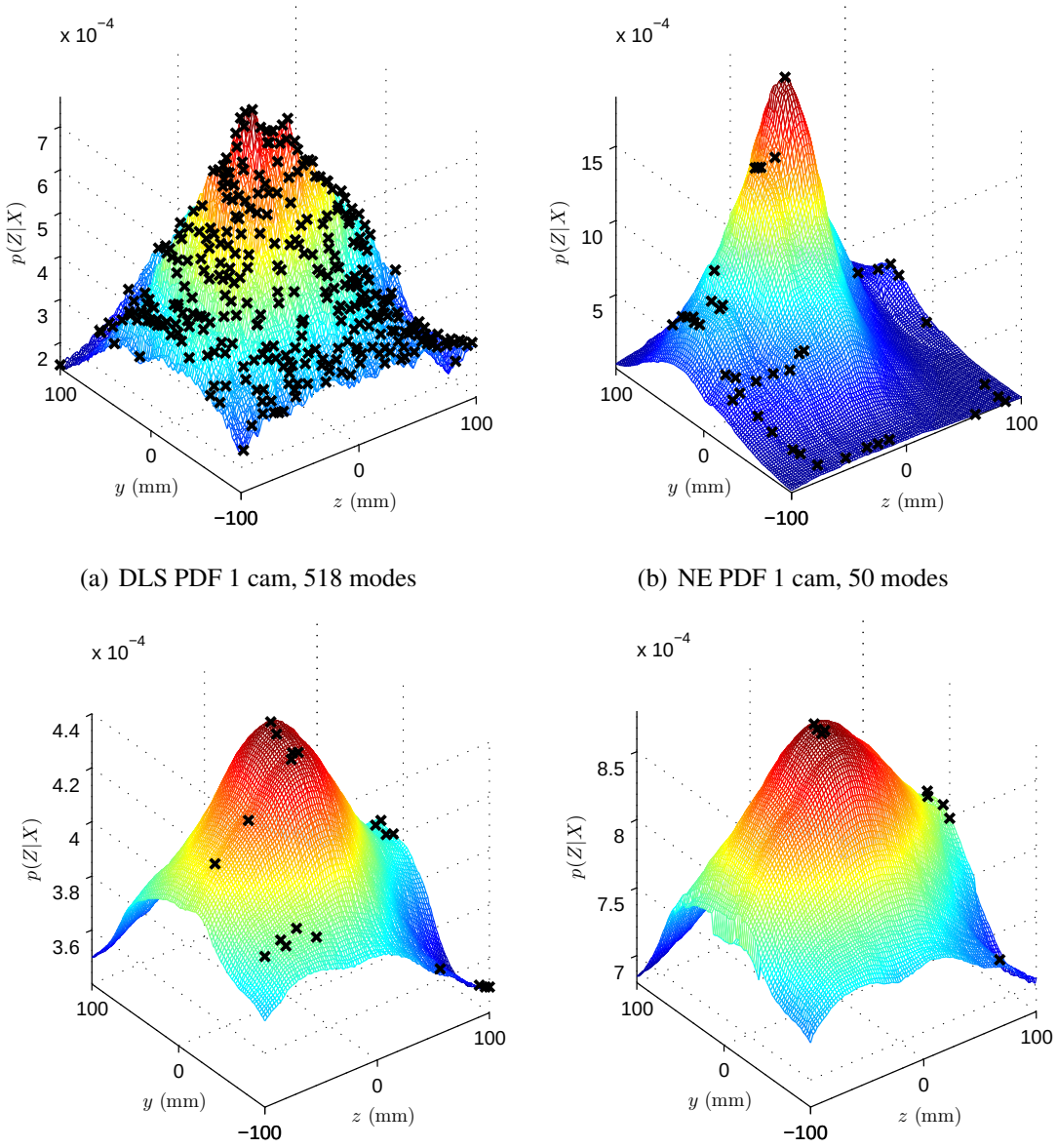
Method	Maxima Err. (°)	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	3	340	99.48%	0.2688	38.0637	4.1342	5.1345	77.4
CP	4	186	99.11%	0.0703	69.9886	9.6141	5.5513	18.3
NE	7	175	99.62%	0.5318	38.8121	4.4796	13.3479	47.7
JCP	2	150	99.10%	0.0580	61.0269	0.0111	11.8890	24.2
RC	24	53	73.39%	0.1380	87.8742	1.6761	2.2167	–

Table 4.20: 2D rotation occlusion experiment *YZ* – data set 2, 2 views

No study was performed to determine if the performance of the DLS and NE methods could be improved by varying the probability distribution parameters, however equivalently no study was performed to determine the effect of changing the edge penalty, M_P for the CP and JCP methods. Given the input images, it is very difficult to extract the true edge features around the low contrast legs for the first data set, or the low contrast hair for the second data set. As a result, the non-detection rate was set quite high (25%) for both methods. The sparsity of edge features highlights the problems with methods reliant upon successful extraction of a set of edge features. The graph based approaches perform better in these low contrast regions, as they are not dependent upon the discretization of edge features, and are less sensitive to spurious edge features as they enforce a consistent measurement along the entire length of the edge. The author was surprised to find that the NE method was on average more accurate than the DLS approach. Given these results, and that the NE implementation is more efficient, there seems little reason to use the DLS edge measurement.

The results shown in Tables 4.13–4.16 show a decrease in the number of modes for the DLS, NE, and CP methods compared to the results presented by Smith and Lovell [95]. This decrease in modality was due to the more rigorous calculation of hypothesized edge locations and the self occlusion map as discussed in Section 3.1.3. The author was quite surprised at how much reworking these two areas improved the modality of the observational likelihoods. For the NE method specifically, a large decrease in the modality was achieved by interpolating the edge distance map, while Smith and Lovell rounded each sample point on a hypothesized edge to the nearest pixel.

The average time of each measurement function evaluation included the region consistency evaluation, although the region score was not included in the probability of the edge methods. The region consistency evaluation was expected to be fast however since the occlusion map is already calculated. Experiments were performed in the MATLAB®environment, on a single core of a 64 bit 2.5Ghz Intel Xeon machine, running Microsoft Windows Enterprise edition, with a variable number of users logged on during the experiments. Care must be taken when interpreting these timing results. The CP and JCP measurement functions were implemented using 'C' routines (compiled to MATLAB 'mex' functions), whereas the DLS and NE measurement functions used pure MATLAB implementations. It is expected that under a fairer timing comparison, the NE method would have reported the lowest average time. These timings then simply show that the JCP approach presented here can be implemented significantly faster than a MATLAB implementation of the NE method, and that the JCP method is approximately twice as computationally intensive as the CP method.



(a) DLS PDF 1 cam, 518 modes

(b) NE PDF 1 cam, 50 modes

(c) CP PDF 1 cam, 25 modes

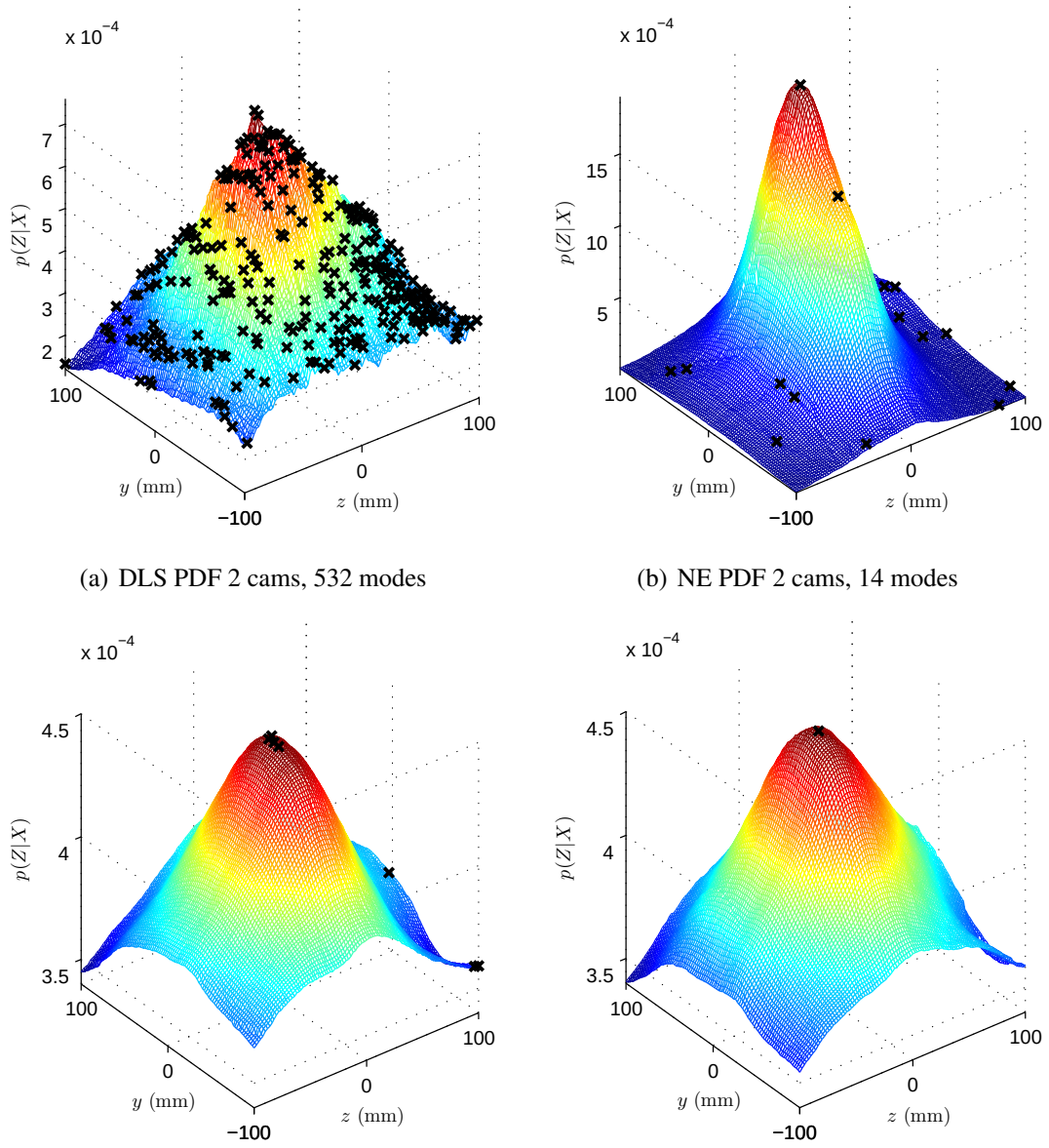
(d) JCP PDF 1 cam, 15 modes

FIGURE 4.14: PDFs for various edge evaluation methods using one camera

Figures 4.14 and 4.15 show the likelihoods for each method for Experiment 1 on Data set 1 in the monocular case and two camera cases, where the location of each mode is shown by a black cross.

4.8.6 Three Dimensional Measurement Function Experiment

In this section, Experiment 1 was modified to include the dimension corresponding to the base link x translation. The search range was reduced to ± 75mm at a 2mm resolution (438976 points) in all three dimensions, so that the maximum magnitude of the base translation was approximately that used in Experiment 1. Translation in the x direction is problematic for the single view case as the lone camera lies in this direction, making it an *uncertain* direction. As such this experiment was designed to test both a higher dimensional state space, and in the single view case the modality of the uncertain



(a) DLS PDF 2 cams, 532 modes

(b) NE PDF 2 cams, 14 modes

(c) CP PDF 2 cams, 7 modes

(d) JCP PDF 2 cams, 1 mode

FIGURE 4.15: PDFs for various edge evaluation methods using two cameras

direction when only a single mode would ideally be detected.

Tables 4.22– 4.25 show the PDF information for the CP, JCP, and RC evaluation schemes for this experiment, with these results summarized in Table 4.21.

Method	Pos. Err. mm (global max)	Pos. Err. mm (max by area)	Number of Modes	Expected Max. Val %	M_1 Height	M_1 Area %
DLS	79.38 ± 30.7	64.59 ± 11.7	4097 ± 919	75.17 ± 8.98	0.37 ± 0.08	68.76 ± 23.9
NE	45.72 ± 20.8	45.72 ± 20.8	202 ± 119.2	69.26 ± 17.5	0.36 ± 0.38	95.40 ± 4.34
RC	59.81 ± 34.8	59.81 ± 34.8	88.25 ± 64.8	83.03 ± 15.2	0.16 ± 0.02	99.71 ± 0.19
CP	22.13 ± 4.77	22.13 ± 4.77	76.25 ± 73.6	96.22 ± 1.52	0.15 ± 0.02	99.10 ± 1.40
JCP	21.87 ± 4.42	21.87 ± 4.42	50.25 ± 60.1	96.38 ± 1.81	0.15 ± 0.02	99.64 ± 0.59

Table 4.21: 3D Translation Experiment, Measurement Function Summary

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	81	4955	61.94%	0.2798	73.2885	10.1505	5.0224	41.2
NE	51	210	48.15%	0.0185	97.7534	13.5065	0.8078	25.9
RC	64	162	93.69%	0.1543	99.6952	2.1707	0.0786	–
CP	24	98	94.26%	0.1233	99.3781	4.9175	0.5173	11.3
JCP	20	71	94.05%	0.1260	99.7827	2.3943	0.0952	14.9

Table 4.22: 3D translation measurement experiment *XYZ* – data set 1, 1 view

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	$122,63^*$	3778	77.26%	0.4483	10.1892	9.3540	35.2748	35.2
CP	27	171	95.80%	0.1568	97.0445	3.5528	1.1060	9.6
NE	70	137	65.39%	0.7574	94.0031	56.7787	2.7505	21.5
JCP	28	126	97.36%	0.1635	98.7724	10.0059	0.6971	11.0
RC	102	122	63.57%	0.1870	99.5922	0.7155	0.0695	–

Table 4.23: 3D translation measurement experiment *XYZ* – data set 2, 1 view

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	53	4707	80.08%	0.3446	91.9383	3.8572	0.4399	71.8
NE	43	95	73.16%	0.0552	99.8494	1.4457	0.0695	42.9
RC	17	24	96.50%	0.1683	99.9790	0.8227	0.0077	–
CP	21	18	97.61%	0.1511	99.9893	0.1485	0.0021	17.7
JCP	21	1	98.19%	0.1536	100.0000	–	–	24.8

Table 4.24: 3D translation measurement experiment *XYZ* – data set 1, 2 views

Method	Maxima Err. mm	Num. Modes	Expected Max. Val.	M_1 Height	M_1 Area %	M_2 Height %	M_2 Area %	time (ms)
DLS	62	2946	81.39%	0.4205	74.5380	8.3749	6.7482	77.7
NE	19	366	90.33%	0.5991	90.0031	3.8050	8.9984	46.6
RC	57	45	78.37%	0.1367	99.5619	2.5109	0.2777	–
CP	16	18	97.21%	0.1524	99.9886	0.0698	0.0021	18.3
JCP	18	3	95.94%	0.1509	99.9975	0.0414	0.0023	24.1

Table 4.25: 3D translation measurement experiment *XYZ* – data set 2, 2 views

From Tables 4.24 and 4.25, the number of modes found using the JCP method decreases in the two camera case (due to the reduced search range) as an extra dimension is added to the state space. This gives the expectation that the number of modes may not increase exponentially as the state space dimensionality increases using the JCP approach, or at least not in regions "close" to the true object state.

In the monocular case, the number of modes did increase significantly due to the uncertain direction using the JCP approach. Tables 4.22 and 4.23 show that the global maxima was still reasonable well localized however, with the error less than 3cm for both data sets. This is more evidence that although a truly local optimizer may not perform well in localizing this direction, that the direction may be resolvable by a more robust optimizer. The optimizer proposed later in Chapter 6 is designed to exploit this knowledge.

4.9 Shortest Paths and Gradient/Hessian Approximations

To calculate an approximation to the Gradient or Hessian using the shortest path, it is assumed that the shortest path remains unchanged when the object's state is perturbed slightly, however the path's *cost* does change due to the edge penalty function. This is similar to the assumption made by Sminchisescu [86], where it is assumed that the feature set along a search line remains unchanged, only the distance of each feature along the search lines changes. It is asserted that this is a better assumption in the shortest path case, because it is less dependent upon the location of sample points. Inherent in this assumption is that relevant self occlusions remain unchanged, which is necessary because as discussed in Section 2.3.4, the self occlusion function is not differentiable everywhere.

To perform this approximation, the measurement function (using the aforementioned assumptions) for the ith unoccluded trellis vertex on the edge row is rewritten as a functional composition:

$$M_i(\mathbf{x}) = M_P\left(P_v^i\left(\mathbf{r}_i(\mathbf{x})\right)\right) \tag{4.58}$$

where $\mathbf{r}_i(\mathbf{x})$ denotes the image coordinates of the ith unoccluded trellis edge vertex as a function of the model state, $P_i^v(.) : \mathbb{R}^2 \rightarrow \mathbb{R}$ projects an image point onto the vth trellis column, and $M_P(.)$ is the trellis penalty function given in Equation 4.45.

The 1^{st} and 2^{nd} order partial derivatives of $\mathbf{r}_i(\mathbf{x}) = [r_i^x, r_i^y]^T$:

$$\frac{\partial \mathbf{r}_i}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial r_i^x}{\partial \mathbf{x}_1} & \cdots & \frac{\partial r_i^x}{\partial \mathbf{x}_d} \\ \frac{\partial r_i^y}{\partial \mathbf{x}_1} & \cdots & \frac{\partial r_i^y}{\partial \mathbf{x}_d} \end{bmatrix}^T \tag{4.59}$$

$$\frac{\partial^2 r_i^k}{\partial \mathbf{x}^2} = \begin{bmatrix} \frac{\partial^2 r_i^k}{\partial \mathbf{x}_1^2} & \cdots & \frac{\partial^2 r_i^k}{\partial \mathbf{x}_d \partial \mathbf{x}_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 r_i^k}{\partial \mathbf{x}_1 \partial \mathbf{x}_d} & \cdots & \frac{\partial^2 r_i^k}{\partial \mathbf{x}_d^2} \end{bmatrix}^T, k \in \{x, y\} \tag{4.60}$$

are calculated numerically as $\mathbf{r}_i(\mathbf{x})$ is highly non-linear. The derivatives of $P_i^v(\mathbf{r}_i)$ can be solved analytically:

$$\frac{\partial P_i^v}{\partial \mathbf{r}_i} = [\hat{v}_i^x, \hat{v}_i^y]^T \tag{4.61}$$

$$\frac{\partial^2 P_i^v}{\partial \mathbf{r}_i^2} = \mathbf{0} \tag{4.62}$$

where $[\hat{v}_i^x, \hat{v}_i^y]^T$ is the unitized trellis column direction. The derivatives of and $M_P(P_v^i)$ are also available analytically:

$$\frac{\partial M_P}{\partial P_v^i} = \frac{2P_v^i}{\sigma^2} e^{\frac{-\left(P_v^i\right)^2}{\sigma^2}} \tag{4.63}$$

$$\frac{\partial^2 M_P}{\partial \left(P_v^i\right)^2} = \frac{2\sigma^2 - 4\left(P_v^i\right)^2}{\sigma^4} e^{\frac{-\left(P_v^i\right)^2}{\sigma^2}} \tag{4.64}$$

Note that the derivatives of $-\log\left(M_P(P_v^i)\right)$ have a simpler form than $\frac{\partial M_P}{\partial P_v^i}$.

The gradient $\mathbf{J}_i$ and Hessian $\mathbf{H}_i$ contributions for unoccluded trellis edge vertex $\mathbf{r}_i$ are then:

$$\mathbf{J}_i = \frac{\partial \mathbf{r}_i}{\partial \mathbf{x}} \frac{\partial P_i^v}{\partial \mathbf{r}_i} \frac{\partial M_P}{\partial P_v^i} \tag{4.65}$$

$$\begin{aligned} \mathbf{H}_i &= \sum_k^{x,y} \frac{\partial^2 r_i^k}{\partial \mathbf{x}^2} \frac{\partial P_i^{v,k}}{\partial \mathbf{r}_i} \frac{\partial M_P}{\partial P_v^i} + \left(\frac{\partial \mathbf{r}_i}{\partial \mathbf{x}}\right) \left(\frac{\partial^2 P_i^v}{\partial \mathbf{r}_i^2} \frac{\partial M_P}{\partial P_v^i} + \frac{\partial P_i^v}{\partial \mathbf{r}_i} \frac{\partial^2 M_P}{\partial \left(P_v^i\right)^2} \left(\frac{\partial P_i^v}{\partial \mathbf{r}_i}\right)^T\right) \left(\frac{\partial \mathbf{r}_i}{\partial \mathbf{x}}\right)^T \\ &= \sum_k^{x,y} \frac{\partial^2 r_i^k}{\partial \mathbf{x}^2} \frac{\partial P_i^{v,k}}{\partial \mathbf{r}_i} \frac{\partial M_P}{\partial P_v^i} + \frac{\partial \mathbf{r}_i}{\partial \mathbf{x}} \frac{\partial P_i^v}{\partial \mathbf{r}_i} \frac{\partial^2 M_P}{\partial \left(P_v^i\right)^2} \left(\mathbf{1} + \frac{\partial P_i^v}{\partial \mathbf{r}_i}\right)^T \left(\frac{\partial \mathbf{r}_i}{\partial \mathbf{x}}\right)^T \end{aligned} \tag{4.66}$$

where the Hessian was derived by recursing the derivative product rule. These approximations could be further simplified in the usual case where the trellis edge row is straight.

4.10 Shortest Paths and Modelling Deformable Objects

Given a training set of cost trellises it is possible to learn a set of possible paths which define the object's projected edge. Typically the link surface models used in visual tracking are simplified in that surface deformation is not modelled. In the case of human body tracking, the dimensionality of the state space is dauntingly high without adding surface deformation parameters. The set of acceptable cost paths for a link is theoretically the set of projected edges of the link under realistic deformation

(*e.g.* muscle flex, dynamic (loose) clothing motion) in the current object state. A set of acceptable paths can be learnt from a training set and used to model these deformations. The parameters to be learnt are: acceptable starting and finishing locations (sources and sinks) for the trellises, and the maximum allowable distance both to the interior and exterior of the object's (modelled) projected edges. This section is included purely as a discussion given this thesis work aims to avoid using learnt feature models.

During the training process, the training set trellises for a link's edge are resized (linearly interpolated) such that they all have the same number of columns. The unconstrained shortest paths across all of the trellises in the training set are found. Outlier removal is performed leaving only trellises with minimum path energies in the middle, say 80%, of values.

The first estimate of acceptable trellis width, sources, and sinks is obtained by the range of the middle, say 60%, of the shortest paths. The shortest path across each trellis subject to the new acceptable path parameters is then calculated. This acceptable path parameter set is then given a score based on the number of trellises whose new shortest path cost is 'close enough' to the unconstrained shortest path cost, as well as the number of vertices the shortest path can utilize. The 'close enough' cost is set be less than, say 105%, of the original cost. The function used to generate the score is $S = \frac{N_p}{\sqrt{N_s}}$, where N_p is the number of 'close enough' paths, and N_s is the number of vertices the constrained path can utilize.

Bi-modality of the 'close enough' paths is then checked by finding the principal eigenvector of these paths and splitting the paths into two groups based on their value along this eigenvector. The statistical fitting procedure described above is performed on both of these groups, and the results merged (union of the source and sink locations for both groups, maximum and minimum interior and exterior edge distances for both groups) to form an alternative path parameter set. This is then scored in the same manner as above. Should it present an improvement, the old parameter set is discarded and this new one is split again until there is no improvement. This technique of splitting via the principal eigenvector is an adaption of work presented by Evans, Alder, and deSilva [36].

The remaining trellises (those whose shortest path is not yet judged 'close enough') are split in the same fashion and the same process repeated for each group of the groups. This continues until there is greater than a threshold percentage, say, 95%, of trellises whose constrained shortest path cost has been judged to be 'close enough' using the new path parameter set.

4.11 Discussion

In this chapter, graph based methods as used in image segmentation problems have been used to perform edge measurements in visual tracking problems. Since self occlusions are not generally modelled in segmentation problems, methods have been proposed to allow the computation of the shortest path in the presence of occluded trellis vertices. This was extended to forming graphs around entire occluding contours, rather than a link by link treatment. It was shown that using the graph based approach improved the conditioning of the observational likelihood, in terms of both accuracy and number of local maxima (smoothness), compared to other edge measurement methods. These improvements are the result of not relying upon the successful extraction of edge features, and on the dense set of measured pixels that are inherently used in the graph based approach.

While the results presented in Section 4.8 show that the joined cost path approach produces better conditioned (smoother) observational densities than the more commonly used methods, it still has some undesirable properties. The first undesirable property is linked to the problem of selecting an edge feature threshold when using edge methods reliant on using discrete edge features, such as in the discrete line search or nearest edge approaches. In these methods, once a suitable edge feature threshold has been found, all edges have the same 'strength' as a result of the discretization. As the graph based approach does not perform this discretization, the only avenue to control the difference between edge 'strengths' is via the cost metric from Section 4.5.

For example, consider two model states, X_1 and X_2, using a single occluding contour with costs, distances, and ratios C^i, D^i, $R^i = \frac{C^i}{D^i}$, $i \in \{1, 2\}$ from Equation 4.47. The cost and distance of traversing the shortest path around the occluding contour is then sum of the cost of traversing the sections of the graph formed around the different links. Let $C^i = \sum_{j=1}^{L} C^i_j$ and $D^i = \sum_{j=1}^{L} D^i_j$, where L is the number of links in the kinematic chain, and C_j and D_j are the cost and distance of traversing the occluding contour in the regions formed around link j. The situation can arise where X_1 has a worse measurement score than X_2, $R^1 > R^2$, despite every individual link measuring equally or better, $R^1_j \leq R^2_j \; \forall \; j \in \{1, ..., L\}$, with $R^i_j = \frac{C^i_j}{D^i_j}$. This occurs when link j has a worse measurement score than average, $R^1_j > R^1$, and it has a larger path length in state X_1 than in state X_2, $D^1_j > D^2_j$. This is generally undesirable behaviour, particularly if link j is an end of the kinematic chain. A local optimizer would then move the link from its (individual) 'best fit' position, to another position simply because the path length is smaller (assuming the cost isn't increased by much). This situation is most common when different links contrast to different extents with the background.

While this could potentially be resolved by setting $R = \frac{1}{L} \sum_{i=1}^{L} R_i$, it is still useful to weight the measurement of each link by the number of points used to make the measurement. Another potential

remedy is to adjust the cost metric from Section 4.5; reducing the differences between the cost values of all of the true edges (balancing the edge strengths) will reduce the effect of this problem. This is not trivial however, and is also an issue in image segmentation. Appleton and Talbot [7] present an optimal solution to finding the shortest path around points in general undirected graphs. As this is an optimal solution, it is known that this algorithm will produce the shortest path for a given cost metric, but devising a general cost metric (Equation 4.43) which reliably gives the desired solution remains an unsolved problem.

One possible solution might be to utilize the colour group information discussed in Section 3.5.2, which is the prior knowledge of the links that are the same colour. Cost and distance scores for each colour group could be calculated, using the cost and distances of traversing the sections of each graph formed around each link in the colour group. Separate edge likelihoods could then be calculated for each colour group. This is based on the assumption that because all of the links contributing to the score for a particular colour group are the same colour, then the object/background contrast will be similar, and hence the edge 'strengths' are inherently balanced within each colour group. Another approach involves modifying the self occlusion likelihood from Section 3.1.3, so that it rewards hypothesized states in which link's have long measurable (un-occluded) edges.

The second problem with the joined cost path method relates to the joining method given in Algorithm 5. Not relying on edge feature sets, and noting that the shortest path will not change much as a trellis vertex becomes occluded, removes the first order measurement function discontinuities discussed in Section 2.3.4. Discontinuities are still present in the measurement function in states on the boundary of where consecutive trellises on an occluding contour can be joined however. Slight perturbations of these states can cause significant differences in the vertex locations of the occluding contours trellis. While the joining algorithm is designed to minimize these difference in vertex locations between states on either side of this boundary, this discontinuity is unavoidable in the formulation presented here. It was noted earlier that fewer interior trellis rows are used than exterior rows to facilitate the joining of trellises because the occluding contour is generally convex.

A possible solution to this is to dynamically control the number of used rows in local regions of an occluding contour's trellis. As such, if the join failed on row i of the trellis, vertices could be placed arbitrarily and given weights (costs) $M(i,j) = \infty$. More interior rows could then be used, which may increase the size of the zone of attraction of the edge measurement function. This retains the trellis structure of the graph, allowing the computationally efficient computation of the shortest path. Another approach would be to adopt a more general graph structure for the occluding contour, possibly such that each (equivalent) row of the new graph of has a different number of vertices. This has

another advantage that the spacing between the vertices of all rows and columns could be kept constant in all regions of the graph. In the current formulation, vertices are generally closer together on interior rows, and the vertex spacing varies across regions of the occluding contour's trellis. Computing the shortest path across a more general graph may be more computationally expensive however. Future research could also investigate making the vertex weight $M(i, j)$, and 'distance' $D(i, j)$, from Section 4.6, a function of the image distance between connected vertices.

5

State Spaces

The state space in which the tracking problem is formulated is discussed in this Chapter. By far the most commonly used state space used in tracking problems is a state space comprised of the angles between the links in the kinematic chain. Such a state space has the advantage that it is compact in its dimensionality, *i.e.*, the entire range of possible human postures is capture by as few dimensions as possible. The properties of this space are analysed, with the relationship between the state variables and the location of each link in an image focussed upon. An alternative state space comprised primarily of the Cartesian coordinates of the ends of rotating links is proposed, as the relationship between the state variables and the location of each link in an image is much more linear in this alternative state space. To the author's knowledge, no visual tracker using a full kinematic chain to model the human body has used a state space based on these Cartesian coordinates, however visual trackers using representations of kinematic chains have used a similar state space formulation, e.g. Morris and Rehg [72], and Sigal, Bhatia, Roth, Black, and Isard [83].

The proposed Cartesian state space requires a higher dimensionality to represent a set of postures than the commonly used rotation based space. The geometrically consistent (obeying link length constraints) regions of the Cartesian state space capture the exact same range of poses however. As such the well acknowledged 'curse of dimensionality' is not applicable when only the geometrically

consistent regions of the Cartesian state are considered. Simple methods are given to project a geometrically inconsistent state onto a consistent state, as well as to control measurement function curvature in 'implausible' directions.

The increased linearity between the state variables and the world (and image) location of a sampled point on the object is shown to improve the performance of several modules used in tracking problems, including: dynamic prediction, principal component analysis, and local optimization performance.

5.1 Motivation

The Cartesian coordinate based state space proposed in this section is motivated by the expectation that both the sampling and optimization processes will be more efficient in a state space where a small perturbation of the state vector causes only a (relatively) small change in the measurement function. This can be thought of as formulating the measurement function in a space such that modes in the observational likelihood have a simple geometric shape (topology), which can be more readily modelled than when modes have a highly curved ridge like shape, *i.e.* long curved 'tails'. This can be expressed more formally as reducing the magnitude of the higher order partial derivatives of the measurement function. While this does not necessarily change the number of modes in the observational likelihood[1], it does make each mode easier sample from. Furthermore, from an optimization perspective, such curved tails are problematic because if an improvement exists at a distance η in a particular direction, there may not be a possible improvement at a distance $\eta^* < \eta$, or $\eta^* > \eta$, which is highly problematic when the optimization problem is ill-conditioned and requires dampening.

For the visual tracking problem, formulating the state space such that a small change in the state vector causes only a small change in the measurement function, roughly translates to a small change in the state vector causing only a small change in the object's appearance in each image (a small perceptual distance). This perceptual distance can be quantified by assuming that a small change to the object's appearance yields only a small change in the set of image pixels measured during the measurement process, which in turn means only a small change in the measurement score for realistic measurement functions. Using $\mathcal{D}(X_i, X_j)$ to denote a measure of the similarity of the measured pixels when applying the measurement function to model states X_i and X_j, a desirable state space property

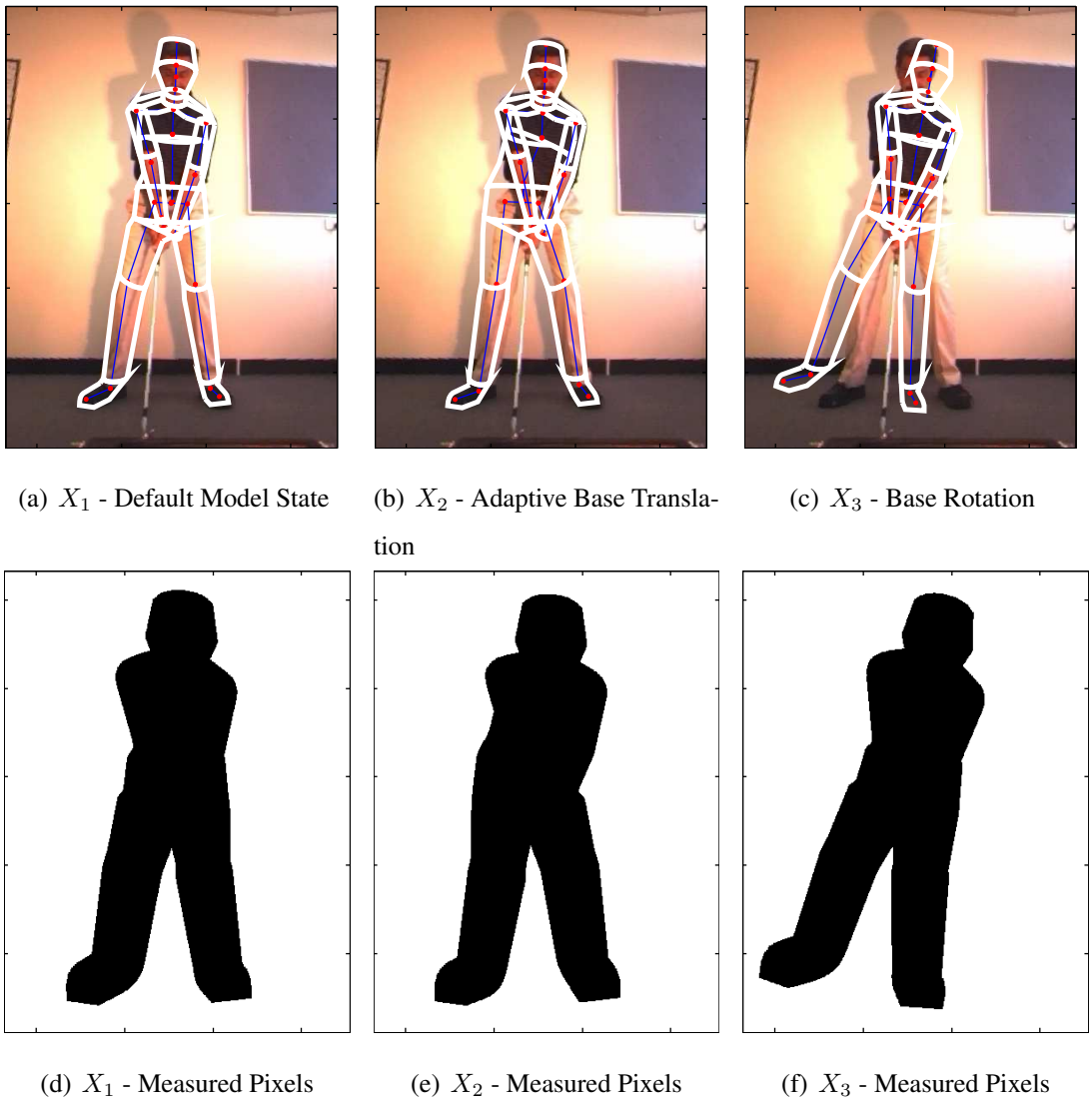
[1] Two spaces will have an equal number of local maxima if a global diffeomorphism exists between two spaces.

is then that $\forall X_i, X_j, X_k \in \mathbf{x}$:

$$\|X_j - X_i\| < \|X_k - X_i\|, \quad \textbf{if } \mathcal{D}(X_i, X_j) > \mathcal{D}(X_i, X_k) \tag{5.1}$$

$$\|X_j - X_i\| > \|X_k - X_i\|, \quad \textbf{if } \mathcal{D}(X_i, X_j) < \mathcal{D}(X_i, X_k) \tag{5.2}$$

where $\|X_j - X_i\| = (X_j - X_i)^T \mathbf{W}(X_j - X_i)$ for some known metric space $\mathbf{W}$. In a state space with this property, 'worm holes' such as used by Heap and Hogg [46], and 'kinematic jumps' as used by Sminchisecu and Triggs [91], would be unnecessary as the states at either end of the 'worm holes' or 'jumps' are necessarily close together in the state space.



(a) X_1 - Default Model State (b) X_2 - Adaptive Base Translation (c) X_3 - Base Rotation

(d) X_1 - Measured Pixels (e) X_2 - Measured Pixels (f) X_3 - Measured Pixels

FIGURE 5.1: Differing Model States and Measured Pixels

As an example of these properties, consider the model states X_1, X_2, and X_3, as shown in Figures 5.1(a–c) respectively, with sets of measured pixels shown in Figures 5.1(d–f) using the joined cost path edge measurement from Chapter 4 and region consistency from Section 3.5.2. Upon visual inspection states X_1 and X_2 are 'close', and they have a similar set of measured pixels, whereas state

X_3 is quite distant from both X_1 and X_2. If state X_1 is considered an 'interesting' state (*e.g.* in terms of probability), then it is likely that state X_2 will also be interesting, however there is no real expectation about how interesting state X_3 is. When sampling around state X_1, it is then more desirable to draw state X_2 than state X_3.

Using $\bar{\mathbf{r}}(X_i)$ to denote the set of measured pixels when applying the measurement function to model state X_i, a crude (ignoring link correspondences, measurement type correspondences, edge direction correspondences *etc*) measure of this measured pixel set similarity can be formulated as:

$$\mathcal{D}(X_i, X_j) = \frac{\mid \bar{\mathbf{r}}(X_i) \cap \bar{\mathbf{r}}(X_j) \mid}{\mid \bar{\mathbf{r}}(X_i) \cup \bar{\mathbf{r}}(X_j) \mid} \tag{5.3}$$

where $\mid . \mid$ used to denote the cardinality of the set. Table 5.1 shows the state distances in a rotation ($\mathcal{R}$) state space described in Section 3.1.1, and a Cartesian ($\mathcal{C}$) based state space presented later in Section 5.2, and the measurement pixel set similarity. To evaluate $\|X_j - X_i\|$, a metric $\mathbf{W}$ equating 2mm to 1° was used[2]. Note how comparatively distant states X_1 and X_2 are in the rotation state space, and that in the Cartesian state space model states X_1 and X_2 are much closer than states X_1 and X_3.

State i	State j	$\|X_j^{\mathcal{R}} - X_i^{\mathcal{R}}\| \times 10^{-4}$	$\|X_j^{\mathcal{C}} - X_i^{\mathcal{C}}\| \times 10^{-5}$	$\mathcal{D}(X_i, X_j)\%$
1	2	5.2552	0.5228	87.87
1	3	0.02	6.3996	53.87
2	3	5.2752	6.5822	56.26

Table 5.1: State space distances vs measured pixel set similarity

From a sampling point of view, it is very difficult to envision randomly drawing state X_2 by sampling around X_1 using a finite number of samples in the rotation space. This would require a sampling from highly complex probability distribution, which in the author's opinion could not be inferred from the 1st and 2nd derivatives of the measurement function as used by techniques such as the covariance scaled sampler [90].

The experiment performed in Section 4.8.5 showed that in the two camera case there is a monotonically non-decreasing path from X_2 to X_1 in the state space in which the experiment was performed. Assuming the existence of a local diffeomorphism between the experimental state space and the rotation state space, there also exists a monotonically non-decreasing path from X_2 to X_1 in the rotation state space. This path would be highly curved in the rotation state space, and very difficult for a local optimizer to follow. Furthermore, the single mode shown in the experiment would have a highly complex geometric shape (topology) in the rotation state space.

[2]The result is quite insensitive to this choice

For a local optimizer to follow this monotonically non-decreasing path, all update steps must preserve the lock of all of the links in the kinematic chain (excluding the hip). The position of each link is a highly non-linear function of all state variables which act upon any earlier link in the link's branch of the kinematic chain. In the rotation state space, it is impossible to give a reasonable estimate of the link's position without knowing all of the state variables upon which the link's position depends. In the case of the model used in this thesis, shown in Figure 3.1, there is a maximum of 20 state variables which may alter a link's position. This (and the nature of rotations) implies that the function describing a link's position may have significant derivatives up to at least order 19.

A Newton like optimizer assumes the objective function can be locally modelled as a quadratic function. This then assumes that in a local region (the *trust region*) the partial derivatives of the measurement function $\frac{\partial^3 \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j \partial \mathbf{x}_k} \approx 0$, $\forall \{\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k\} \in \mathbf{x}$, with the same being true for all higher order derivatives. As such only up to second order information can be used to try to preserve the lock of each link. Combining this with the assertion that in the rotation state space the measurement function contains significant derivatives up to order 19, gives a reason why an improvement may not be possible at any distance along a update direction calculated using a quadratic approximation, *i.e.* the *trust region* has zero size. The use of Cartesian joint locations as the parameters of the state vector removes much of the non–linear correlation between variables in the state space, although some is still present due to the need to preserve geometric (link length) constraints.

Sminchisescu [86] comments that no global diffeomorphism exists between $\mathbb{R}^3$ and SO_3, where SO_3 is the set of possible rotation matrices in $\mathbb{R}^3$, citing the different topologies of $\mathbb{R}^3$ and SO_3 as the primary reason for this – SO_3 is closed and bounded, while $\mathbb{R}^3$ is not. Practically this means that a small change in rotation angle parameters can not be guaranteed to produce a small change in the measured pixels, a source of *irregular* points in the observational likelihood.

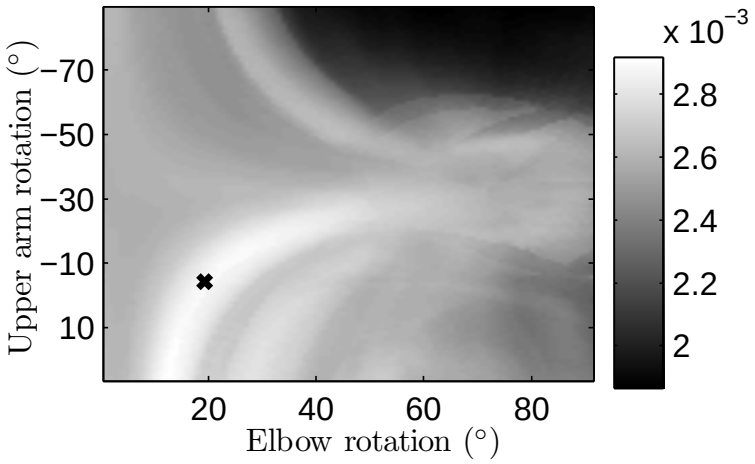


FIGURE 5.2: Observational Likelihood

Figure 5.2 shows the observational likelihood for the joined cost path measurement function in the arm rotation experiment, described in Section 4.8.5. Only the joint angles manipulating the wrist were varied in this experiment. The global maximum state is shown by the black cross. Measurements were performed using only a single camera in this experiment, leading to the observational likelihood having an 'uncertain direction' due to the monocular depth ambiguity. As can be seen from Figure 5.2, this uncertain direction causes the primary mode in the likelihood to have a long curved 'tail'. The covariance scaled sampler proposed by Sminchisescu and Triggs [90] was designed specifically to concentrate sampling in uncertain directions, predominately caused by monocular depth ambiguities. They modelled the observational likelihood locally with a Gaussian distribution. As can be seen from Figure 5.2, the region where this approximation holds is limited by the curvature of these tails.

In the proposed Cartesian space, uncertain directions caused by monocular depth ambiguities are straight lines in the state space. This may lead to the localized Gaussian model of the likelihood to hold over a larger region, facilitating the temporal propagation of the maxima.

5.2 Cartesian State Space Formulation

In the Cartesian state space formulation, the state space comprises primarily of the Cartesian coordinates of the end of each rotating link in the kinematic chain. Where possible joint rotations are clustered into groups. In this process some joint rotations are inherited by another link further down the kinematic chain. For example, the about axis rotation at the shoulder (rotating the upper arm), is inherited by the elbow. The elbow then has two rotational degrees of freedom allowing the wrist to be positioned anywhere on a sphere centred on the elbow. These clustered rotation groups are denoted $\mathcal{G}^*$ throughout this thesis. This approach is similar to that of Gravila and Davis [39], and Wachter and Nagel [103]. Rotations which can not be paired are treated as they are in the rotation state space formulation.

Figure 5.3 shows the model's degrees of freedom in the Cartesian based space. Rotation angles are kept for the base link because the base link does not inherit a variable coordinate frame. While this formulation contains 14 more dimensions than the rotation state space, geometric (link length) constraints mean the two state spaces have equivalent content, *i.e.* there are no geometrically consistent states that do not exist in the rotation state space. This avoids the so called *curse of dimensionality*, which only applies in situations where the content of the state space increases with its dimensionality.

Re-examining the discussion regarding the significance of higher order derivatives of the function to calculate each link's position presented in Section 5.1, it was expected that in the rotation space this function would have significant higher order derivatives. In the geometrically consistent regions of

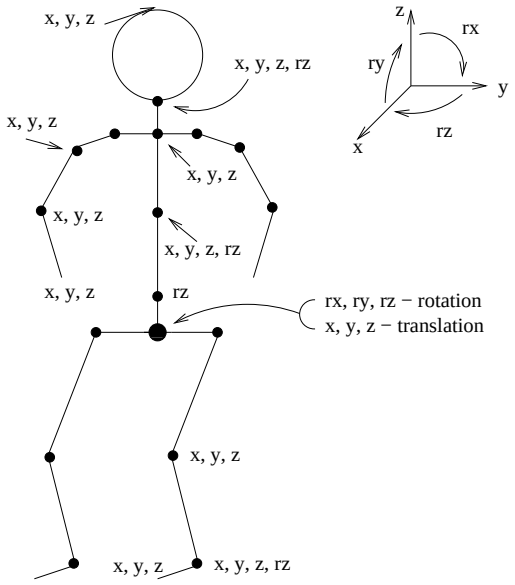


FIGURE 5.3: The 57 Cartesian DOFs of the human model

Cartesian space, and ignoring any about axis link asymmetry, the link's position is a trivial function of the six state variables describing the position of the start and end of link. This is suggestive that in geometrically consistent regions of the Cartesian space, the measurement function will have far fewer significant higher order derivatives. As a point in the Cartesian space is moved away from a geometrically consistent region however, it is expected that the magnitude of these higher orders derivatives will increase due to the non-linear constraint process.

5.2.1 Geometric Constraints

To enforce geometric (link length) constraints for a link $\mathbb{L}$, the link's unconstrained end position $\mathbb{L}_e^* \in \mathbb{R}^3$ is projected onto a *constraint* sphere. The constraint sphere is the sphere which defines all of the possible Cartesian positions the link's end $\mathbb{L}_e$ can take, given a known position for the link's start $\mathbb{L}_s$. This sphere is then centred at the link's start, with radius equal to the link's length $\mathbb{L}_l$. As such the geometrically constrained Cartesian position of the link's end is simply:

$$\mathbb{L}_e = \mathbb{L}_s + \mathbb{L}_l \frac{\mathbb{L}_e^* - \mathbb{L}_s}{||\mathbb{L}_e^* - \mathbb{L}_s||} \tag{5.4}$$

The constraint process is started at the base link, so that each link's start lies in a geometrically consistent position before it's end is constrained. This means that the constrained configuration may not be optimal, in the sense that another geometrically consistent configuration closer to the unconstrained configuration may exist. An optimal solution has not been used due to the computational cost of an iterative fitting process. Whether using an optimal an solution improves optimization performance has not been investigated.

In general, there are two joint angle solutions which position a link's end in the desired location on the constraint sphere. The surface normal of any point on the link's surface will have the same direction but different sign at these two solutions. This is unimportant when the link's surface is symmetric in x and y directions of the link's coordinate frame. As described in Section 3.1.2 however, the surface model used in this thesis includes surface axis offset parameters $\{o_x, o_y\}^T$ for the start and end, which violates this symmetry. As such, 'hidden' rotation variables are included in the state vector which indicate which of the two solutions should be used. In the tracking methods considered in this thesis, each new object hypothesis is an update of another hypothesis, and so the rotation solution closest to the original hypothesis's rotations is used. In many cases only one solution will satisfy anatomical joint constraints.

5.2.2 Standardised Coordinates

Since humans, and hence the articulated models used for human tracking, come in varying sizes, joint locations are taken to correspond to a standardised human model. This is computationally inexpensive as each link will point in the same direction, only the link's length changes. This allows consistency for training purposes, *i.e.* training data from different subjects can still be combined to learn: dynamic models, principal component approaches *etc*. These standardised coordinates were used for the two data sets used in the experiments performed later in Sections 5.4 and 5.5.

5.3 The Hessian in a Cartesian State Space

5.3.1 Implausible Directions

Any point in the Cartesian state space will have many implausible directions, where any point along the implausible direction projects back to the original constrained state. The measurement function gradient and curvature in these implausible directions is then necessarily zero. This is problematic for techniques such as covariance scaled sampling algorithm [90], and to a lesser extent for Newton like optimizers. For covariance based sampling, it is necessary to separate directions that have low curvature due to measurement ambiguities, and directions which have low curvature because they are implausible. As such, Hessian dampening is used to artificially add curvature in these implausible directions. Dampening is applied for each rotating link running in the $\{dx, dy, dz\}^T$ direction in $\mathbb{R}^3$,

which is represented by the Cartesian state space direction:

$$\mathbf{V} = \{\mathbf{0}, dx, dy, dz, \mathbf{0}\}^T \tag{5.5}$$

$$\mathbf{H} = \mathbf{H} + s\mathbf{V}\mathbf{V}^T \tag{5.6}$$

where the magnitude of s is large, but not large enough to effect the numerical stability of $\mathbf{H}$.

In practice the author has found it beneficial to rescale state space dimensions (and hence the Hessian) before applying dampening to the Hessian. This is done because much higher curvature values are expected in dimensions measured in radians than for those measured in millimetres, *i.e.* altering the object's state by one radian in a given dimension changes the measurement function significantly more than altering the object's state by one millimetre in another dimension. In the experiments presented throughout this thesis, the metric relating millimetres to radians was the same as the distances used for central differencing gradients, approximately $2mm \equiv 1^\circ$. As such, the Hessian is rescaled before any dampening as if all rotation dimensions were measured in half degrees. This keeps the order of magnitude of the curvature in a rotation dimension equivalent to a translational direction, facilitating dampening of the Hessian.

5.3.2 Computation

The Cartesian state space used in later experiments has $d_{\mathcal{C}} = 57$ degrees of freedom, as opposed to the $d_{\mathcal{R}} = 43$ for the rotation state space. An exact Hessian calculation using central differences requires $2(d^2 + d)$ measurement function evaluations, resulting in 3784 evaluations for the rotation space, and 7080 evaluations for the Cartesian space.

In the experiments performed later, knowledge of the kinematic chain and camera geometry were used to create a Hessian sparsity structure. The camera geometry was required for the self occlusion map, so that it was known which links could occlude each other. If the intersection of the set of link's manipulated by state dimension i, and the set of link's manipulated by state dimension j is empty, and these sets of links could not occlude each other[3], then $\frac{\partial^2 \mathbf{H}}{\partial \mathbf{x}_i \partial \mathbf{x}_j} = 0$. This sparsity structure reduces the number of measurement function evaluations required for numeric Hessian computation to 1916 for the rotation state space, and 3212 for the Cartesian state space.

The transformation from the Cartesian state space to the rotation state space can in principle be used to transform the Jacobian and Hessian from the rotation state space to the Cartesian state space. Using $\mathbf{J}^{\mathcal{C}\to\mathcal{R}}$ and $\mathbf{H}^{\mathcal{C}\to\mathcal{R}}$ to denote the Jacobian and Hessian of the transformation from the Cartesian

[3]The colour groups discussed in Section 3.5.2 should also be considered here

space to the rotation space,

$$\mathbf{J}^{\mathcal{C}\to\mathcal{R}}(i,j) = \frac{\partial \mathbf{x}_j^{\mathcal{R}}}{\partial \mathbf{x}_i^{\mathcal{C}}} \tag{5.7}$$

$$\mathbf{H}_k^{\mathcal{C}\to\mathcal{R}}(i,j) = \frac{\partial^2 \mathbf{x}_k^{\mathcal{R}}}{\partial \mathbf{x}_i^{\mathcal{C}} \partial \mathbf{x}_j^{\mathcal{C}}} \tag{5.8}$$

where $\mathbf{J}^{\mathcal{C}\to\mathcal{R}}$ and $\mathbf{H}^{\mathcal{C}\to\mathcal{R}}$ are calculated numerically. The Jacobian and Hessian in the Cartesian state space, $\mathbf{J}^{\mathcal{C}}$ and $\mathbf{H}^{\mathcal{C}}$, can then be approximated using the Jacobian and Hessian in the rotation state space, $\mathbf{J}^{\mathcal{R}}$ and $\mathbf{H}^{\mathcal{R}}$, by:

$$\mathbf{J}^{\mathcal{C}} \approx \mathbf{J}^{\mathcal{C}\to\mathcal{R}}\mathbf{J}^{\mathcal{R}} \tag{5.9}$$

$$\mathbf{H}^{\mathcal{C}} \approx \mathbf{J}^{\mathcal{C}\to\mathcal{R}}\mathbf{H}^{\mathcal{C}}\left(\mathbf{J}^{\mathcal{C}\to\mathcal{R}}\right)^T + \sum_{k=1}^{d_{\mathcal{R}}} \mathbf{H}_k^{\mathcal{C}\to\mathcal{R}}\mathbf{J}^{\mathcal{R}}(k) \tag{5.10}$$

In practice however, the author has found that this method gives a poor approximation to the true Hessian in the Cartesian space, particularly as the number of links in the kinematic chain increases. This implies that there are significant higher order partial derivatives for the transformation from the Cartesian state space to the rotation state space. This approximation has not been used in later experiments due to the poor quality of the approximation.

When calculating the Hessian, the measurement function's computational time can be reduced by noting that not all of the partial derivatives $\frac{\partial^2 \mathbf{H}}{\partial \mathbf{x}_i \partial \mathbf{x}_j}$ change the entire kinematic chain. If state variables $\mathbf{x}_i$ and $\mathbf{x}_j$ both act upon a foot, then only the position of the foot needs to updated. For the joined shortest path method presented in Section 4.4, the measurement function evaluation's average computational time is reduced by 45% when computing the Hessian. In this case the occlusion map was entirely recalculated, and so further computational gains could be made by making the same occlusion assumptions as were used to generate the Hessian's sparsity structure.

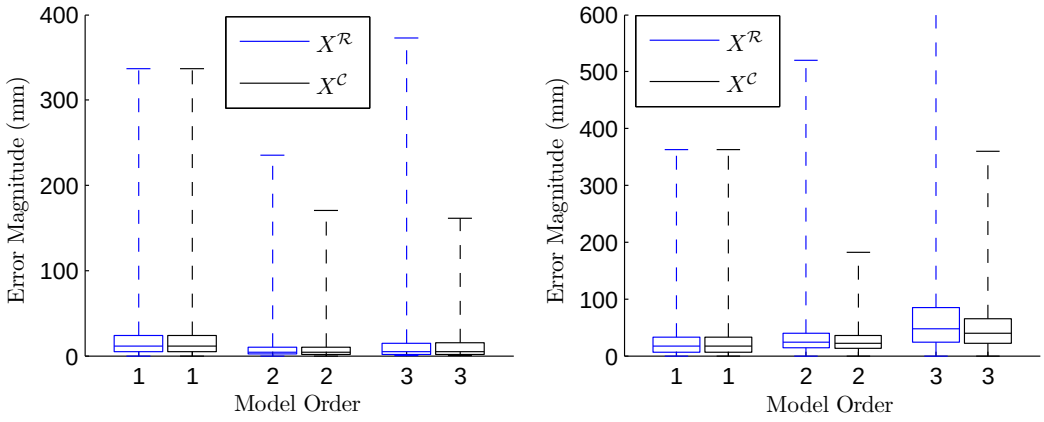
5.4 State Spaces and Dynamic Models

The predictions of some simple dynamic models in the rotation and Cartesian state spaces are compared in this section. It is expected that perturbations (errors) in the estimates of state variables manipulating links near the start of the kinematic chain will propagate down the kinematic chain in the rotation space, whereas these errors will have a reduced impact in the Cartesian space.

To test this hypothesis, predictions were generated for the two image sequences used throughout this thesis, which were shown in Section 1.4. Two types of ground truth data were used for each of these image sequences, denoted *smoothed* and *raw*. The *raw* data set was generated by a user hand labelling a golfer's posture during a swing. The *smoothed* data set was obtained by applying

a low pass filter to the joint angles in the raw data set, and then re-labelling the image sequence to ensure that the model–image correspondence was maintained. This process was iterated until the joint angles were as smooth as possible, while preserving the model–image correspondence. The *raw* data set is then the state estimates returned by an ideal causal tracker[4] during real time tracking, while the *smoothed* data set is the result of non-causal offline processing.

To evaluate the prediction errors, mesh points were (approximately) uniformly spaced on each the links' surface in $\mathbb{R}^3$, with a point every 5cm^2. The prediction error of a mesh point was the Euclidean distance between the mesh point's position in the data set configuration, and its position in the predicted configuration. There are 950 mesh points[5] for each frame for both data sets, giving ≈ 44000 total mesh points for each data set. The Cartesian prediction was geometrically constrained before the prediction error was calculated.



(a) Data set 1 - smooth

(b) Data set 1 - raw

(c) Data set 2 - smooth

(d) Data set 2 - raw

FIGURE 5.4: Dynamic Model Prediction Errors

Predictions were generated using three different dynamic models: constant position (1^{st} order),

[4]The author knows of no visual tracking method that can outperform human interpretation.

[5]There were 933 mesh points for the human body model corresponding to Data set 1, and 976 mesh points for the taller human body model corresponding to Data set 2.

constant velocity (2^{nd} order), and constant acceleration (3^{rd} order). Figure 5.4 shows the mesh error distribution for each of these dynamic models for each data set in both the rotation and Cartesian state spaces. These results are summarized in Table 5.2, with the minimum of the errors in both state spaces shown in bold. The error distributions for the 1^{st} order constant position dynamic model are naturally identical.

	Median Error $X^{\mathcal{R}}$ / $X^{\mathcal{C}}$			Maximum Error $X^{\mathcal{R}}$ / $X^{\mathcal{C}}$		
Data set	Order: 1	2	3	1	2	3
Data set 1 - smooth	11.2	4.4 / **4.1**	4.7 / **4.7**	337	235 / **171**	373 / **162**
Data set 2 - smooth	8.1	2.5 / **2.3**	1.2 / **1.2**	219	962 / **155**	841 / **268**
Data set 1 - raw	17.0	24.1 / **22**	47.2 / **40**	363	520 / **182**	1205 / **360**
Data set 2 - raw	29.7	47.8 / **44**	95.2 / **75**	228	1082 / **213**	1126 / **322**

Table 5.2: Prediction Errors (mm)

From Table 5.2 there was little difference between the *median* prediction errors for the rotation ($\mathcal{R}$) and Cartesian ($\mathcal{C}$) state spaces, however these differences increase as the dynamic model order increases. The *maximum* prediction errors for the Cartesian state space were significantly lower than for the rotation space however. This is believed to be the result of the prediction errors accumulating down the kinematic chain, making the mesh point predication errors for mesh points on the ends of the kinematic chain far greater in the rotation state space. In the Cartesian space, prediction errors do not accumulate in this manner.

These reduced errors in the Cartesian space occur even for the smoothed data, despite the data smoothing having been performed in rotation state space. Furthermore, the primary motion of the golfer in the data sets is a rotation – the rotation of the lead hand about the lead shoulder, as the lead arm is straight ideally. Table 5.2 also indicates that the predictions in the Cartesian space are more robust to the noisy (raw) data. This is of particular importance since there is little temporal smoothing during online tracking. Additionally, Sminchisescu and Triggs [90] advocate adding noise to the previous state estimates when propagating the model during tracking.

From the point of view of choosing a dynamic model to use during tracking, the maximum prediction errors are considered since it is considered more desirably to have many small prediction errors for link positions than one large prediction error for a link. This is because it is more likely that a local optimizer could resolve many small errors than one large error. In the rotation space, the 1^{st} order constant position dynamic model generally gave the smallest maximum mesh errors, and was used by authors such as Deutscher and Reid [33], and Sminchisescu and Triggs [90]. In the Cartesian space however the 2^{nd} order constant velocity dynamic model generally outperformed the other models, and

so has been used in later experiments and tracking.

5.5 State Spaces and Principal Component Analysis

A principal component analysis (PCA) is used to find a compact subspace representation of the data set $\mathbf{X} = \{X_i, i = 1, 2, ..., n\}$, by choosing a set of basis vectors $\mathbf{V}$ to minimize:

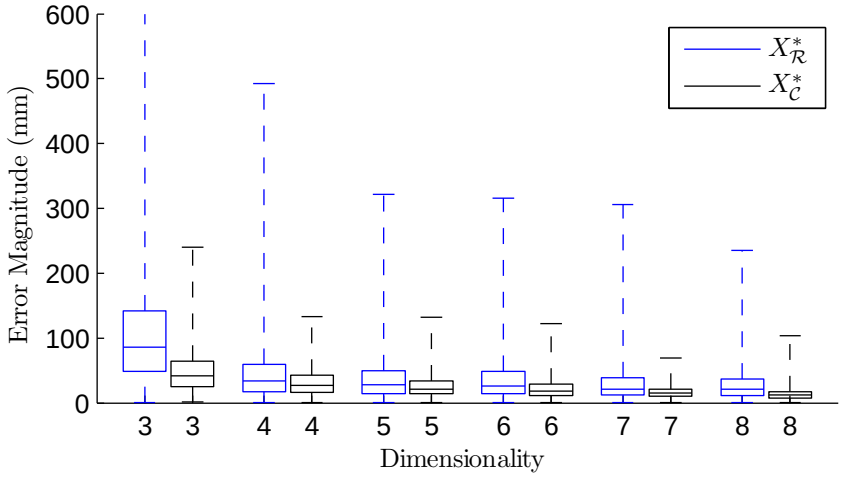
$$\sum_{i=1}^{n} \left((X_i - \mathbf{V}X_i^* - \overline{X})^T \mathbf{W} (X_i - \mathbf{V}X_i^* - \overline{X}) \right)^2 \tag{5.11}$$

with

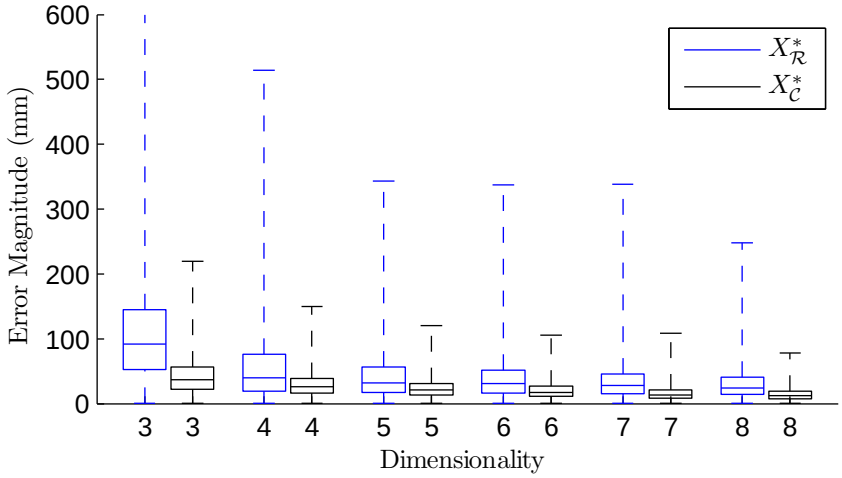
$$\begin{aligned} X_i^* &= \mathbf{V}^*(X_i - \overline{X}) \\ \mathbf{V}^* &= \left(\mathbf{V}^T\mathbf{W}\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{W} \\ \overline{X} &= \frac{1}{n}\sum_{i=1}^{n} X_i \end{aligned}$$

where $\mathbf{X}^*$ is the subspace representation of data set $\mathbf{X}$, and $\mathbf{W}$ is a linear metric function [50]. As discussed in Section 5.1, in a state space comprised of joint angles it is impossible to define a linear metric $\mathbf{W}$, which gives a meaningful measure of the distance two object states are apart. Given a fixed set of basis vectors, there is no guarantee that the calculated subspace point X_i^* is the 'closest' point in the subspace to X_i. This has implications for approaches utilizing principal component analysis, such as that of Urtasun *et al.* [99] discussed earlier in Section 2.5.3, and more recently Zhao and Liu [110].

To illustrate this, a principal component analysis was performed on both smoothed data sets, described in Section 5.4, in both the Rotation ($\mathbf{x}^{\mathcal{R}}$) and Cartesian ($\mathbf{x}^{\mathcal{C}}$) spaces. The smoothed data sets were used because the subspace calculation is performed offline, and so non-causal data may be used. The dimensionality of the subspace was varied from 3–8, where a 4 dimensional space was used by Urtasun *et al.* The metric used to evaluate the distances between $\mathbf{X}$, $\mathbf{X}^*_{\mathcal{R}}$, and $\mathbf{X}^*_{\mathcal{C}}$, is the mesh point displacement in $\mathbb{R}^3$ used in Section 5.4. The Cartesian states were again geometrically constrained before evaluation. The two data sets were combined during calculation of the subspace, in the Cartesian case using the coordinates for a standardised model as discussed in Section 5.2.2. These 'standardised coordinates' account for the height difference between the two golfers. Figure 5.5 shows the distribution of the positional error of all mesh points for all frames of each data set, as the dimensionality of the subspace is varied. The maximum error for the 3 dimensional rotation subspace, not shown in Figure 5.5, is approximately 1000mm for both data sets.



(a) Projection errors, data set 1



(b) Projection errors, data set 2

FIGURE 5.5: PCA subspaces projection errors

Figure 5.5 shows that performing the principal component analysis in the Cartesian space results in 'closer' configurations than when the principal component analysis is performed in the rotation space. Most notable is the reduction in the maximum mesh point error when the PCA is performed in the Cartesian space. This is the result of subspace projection errors not propagating down the kinematic chain in the Cartesian space. The difference between the two spaces reduces as the dimensionality of the subspace increases as expected. As an example, for Data set 1 the median error for the 3 dimensional Cartesian subspace is slightly larger than for the 4 dimensional rotation subspace, however the maximum error of the Cartesian subspace is only approximately half that of the rotation subspace. Also of note is that while the median error monotonically decreases as the dimensionality of the subspace increases in this experiment, the maximum error does not.

5.6 Optimization Comparison for the Different State Spaces

5.6.1 Synthetic Example - Simulated Annealing

The 'topological dominance' phenomenon encountered by Deutscher and Reid [33] is investigated in this section. This topological dominance led to the development of a hierarchical soft partitioning approach for annealed particle filters. It is shown that this 'topological dominance' phenomenon is primarily a function of using a state space comprised of rotation angles.

Deutscher and Reid presented an experiment based on localizing the position of a 4 degree of freedom planar arm, anchored at the origin. This arm and state variables for a rotation ($\mathbf{x}^{\mathcal{R}}$) and a Cartesian ($\mathbf{x}^{\mathcal{C}}$) state space are shown in Figure 5.6. In the experiment presented here, the probability of a configuration was evaluated by generating 16 points uniformly along the 80 pixel long arm, and taking the distance from these points to the closest point on the arm in the true configuration.

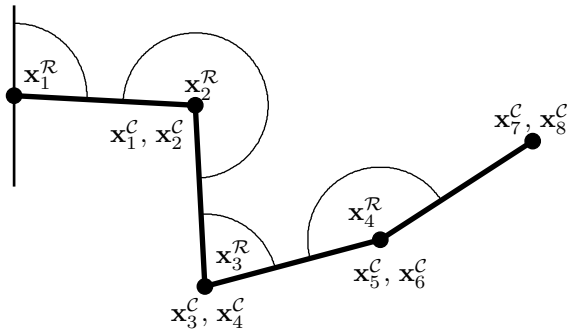


FIGURE 5.6: Planar Arm State Space

Following the experiment presented by Deutscher and Reid, a particle set $\mathbf{X}_0$ was initialized by uniformly sampling each rotational dimension in an interval known to contain the true state angle. A weighted resampling operation was then performed on this original sample set to produce a new particle set, $\mathbf{X}_1^*$. As advocated by Deutscher and Reid, the *particle survival rate* was set to 50% during resampling , so that the most probable 50% of the particles in $\mathbf{X}_0$ were sampled from. The variances for each state dimension of the original and resampled particle sets are shown in Figure 5.7, expressed in both rotation and Cartesian state spaces. The rotation state space variances for $\mathbf{X}_0$ and $\mathbf{X}_1^*$ are approximately the result obtained by Deutscher and Reid.

Deutscher and Reid commented that while the resampling operation helped to localize the position in $\mathbf{x}_1^{\mathcal{R}}$ dimensions, it did little to localize the position in the $\mathbf{x}_2^{\mathcal{R}}$, $\mathbf{x}_3^{\mathcal{R}}$ and $\mathbf{x}_4^{\mathcal{R}}$ dimensions. They argue that $\mathbf{x}_1^{\mathcal{R}}$ dominates the topology of the state space, and that $\mathbf{x}_2^{\mathcal{R}}$, $\mathbf{x}_3^{\mathcal{R}}$ and $\mathbf{x}_4^{\mathcal{R}}$ had little influence on whether a particle was resampled from.

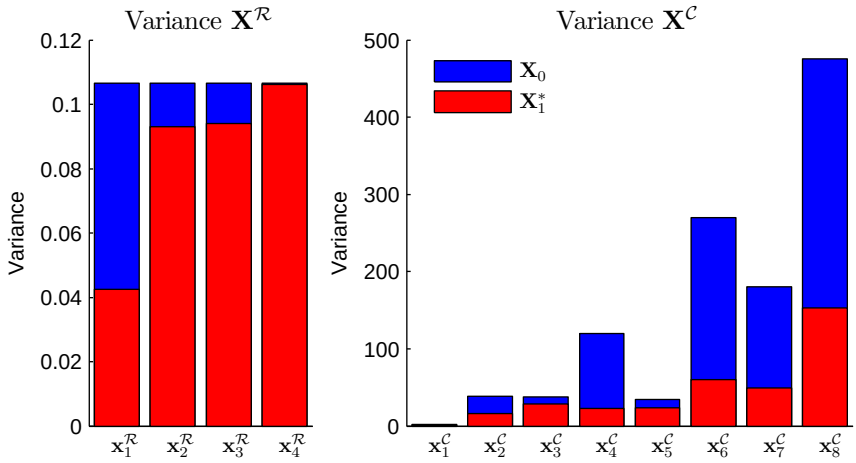


FIGURE 5.7: State variances of the original and resampled particle sets

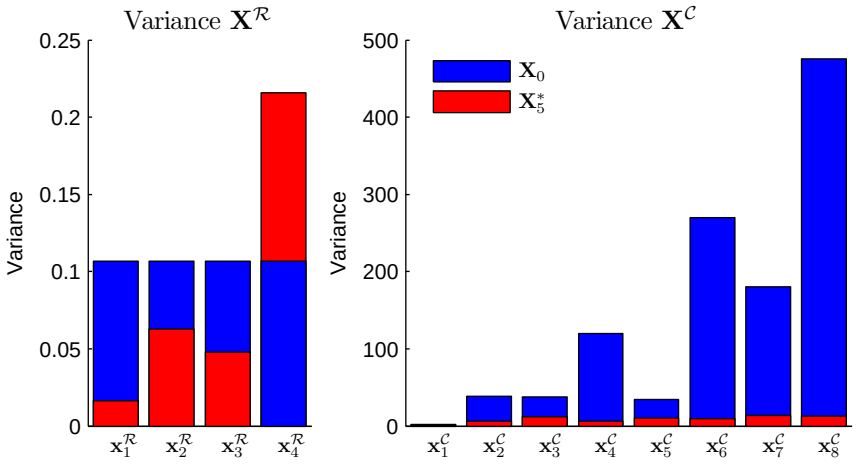
Examining the resampled particle set in the Cartesian state space however shows that the resampling operation helped localize all state variables. It is clearer that the original particle set searched a larger region of the image for the end link ($\mathbf{x}_7^{\mathcal{R}}$, $\mathbf{x}_8^{\mathcal{C}}$) than for the start link ($\mathbf{x}_1^{\mathcal{R}}$, $\mathbf{x}_2^{\mathcal{C}}$), and that the end link was localized to some extent during resampling. While this is simply interpretive as different expressions for the same particle set are shown, quantitative differences arise when noise is added to the resampled set. Deutscher and Reid state that the noise added to each particle in $\mathbf{X}_i^*$ should be dynamically adjusted, and drawn from a Gaussian distribution with covariance proportional to the covariance of $\mathbf{X}_i^*$.

To test the difference between using dynamically adjusted noise in the rotation and Cartesian state spaces, the annealing process started above was continued for 10 annealing layers. Noise added to the particle set $\mathbf{X}_i^{\mathcal{C}*}$ in Cartesian space was drawn from a Gaussian distribution with covariance $\Sigma_i^{\mathcal{C}} = \text{cov}\left(\mathbf{X}_i^{\mathcal{C}*}\right)$. In the Cartesian state space, this noisy particle set $\mathbf{X}_i^{\mathcal{C}}$ was geometrically constrained immediately, *i.e.* constraints were enforced before probabilistic interpretation.

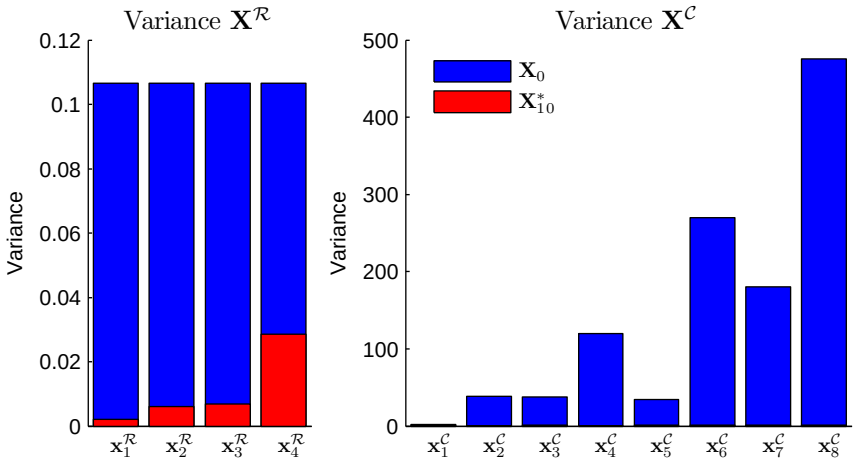
Anecdotal evidence indicates that the constraint process can either add or reduce noise, however a study into whether it is biased towards adding or removing noise has not been performed. Because particles in $\mathbf{X}_0$ are near the true object location, in this experiment the annealing process converges faster when less noise is added to the particle set. As such, to avoid creating a potential bias towards the Cartesian state space results, the ratio, r, of the actual added noise compared to the intended added noise was calculated as:

$$r = \frac{\text{trace}\left(\text{cov}\left(\mathbf{X}_i^{\mathcal{C}} - \mathbf{X}_i^{\mathcal{C}*}\right)\right)}{\text{trace}\left(\Sigma_i^{\mathcal{C}}\right)} \tag{5.12}$$

This variance reduction measure is commonly used when comparing data set variation when performing principal component analysis. This ratio was then used to determine the amount of noise added to the particle set in the rotation space $\mathbf{X}_i^{\mathcal{R}*}$, such that $\Sigma_i^{\mathcal{R}} = \min(r, 1) \times \text{cov}\left(\mathbf{X}_i^{\mathcal{R}*}\right)$.



(a) Annealing Layer 5



(b) Annealing Layer 10

FIGURE 5.8: State variances at different annealing layers

Figure 5.8 shows the variance in each state dimension for the noiseless particle sets drawn at annealing layers 5 and 10. Of note is that is that in annealing layer 5 the variance in state dimensions $\mathbf{x}_4^{\mathcal{R}}$ is larger than in the original sample set. As mentioned above, in the rotation state space the degree of localization of the final link is not truly represented by the variation in $\mathbf{x}_4^{\mathcal{R}}$, and as such the noise in this dimension does not reduce as the localization improves. By annealing layer 10 however, the reduced variation in $\mathbf{x}_1^{\mathcal{R}}$, $\mathbf{x}_2^{\mathcal{R}}$, and $\mathbf{x}_3^{\mathcal{R}}$ means the variation in $\mathbf{x}_4^{\mathcal{R}}$ better represents the localization of the end link, and as such the variance in $\mathbf{x}_4^{\mathcal{R}}$ has started to reduce. This process means the annealing process in rotation space does not converge quite as quickly as in Cartesian space, as shown by Figure 5.9. Perhaps the most interesting aspect of Figure 5.9 is the probability of the least likely sample in each set, shown by the lower end of each whisker. This probability increases faster in the Cartesian space, suggesting that annealing in the Cartesian space does a better job of concentrating samples in 'interesting' areas.

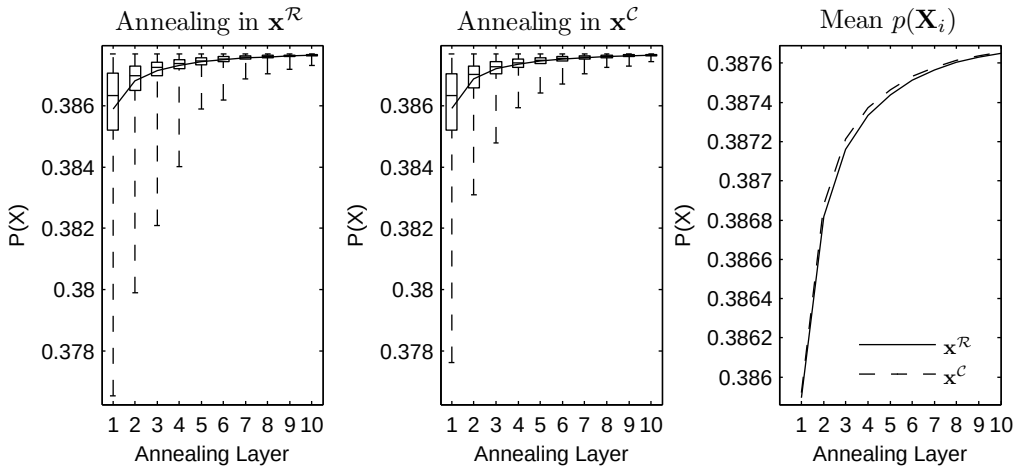


FIGURE 5.9: Annealing performance vs annealing layer

Anecdotal evidence also suggests that the Cartesian state space annealing process is more robust to increasing the amount of noise used for resampling, $\Sigma_i^{\mathcal{C}} = s \times \text{cov}\left(\mathbf{X}_i^{\mathcal{C}*}\right)$ and $\Sigma_i^{\mathcal{R}} = s \times \min(r, 1) \times \text{cov}\left(\mathbf{X}_i^{\mathcal{R}*}\right)$, with $s > 1$. It was found that a value of s could be chosen which caused the annealing process in the rotation state space to diverge, but did not cause the Cartesian annealing process to diverge.

5.6.2 Synthetic Example - Newton Optimizer

The performance of a damped Newton trust region local optimizer operating in a rotation ($\mathbf{x}^{\mathcal{R}}$) and a Cartesian ($\mathbf{x}^{\mathcal{C}}$) state space is examined in this section. The experiment was performed using the articulated planar arm example discussed in Section 5.6.1. Rather than maximizing the probability however, the optimization was formulated as a minimization problem, where the objective function was the sum of the squared distances from the estimated state's sampled points to the closest point on the arm in its true position. This was done to give the objective function a more quadratic form, which improves the convergence rate of a Newton based optimizer.

As discussed in Section 5.2, it is expected that the reduced number of significant higher order derivatives in the Cartesian space should yield improved performance of the optimizer. Of note, if only one sample point were used at the end of each link, and a known correspondence was assumed between this point and its equivalent point in the true state, then the objective function in geometrically consistent regions of the Cartesian space reduces to $\left\| X^{\mathcal{C}}_{\text{estimate}} - X^{\mathcal{C}}_{\text{true}} \right\|_2$. Such a situation could arise if each of the arm's hinges were a different colour.

In this experiment, the Jacobian and Hessian were calculated numerically in both state spaces, at an equivalent distance for central differences. The same Hessian dampening scheme, based on

manipulation of the Hessian's eigenvalues, was used in both spaces. In both cases the trust region was initialized larger than the maximum plausible step size. As in Section 5.6.1, the objective function was never evaluated in a geometrically inconsistent Cartesian state. This was achieve by projecting the points to be calculated for the central differencing back onto the constraint surface.

The Cartesian space the implausible direction dampening, discussed in Section 5.3.1, was applied after the normal Hessian dampening. This was done because the implausible direction dampening can increase the maximum eigenvalue of the Hessian, which then changes the minimum eigenvalue allowable during dampening. This could be seen as reducing the comparability of the two optimizations. In normal circumstances the implausible direction dampening should occur before the normal Hessian dampening, as it can effect the numerical stability of the Hessian.

In both experiments, the same seed configuration was used for both optimizations. The seed was generated by adding a variable Gaussian noise (in the rotation state space) to the true configuration, with the same noise level σ used for all state dimensions. Where extra links are added to the arm, the true angle of these links was randomly drawn from a uniform distribution ranging from 0 to 2π. This seed configuration was then converted to a Cartesian state. If a seed configuration contained intersecting links was drawn, the seed configuration was discarded and a new seed configuration drawn.

Experiment 1 - Convergence Rate

In this optimization experiment, the ability of the optimizer to make small adjustments to the state estimate was tested by altering the objective function's convergence threshold. It is expected that the more linear relationship between the measured points' locations and the Cartesian state vector should result in improved performance in the Cartesian space. The synthetic nature of this experiment means that a state with zero cost is guaranteed to exist. The success of the optimizer was judged by measuring the number of iterations required to reach a given convergence tolerance.

Figure 5.10 shows the performance of the optimizer for convergence error values of 10^{-3}, 10^{-6}, and 10^{-9}, while the seed noise's standard deviation was varied from 10° to 50°, in 10° increments. It can be seen that the performance in both spaces is reasonable independent of the noise used to generate the seed state, suggesting that a large initial update step is used in both state spaces. In the Cartesian state space there is little difference in the number of iterations required to reach the different convergence error values, while in the rotation state space the required number of iterations grows quickly as the convergence error value is decreased. The number of iterations required to achieve a convergence error value of 10^{-9} in the rotation space is not shown as more than 100 iterations were

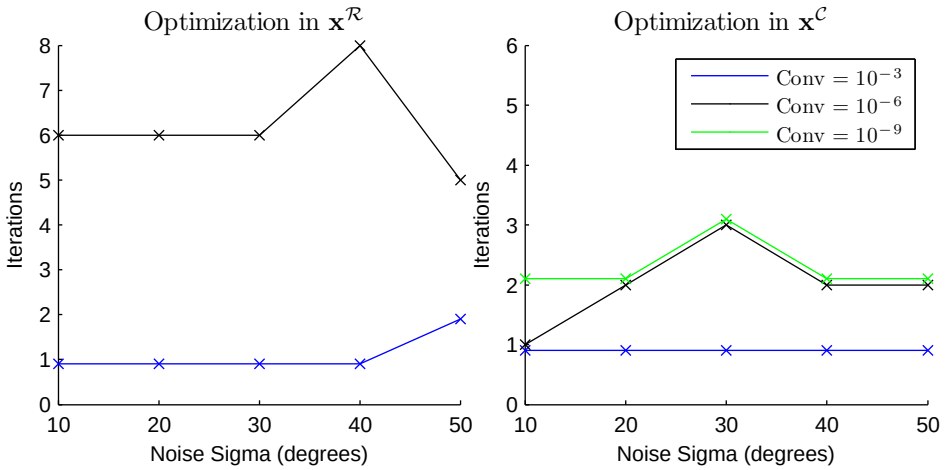


FIGURE 5.10: Optimization iterations and the error tolerance

required for all noise levels. This experiment shows that the convergence rate decreases much faster in the rotation state space.

Experiment 2 - Number of Links

This optimization experiment examines the optimizer's performance as the number of links L in the arm increases. In the Cartesian space, the more linear relationship between the measured points' locations and the Cartesian state vector is preserved as extra links are added to the arm, whereas in a rotation space this relationship becomes more complex as links are added. As such it is expected that the optimizer's performance will decrease much faster in the rotation state space. The success of the optimizer was judged by measuring the number of iterations required to reach a convergence value of 10^{-6}.

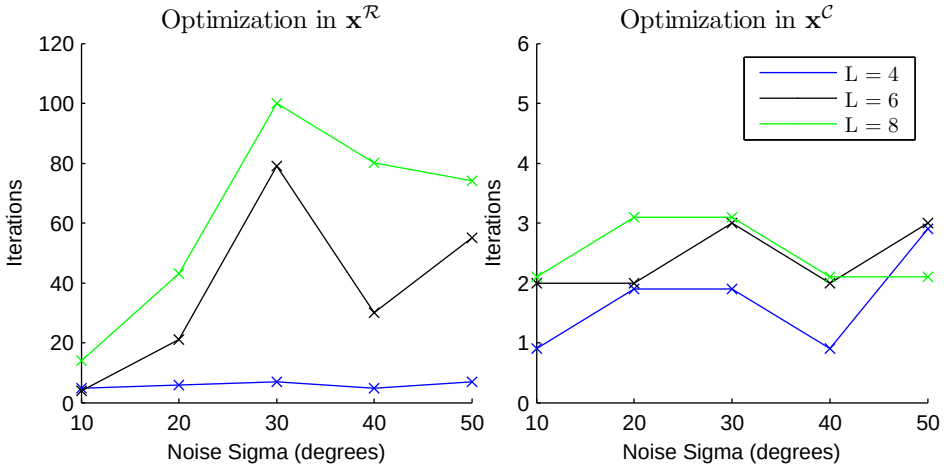


FIGURE 5.11: Optimization iterations and the number of links (L)

Figure 5.11 shows the results of the optimization in the two spaces using 4, 6, and 8 links. From

Figure 5.11 it can be see that the optimization performance in the Cartesian space is somewhat independent of the number of links, whereas in rotation space the number of iterations required for convergence increases as the number of links increases.

As mentioned above, no seed or true states were used in which link intersections occurred. The author notes that when link intersections are present, the optimization in Cartesian space often fails to converge to a 'reasonably close' solution. The reason for this has not been investigated, as body part interpenetration avoidance constraints [90] may be used in tracking problems if required.

5.6.3 Real Example - Newton Optimizer

In this section an experiment is presented to determine if the advantage of performing local optimizations in a Cartesian based state space, demonstrated in Section 5.6.2, is preserved when using a full body articulated model and real images for measurement.

The seed position for all optimizations in each frame was the prediction of a constant velocity dynamic model using the *raw* ground truth state data in the Cartesian space, as discussed in Section 5.4. The raw data set was used as it represents state estimates from an ideal causal tracker. The first frame of the sequence was shown in Figure 4.11, where the image sequence is again chosen because of the combination of both low and high contrast edges.

The same objective function was used in all cases, $-log(p(\mathbf{z}_t|\mathbf{x}_t))$, where $p(\mathbf{z}_t|\mathbf{x}_t)$ was calculated using: the edge joined cost path score from Section 4.4, the region consistency score from Section 3.5.2, and the occlusion likelihood score from Section 3.1.3. In each case convergence was indicated when a step was taken which reduced the objective function by less than 10^{-6}, which is an order of magnitude lower than is used in the tracking results presented in Chapter 7. No anatomical joint limit constraints were used in this experiment, because as discussed later in Section 5.6.4, a different form of these constraints is used in the Cartesian state space.

As in Section 5.6.2, in the Cartesian space the objective function was never evaluated at a geometrically implausible state. Implausible update states were always projected back onto the constraint surface using the method discussed in Section 5.2.1. This meant that the gradient and Hessian were always calculated at a geometrically consistent state, reducing the number of significant higher order derivatives of the measurement function as discussed in Section 5.2.

Due to the high computational cost of a Hessian evaluation for the full articulated model, modifications aimed at reducing the number of times the Hessian must be calculated numerically were made to Algorithm 2. The first modification was that a quadratic 1D line search was performed in the update direction $\frac{\delta X_i}{\|\delta X_i\|}$ at iteration i, initialized at the trust region governed distance along this line,

1) Starting with an initial (seed) estimate X_0 and trust region R_H, set *converged* = *false*,
$X = X_0$, and $c_{best} = -\log(p(Z|X_0))$.
2) Loop until convergence using the true Hessian:
while $\neg$ *converged*
2a) Calculate the gradient $\mathbf{J}(X)$ and the exact Hessian $\mathbf{H}(X)$.
2b) Dampen the exact Hessian $\mathbf{H}(X)$.
2c) Set the quasi-newton convergence flag $converged_{quasi} = false$, the exact Hessian flag
$T_H = true$, and the approximated trust region, $R_A = R_H$.
3) Loop until convergence using the approximated Hessian:
while $\neg converged_{quasi}$
3a) Calculate the distance to move, $\delta X = -\mathbf{H}(X)^{-1}\mathbf{J}(X)$.
3b) Restrict the step size δX to the trust region R_A.
3c) Calculate the new estimate, $X^* = X + \delta X$.
3d) Update the approximate trust region R_A using $c^* = -\log(p(Z|X^*))$.
if $T_H = true$ **then**
Set $R_H = R_A$.
4) Perform a Newton based 1D line search in the direction of δX seeded at X^*, returning the
updated estimate X^* with cost c^*.
if $(c_{best} - c^*) \geq$ *Function Tolerance* **then**
5a) Set $X = X^*$, and $c_{best} = c^*$.
5b) Calculate $\mathbf{J}(X)$, and use the BFGS update to approximate the Hessian $\mathbf{H}(X)$.
5c) Set the exact Hessian flag $T_H = false$.
else
5d) Set $converged_{quasi} = true$ (alternatively reduce R_A).
6) Set $converged = \neg T_H$.
7) **Return** X and c_{best}.

Algorithm 9: Damped Newton Optimization with Line Search

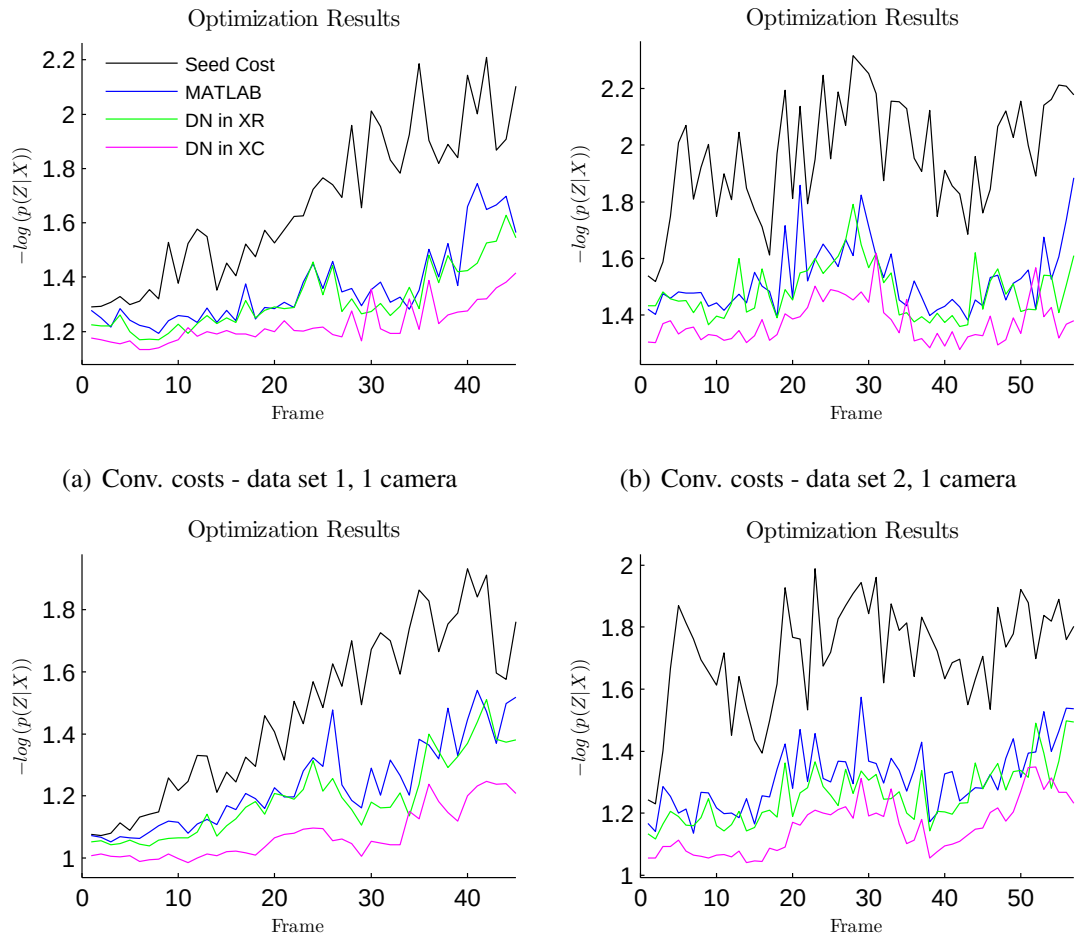
$\min(\|\delta X_i\|, R_i)$, where R_i is the trust region.

The second modification was that the Hessian was updated using the BFGS update discussed in Section 2.3.3. When convergence was achieved using the approximated Hessian, the exact (numerically calculated) Hessian was recomputed in an attempt to break this convergence. This process was repeated until convergence was achieved using the exact Hessian. Separate trust regions were maintained during this process, the first the trust region for update steps calculated from the exact Hessian, R_H, and the second the trust region for update steps calculated from the approximated Hessian, R_A. Each time the exact Hessian was recalculated, R_A was reset such that $R_A = R_H$. This modified Newton optimizer is given in Algorithm 9.

To validate this modified optimizer, optimizations in the rotation space were also performed using

the MATLAB® optimization function *fminunc()*, with both optimizers using the same function for calculating the Jacobian and Hessian via central differences. The step distance used for the central differencing was set large enough to cause a measurable difference in the object's appearance in the image. This was designed to overcome problems such as the non-differentiability of the objective function in some places, discussed in Section 2.3.4.

Results



(a) Conv. costs - data set 1, 1 camera

(b) Conv. costs - data set 2, 1 camera

(c) Conv. costs - data set 1, 2 cameras

(d) Conv. costs - data set 2, 2 cameras

FIGURE 5.12: Optimization Convergence Cost Comparison

Figure 5.12 shows the optimization results for the damped Newton (DN) approaches performed in the rotation $\mathbf{x}^{\mathcal{R}}$ and Cartesian $\mathbf{x}^{\mathcal{C}}$ state spaces, as well as the MATLAB® unconstrained optimization function *fminunc()*. These results are summarized in Table 5.3. In the rotation space, the hybrid optimization scheme shown in Algorithm 9 had comparable performance in terms of convergence costs to the MATLAB® implementation, however required significantly fewer objective function evaluations as intended. The convergence performances were more closely matched in an earlier implementation where rotation angles were recorded in degrees in the state vector, and so the Hessian re-scaling

discussed in Section 5.3.1 was not required.

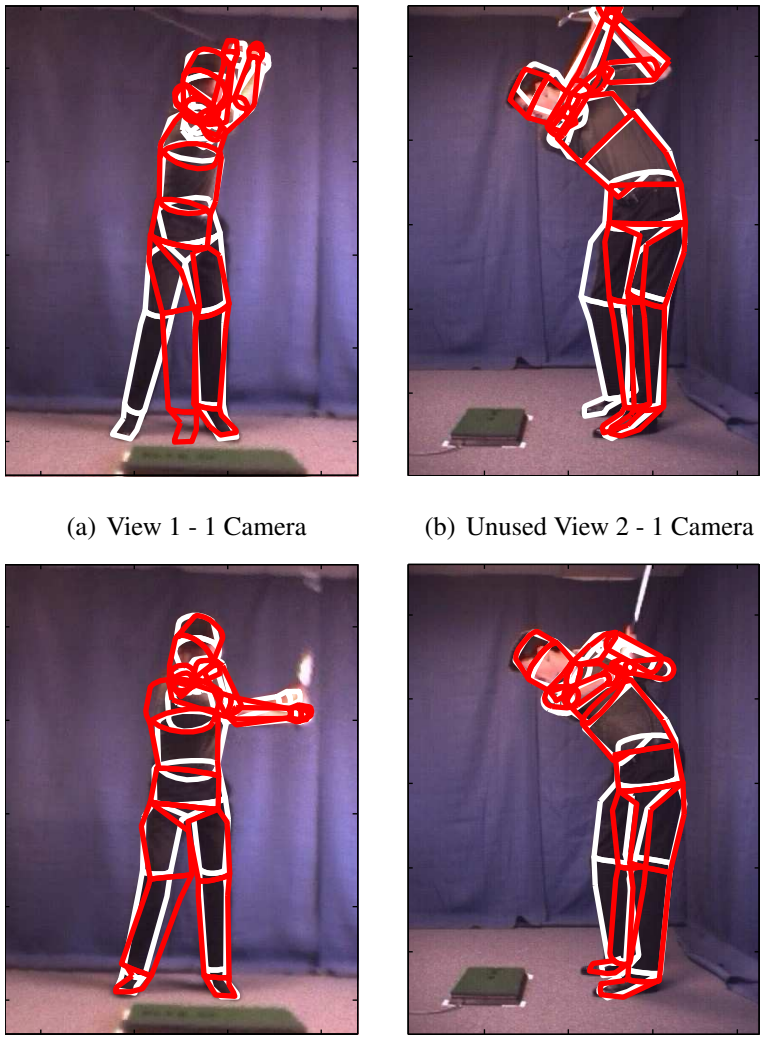
Type	Mean Cost	Std	Total Cost	Mean func. evaluations	Total func. evaluations	% Best
1 Camera						
MATLAB in $\mathbf{x}^{\mathcal{R}}$	1.4533	0.1540	148.2334	42606	4346×10^3	2%
DN in $\mathbf{x}^{\mathcal{R}}$	1.4102	0.1323	143.8413	12114	1236×10^3	5%
DN in $\mathbf{x}^{\mathcal{C}}$	1.3089	0.1042	133.5125	20736	2115×10^3	93%
2 Cameras						
MATLAB in $\mathbf{x}^{\mathcal{R}}$	1.2815	0.1285	130.7087	41911	4275×10^3	0%
DN in $\mathbf{x}^{\mathcal{R}}$	1.2286	0.1123	125.3194	12602	1285×10^3	4%
DN in $\mathbf{x}^{\mathcal{C}}$	1.1190	0.0924	114.1360	19056	1944×10^3	96%

Table 5.3: Optimization Comparison

Following the synthetic optimization results presented in Section 5.6.2, Figure 5.12 shows that in the two camera case the optimization in the Cartesian state space outperformed the rotation state space optimization in all but 4 frames. This indicates that the optimization in the rotation state space was not converging to a true local mode, but rather to a state where the chosen search direction was not a descent direction. In the monocular case the optimization in Cartesian space outperformed the rotation space optimizations 93% of the time. This higher variability is expected in the monocular case, as Section 4.8.5 showed that more local modes exist in the monocular case, increasing the chances that the different optimization methods will converge to different local modes. A Wilcoxon signed rank test [109] applied to the optimizations in the Cartesian and rotation state space reveals that there is less than a $10^{-10}\%$ chance that the differences between the two methods' convergence costs has zero median, in both the monocular and in the two camera cases.

Figures 5.13(a) and 5.13(b) show the optimized states from the monocular optimization for the rotation (red) and Cartesian (white) spaces in frame 28 of data set 2, where this frame was chosen as it has the largest difference in convergence values in the monocular case. The view shown in Figure 5.13(b) was not used by the measurement function. Figures 5.13(c) and 5.13(d) show the worst comparative performance of the rotation space optimizer in the two camera case, corresponding to frame 19 of data set 2. Figure 5.13 shows that the performance difference between the optimizers convergence cost is significant in terms of the golfer's appearance in the image. In both cases shown, the Cartesian optimizer has converged to approximately the correct result.

The synthetic experiment presented in Section 5.6.2 also showed that optimizing in the Cartesian state space yielded an increase in the convergence speed. Figure 5.14 shows the progress of the Cartesian space optimizer, linearly interpolated between accepted state updates, at the same number



(a) View 1 - 1 Camera (b) Unused View 2 - 1 Camera

(c) View 1 - 2 Cameras (d) View 2 - 2 Cameras

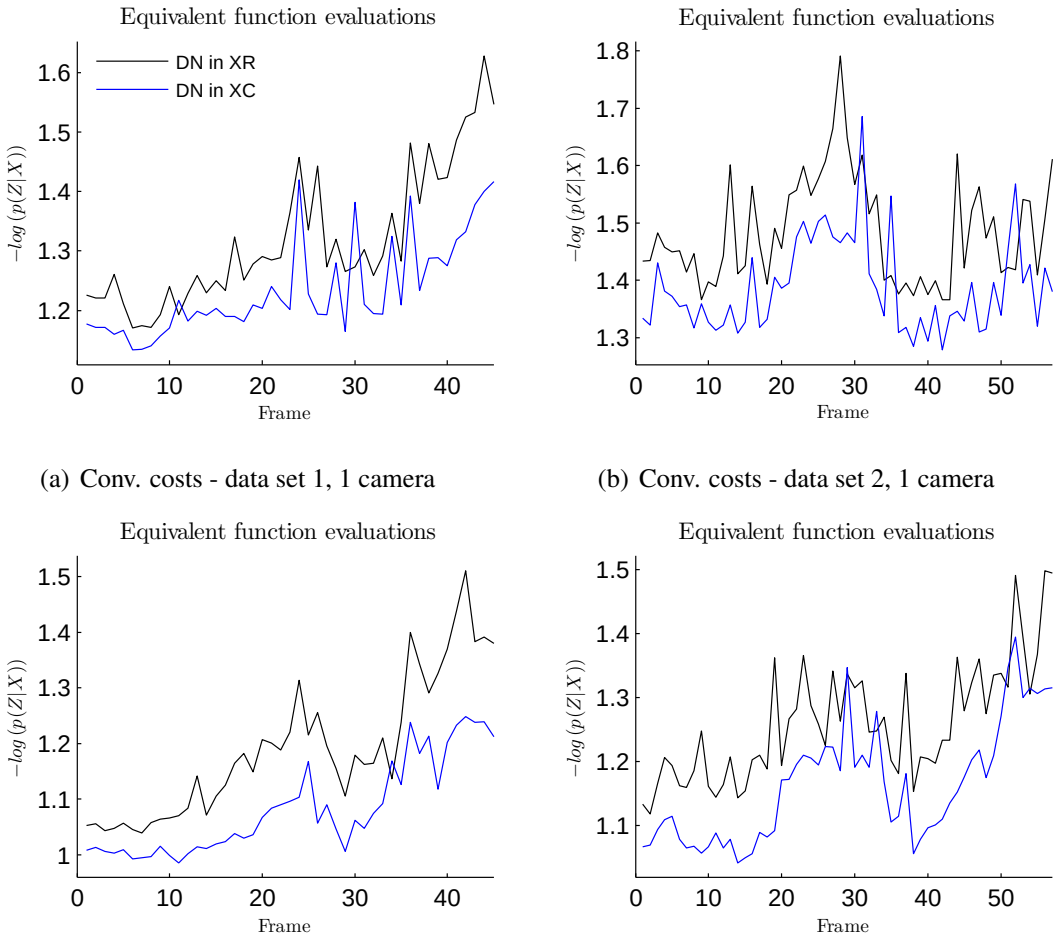
FIGURE 5.13: Converged states for the rotation (red) and Cartesian (white) optimizations

of objective function evaluations as was used by rotation space optimizer. This shows that Cartesian space optimization generally has a greater convergence speed than the rotation optimizer. This is despite each exact Hessian calculation in the Cartesian state space requiring $\frac{1916}{3212} = 167\%$ more function evaluations than in the rotation state space.

5.6.4 Joint Limits and the Cartesian state space

The results presented in Section 5.6.3 were obtained using an *unconstrained* local optimization[6]. For practical tracking problems, additional constraints such as joint angle limits and body part interpenetration avoidance may be desirable. When working in the rotation space, joint angle limits can be formulated as linear inequalities, or "box constraints" [90], which can be implemented effectively using methods such as described by Fletcher [38]. In the Cartesian space however, the joint angle

[6]With the exception of the geometric (link length) constraints when working in the Cartesian space



(a) Conv. costs - data set 1, 1 camera

(b) Conv. costs - data set 2, 1 camera

(c) Conv. costs - data set 1, 2 cameras

(d) Conv. costs - data set 2, 2 cameras

FIGURE 5.14: Optimization costs at equivalent objective function evaluations

limits cannot be expressed as linear inequalities.

If joint angle limits are desired in the Cartesian space, they may be implemented as a "barrier function" [38], where the objective function is updated to be the sum of the measurement plus a strict penalty function. This is the same method by which body part interpenetration avoidance can be enforced, since body part interpenetration avoidance will be a nonlinear constraint in all of the state spaces examined in this thesis. In the tracking results presented later in Chapter 7, an alternative formulation for joint limits constraints was used for the clustered rotation groups. This formulation is based upon projecting the rotated link onto the axes of its un-rotated (input) coordinate frame. This can be thought of as restricting the x and y angles between the link and its 'natural' direction, *i.e.* the link's direction when its rotation angles are zero. These projections are non-linear functions of the state variables however. The orientation of the input coordinate frame as a function of the state variables is highly non-linear. Given a constant input coordinate frame however, it is trivial to map a link whose joint limits are violated to a position which satisfies these limits. This makes it trivial to constrain the link to a plausible direction during the geometric constraint process.

5.6.5 Discussion

In this Chapter, formulating the tracking problem in a state space comprised of the Cartesian coordinates of links in the kinematic chain was studied. Using this state space was shown to improve: principal component analyses, predictions via dynamic models, and optimization performance, when compared to a state space comprised of link joint angles. To the author's knowledge, no visual tracker using a full kinematic chain to model the human body has used a state space based on these Cartesian coordinates, however visual trackers using representations of kinematic chains have used a similar state space formulation, e.g. Morris and Rehg [72], and Sigal, Bhatia, Roth, Black, and Isard [83].

While this Cartesian based state space has a higher dimensionality than the rotation based state space, geometric (link length) constraints means the two state spaces effectively have the same content, negating the so called "curse of dimensionality". A computationally efficient method to project an implausible Cartesian state onto the constraint surface is given, as well as an implausible direction dampening scheme to force optimizer updates and sampled particles, to lie in plausible directions.

An unexplored potential advantage of the Cartesian based state space is related to the approximation of the Hessian during Newton like optimizations. Section 4.9 detailed an approximation to the Hessian for the proposed JCP edge measurement function. It was discussed that the function to calculate the position of the ith sampled point on a projected edge as a function of the model state, $\mathbf{r}_i(\mathbf{x})\colon \mathbb{R}^d \rightarrow \mathbb{R}^2$, is highly non-linear and so its derivatives must be calculated numerically. Formulated in the Cartesian space however, $\mathbf{r}_i(\mathbf{x}^{\mathcal{C}})$, is much closer to a linear function, which is essentially the reason for the improved optimizer performance in the Cartesian space. For a geometrically consistent state, the non-linearities of $\mathbf{r}_i(\mathbf{x}^{\mathcal{C}})$ are due to the non-linear camera perspective projection, and any about axis asymmetry of link's surfaces. Approximating $\mathbf{r}_i(\mathbf{x}^{\mathcal{C}})$ with a linear function would mean the Hessian could be entirely analytically approximated, a potential source of a computational saving.

An unexplored potential disadvantage of the Cartesian based state space is related to link intersections. In the experiment synthetic local optimization experiment presented in Section 5.6.2, it was stated that the seed positions were chosen to be states with no intersecting links. When the seed state contained intersecting links, the rotation space optimization often outperformed the Cartesian optimization. The reason for this has not been investigated. The ability of the Cartesian optimizer to 'follow' body part interpenetration penalty functions requires investigation. In the tracking results presented later in Chapter 7, it was found that the edge measurement function proposed in Chapter 4 sufficiently penalized body part interpenetration, and so a specific penalty function was not required. Any states generated by the dynamic model with body part interpenetration were either not selected for optimization, or the optimizer resolved the interpenetration using purely visual cues.

6

Proposed Optimization Scheme

A local optimization procedure utilizing camera geometry is proposed and evaluated in this chapter. This optimization scheme attempts to decompose the optimization into several subproblems, where each subproblem can be efficiently solved using a novel local optimization algorithm that uses camera geometry to choose a series of one dimensional search directions.

6.1 Motivation

Newton based local optimization techniques calculated a search direction using the gradient and curvature of the objective function. The success of this approach is largely governed by the size of the region of the objective function that can be well approximated by a quadratic function, the *trust region*. Chapter 4 discusses the difficulty of designing measurement (objective) functions for visual tracking problems, due largely to the image's high noise levels. There is then an expectation that the trust region of a Newton like optimizer may be 'small' in much of the state space. Non-stationary points may exist for which no point in the calculated search direction presents an improvement, due to 1^{st} order discontinuities expected in the measurement function as discussed in Section Section 2.3.4. Section 2.2 discussed the importance of using information specific to the optimization problem when

creating a good optimizer. The optimization method proposed here continues in this line of thought.

The covariance scale sampling algorithm presented by Sminchisescu and Triggs [90] uses the idea that some directions in the state space are inherently more 'interesting' than other directions, and so samples (state hypotheses) should be concentrated along these directions. One cause of such ambiguous directions is the monocular depth ambiguity. Sminchisescu and Triggs infer these uncertain directions from the measurement function's curvature. As mentioned above, this curvature may be unreliable due to a number of sources including the image's high noise levels. The ambiguous depth directions can be more efficiently calculated using the relative orientation of the object and the camera. That 'interesting' directions can be determined directly from the relative orientation of the object and camera is the motivation for the local optimization strategy presented in this paper.

In Chapter 5, an experiment was performed to compare the performance of a damped Newton based optimizer working in both rotation and Cartesian based state spaces. It was shown that the optimization in the Cartesian space outperforms the rotation space optimization, in terms of both convergence values and descent rates. This however came at the cost of using a large number of function evaluations during each optimization. As commented is Section 5.3.2, each iteration of the optimizer in the Cartesian Space required a minimum of 3212 function evaluations, with the average function evaluation taking ≈ 7.75ms to compute in the monocular case. Each iteration then takes 25s to compute, a far greater computational cost than is desirable. It is then desirable to create an optimization algorithm with similar performance to the damped Newton optimization in the Cartesian space, but at a greatly reduced computational cost.

6.2 Problem Decomposition

A natural starting point to reducing the computational complexity of the optimizer is to decompose the problem into several subproblems, *i.e.* form subsets of the state variables, and then to recursively optimize each of these sets. Furthermore, these subproblems are formed such that search directions can be chosen using camera geometry, rather than by calculating them using the sometimes unreliable gradient and curvature information.

The problem decomposition begins by examining which state variables act directly upon which links, where acting directly is taken to mean the first link in the kinematic chain whose position (or surface orientation in the case of the ends of the kinematic chain) is dependent upon the state variable. The method of choosing search directions detailed in Section 6.3 is based on moving a link's end in a desired direction, and as such is only suitable for links with two state variables acting upon them. As such, links with more than two state variables acting upon them are optimized using the standard

damped Newton approach. In the model formulation used throughout this thesis, which was shown in Figure 3.1, these are:

- The hip - with three translational and three rotational degrees of freedom,
- Each foot - with three rotational degrees of freedom,
- The head - with three rotational degrees of freedom,
- The upper back - with the *z* rotation from the middle back and its own *x* and *y* rotations,
- The lower back - with the *z* rotation from the base of the spine and its own *x* and *y* rotations.

The remaining state variables are grouped according to their assignment to: each lower leg, each upper leg, each lower arm, each upper arm, each shoulder, and the neck. As such there are 5 subproblems to be solved using a damped Newton optimization, and 11 subproblems to be solved using the camera geometry based search direction scheme presented in Section 6.3. There are then 16 *rotations groups* $\mathcal{G}_i, i \in \{1, 2, ..., 16\}$, 11 of which have two variables manipulating a single link, and these groups are denoted *rotation clusters*, $\mathcal{G}_i^*, i \in \{1, 2, ..., 11\}$, with $\mathcal{G}^* \subset \mathcal{G}$. The term *rotation cluster* is used to match the description of the clustering scheme given in Section 5.2.

For the non-clustered rotation groups $\mathcal{G} \ni \mathcal{G}^*$, a damped Newton optimization is performed on the Cartesian state space dimensions corresponding to the rotational space variables given in the above list. This is because, as shown in Section 5.6.3, better convergence performance was achieved when working in the Cartesian state space. This is not important for the rotation groups corresponding to the head and feet however, as they correspond to ends of branches of the kinematic chain.

This problem decomposition is similar to the parallel filter banks approach presented by Sigal *et al.* [83], which was discussed in Section 2.4.9. In this implementation however, the measurement function is calculated using the entire object configuration for each subproblem. While this means that the subproblems can not be solved simultaneously, it also means that measurements can be performed on the 'joins' between links, using the joined cost path approach presented in Section 4.4. The measurements performed on the joins between links are analogous to the conditional probability between links used by Sigal *et al*, however Sigal *et al.* learn these distributions from training data and used them to enforce (loose) geometric consistency. In Section 4.8 it was shown that taking measurements in the join regions reduced the modality of the observational likelihood.

6.2.1 Subproblem Hierarchy

As discussed in the preceding section, the various subproblems in this approach can not be solved simultaneously. This then means that an order must be created for solving each of these problems.

A valid approach would be to balance the progress of the optimizer as a whole across each of the subproblems, by performing only a single iteration of each subproblem's optimizer before moving to the next subproblem, and then repeating this process until all of the subproblems have converged. This approach is not used however as it was judged that it would be more computationally efficient to solve each subproblem fully before moving to the next sub-problem.

This leaves the problem of how to choose the order in which to perform the optimization on each of the rotation groups. Because each subproblem is optimized to convergence, there is a high likelihood that the optimization as a whole will be quite sensitive to the chosen order. There is a trade off when selecting this order. If the rotation groups are ordered such that the links closer to the start of the kinematic chain are optimized first, and working *outwards* to the ends of the kinematic chain, it is possible that state updates will be accepted not because they improve the 'fit' of the link associated with the rotation group, but rather because the model adjustment process, described later in Section 6.4, inadvertently improves the 'fit' of links further down the kinematic chain. This will reduce the efficiency of the optimization process, and can potentially lead the optimization process as a whole into a small local mode.

If the rotation groups are ordered such that the ends of the kinematic chains are optimized first, and working *inwards* to the start of the kinematic chain, it is more likely that the local optimizers will fall into a spurious local mode. The reason for this has two parts. Firstly, if link i is well localized, there will be fewer modes in the marginal observational density for the variables which act on a link j, when link j is a neighbour of the well localized link i. This is because the minimum cost path (from Section 4.4) around link j will be biased towards following the true edge as it will start and finish near the true edges of link i. Secondly, there is an expectation that there will be smaller errors in the temporal predictions of links near the start of the kinematic chain. This is because the base link in the kinematic chain for the model formulation used throughout this thesis is the hip, and in usual human motions the hip moves more predictably than the hands, feet, or head. It is worth noting that in general, biodynamic efficiency is increased when the center of mass (approximately the hip) moves in a more linear fashion.

The subproblem ordering scheme used in the later experiments attempts to get the best of both ordering schemes. This is achieved by using the *outwards* ordering scheme for the first iteration through all of the rotation groups, and then using the *inwards* ordering scheme for all subsequent iterations. This is done so the first iteration moves the end links into the 'correct' local mode, with efficiency then not comprised for the subsequent iterations. As mentioned above however, when using the outwards ordering scheme the optimization process as a whole can be led into an uninteresting local mode. To counteract this, in the first iteration a potential state update is only accepted if it improves the 'fit'

of the link associated with the rotation group, irrespective of how much it inadvertently improves the 'fit' of links further down the kinematic chain. This can be thought of as initially ignoring a descent direction while searching for a potentially steeper descent direction.

To further improve the efficiency of the first outwards ordered iteration, the kinematic chain is broken is broken into a trunk (the body), and five branches (the legs, arms, and head). For the first iteration, the rotation groups are ordered such that optimization as a whole moves out along each of the five branches, and then outwards along the trunk. This was found to reduce the number of measurement function evaluations required by the optimizer.

In Section 6.6, optimization results are presented for the optimization scheme described here. These results are indeed sensitive to the choice of subproblem ordering scheme described here. If the inwards style ordering scheme is used for all iterations the legs converge to an undesirable locations in several frames of the first data set. It is believed that the legs are the most sensitive because of the combination of the lack of contrast between the legs and the background in the first data set, and also the presence of strong 'spurious' edges near the feet. The effects are more prevalent in the monocular case, probably because there is less image information in the monocular case. The edge metric for the first frame of the image set was shown in Figure 3.7.

6.3 Choosing Search Directions

6.3.1 Monocular Problem

In the monocular case the search directions of interest for a link are chosen to be: the *ambiguous depth direction*, and the *image displacement maximizing direction* (the direction which maximizes a link's displacement in the image). The ambiguous depth direction is not entirely ambiguous, in that movement in this direction changes the width and length of the link in the image. While this has only a limited effect on the measurement function, it is hoped that isolating the ambiguous direction will the depth resolvable to some extent. The experiments performed in Section 4.8.5 showed that the position of the local maxima with the largest area was quite close to the true position when using the JCP measurement approach, even in the presence of a monocular ambiguity. In a Newton based optimizer operating on all state variables simultaneously, these small changes in the objective function may be lost due to unrepresentative gradient and curvature information caused by image noise and other effects.

This approach has similarities to the kinematic jump process proposed by Sminchisescu and Triggs [91], which was discussed in Section 2.4.8. The key difference between these methods is

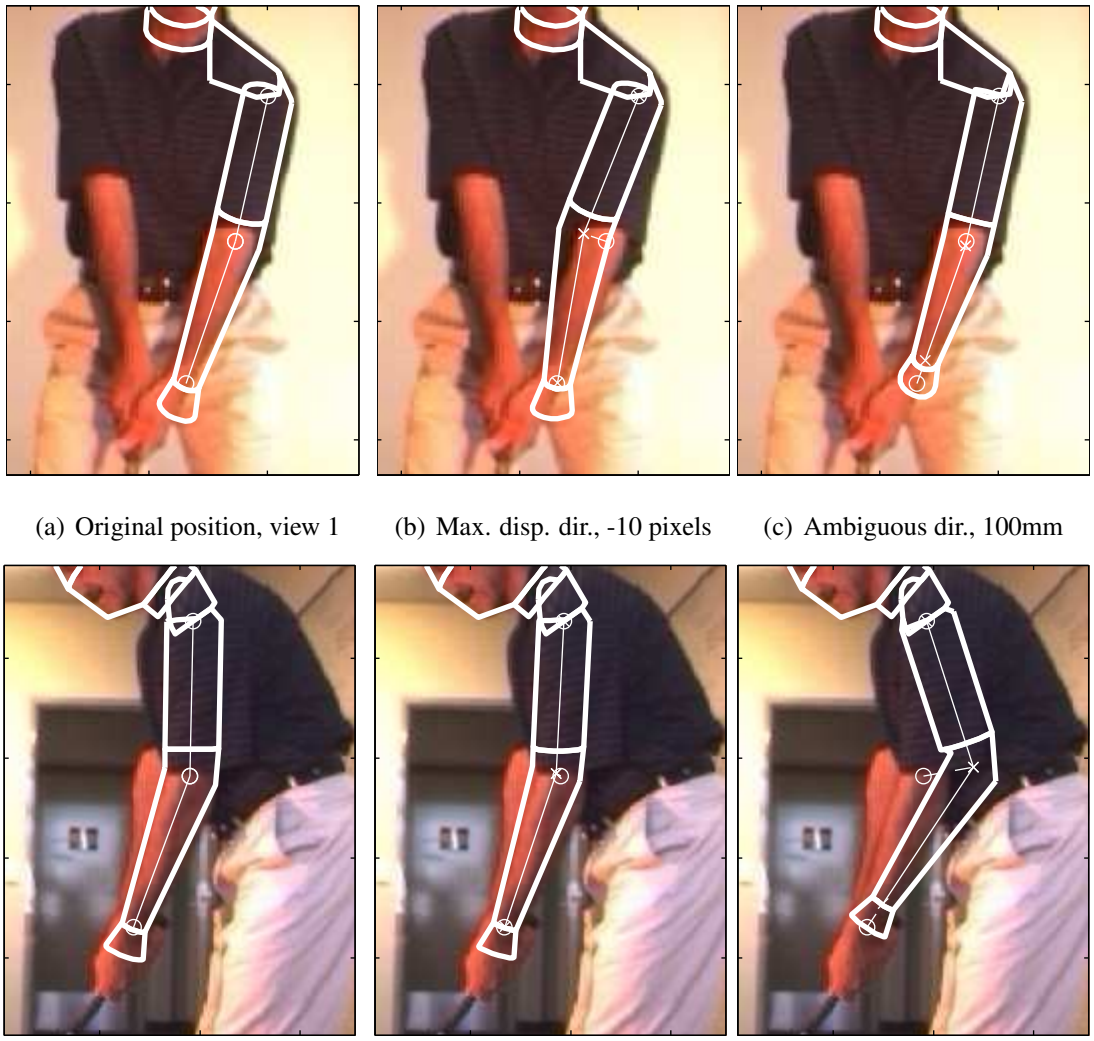
that the kinematic jump process operates at the global optimization level, effecting the particle placements and hence the seed locations for local optimizations when using a multiple hypothesis style global search algorithm [21]. The method presented here acts at the local optimization level, and it is hoped that by isolating ambiguous depth directions and searching them at larger search distances than are usually used, that the forwards–backwards link ambiguities can be sometimes resolved by the local optimizer itself. This will then reduce the number of particles which need to be propagated and optimized. It is also worth noting that the ambiguous search directions are straight lines in the Cartesian state space, whereas they are curves in the rotation state space.

For each of the clustered rotation groups described in Section 6.2, the *unconstrained* ambiguous depth direction for the link is calculated. This is the line from the camera's position, $C_1 \in \mathbb{R}^3$, to the link's end $\mathbb{L}_e$ in the current state estimate. Since the link may not be able to move in this direction, such as in the pathological case where the link points towards the camera, this direction is projected onto the plane $\mathcal{P} \in \mathbb{R}^3$ describing the directions the link can move, giving the *constrained* ambiguous depth direction. The plane $\mathcal{P}$ is the tangent plane to the link end's constraint sphere $\mathcal{S}$ discussed in Section 5.2, which will contain the link's end $\mathbb{L}_e$ and has surface normal given by the link's direction $\mathbb{L}_d = \mathbb{L}_e - \mathbb{L}_s$.

The image displacement maximizing direction is then the direction in $\mathcal{P}$ orthogonal to the constrained ambiguous depth direction, which is also necessarily orthogonal to the link's direction. Algorithm 10 gives a method for calculating the clustered joint angles, $\boldsymbol{\theta} = \{\theta_1, \theta_2\}^T$, for the ith rotation cluster, $\mathcal{G}_i^*$, giving points at a distance $\boldsymbol{\eta} = \{\eta_{pixels}, \eta_{mm}\}$ in the ambiguous and image displacement maximizing search directions. Figure 6.1 shows examples of states in the image displacement maximizing and ambiguous direction for the upper left arm relative to the front view.

In cases where the line from the camera's position to the link's end is (near to) collinear with the link's axis, *i.e.* where the link 'points' towards the camera, it is judged that no ambiguous depth direction exists. To address this, if these two lines are less than, say 20° (θ_{thresh} in Algorithm 10), from collinear, the distance along both directions is measured in pixels.

In the optimization experiment performed in Section 5.6.3, it was discussed that the spacing of the points used for central differencing gradients was chosen to 'noticeably' change the object's appearance in the image. This distance is very different for the two search directions described here. In the experiments performed in this Chapter, distances along the image displacement maximizing search directions (or more precisely the projection of it into the image) are measured in pixels. This allows for direct control of the change in the object's appearance in the image. Because moving in the ambiguous depth direction will generally only cause a slight change in the object's appearance in



(a) Original position, view 1 (b) Max. disp. dir., -10 pixels (c) Ambiguous dir., 100mm

(d) Unused view 2 (e) Max. disp. dir., view 2 (f) Ambiguous dir., view 2

FIGURE 6.1: Monocular tracking search direction examples

the image, larger search distances are used in this direction, and the distances are measured in millimeters. In the experiments presented later in this Chapter, $\tau = 4$ (from step 3d of Algorithm 10) was used so that the search distance used in an ambiguous direction is up to five times larger than the equivalent step in the more observable direction. It is hoped that using this (equivalently) much larger distances along the ambiguous depth direction than along the image displacement maximizing search direction, will provide a mechanism to allow the ambiguous direction to be resolved to some extent.

6.3.2 Two Camera Problem

In the monocular case, search directions were chosen to try to isolate the observable and unobservable directions. Ambiguous depth directions do not exist[1] in the two camera case however. Instead, search directions are chosen to maintain the link's position in one image while varying the link's projection

[1]Ignoring ambiguous depth directions caused by self occlusions.

1) Find the unconstrained ambiguous depth direction
$\mathcal{L}_a^* \in \mathbb{R}^3$ from the position of the link's end, $\mathbb{L}_e$, to the camera's position C_1, $\mathcal{L}_a^* = C_1 - \mathbb{L}_e$.
2) Find the constrained ambiguous direction $\mathcal{L}_a$ by projecting $\mathcal{L}_a^*$ onto the plane $\mathcal{P}$, defined by the link's end, $\mathbb{L}_e$, with surface normal equal to the link's direction, $\mathbb{L}_d$.
3) **if** *(desired direction = maximal image displacement direction)* **then**
3a) Set the search direction $\mathcal{L}_s$ to the direction in $\mathcal{P}$ orthogonal to the constrained ambiguous direction $\mathcal{L}_a$.
3b) Set *distance units = pixels*.
else Check if an ambiguous direction exists:
3a) Find the angle θ between $\mathcal{L}_a^*$ and $\mathbb{L}_d$
if $(\theta < \theta_{thresh})$ **then**
3b) Set *distance units = pixels* (there is no ambiguous direction).
3c) Set the search direction $\mathcal{L}_s = \mathcal{L}_a$.
else
3b) Set the search direction $\mathcal{L}_s = \mathcal{L}_a$.
3c) Set *distance units = millimeters* (there is an ambiguous direction).
3d) Increase the search distance, $\eta_{mm} = (1 + \tau \sin\theta)\eta_{mm}$
4) Find the point at a distance $\boldsymbol{\eta} = \{\eta_{pixels}, \eta_{mm}\}$ along the line $\mathcal{L}_s$:
if *(distance units = millimeters)* **then**
4a) Calculate the unconstrained point $R_n^* \in \mathbb{R}^3$ along $\mathcal{L}_s$ at a distance η_{mm} from $\mathbb{L}_e$.
else
4a) Project $\mathcal{L}_s$ into the image, giving the line $\mathcal{L}_{Im} \in \mathbb{R}^2$.
4b) Solve for the image point I_n on $\mathcal{L}_{Im}$ which is the desired image distance η_{pixels} from the projection of the link's end point $\mathbb{L}_e$.
4c) Back project I_n onto $\mathcal{L}_s$ giving the unconstrained point $R_n^* \in \mathbb{R}^3$.
5) Project R_n^* onto a sphere with radius equal to the link's length and centered at the link's origin, giving the constrained point R_n.
6) Solve for the clustered rotation angles $\boldsymbol{\theta} = \{\theta_1, \theta_2\}^T$ corresponding to R_n.

Algorithm 10: Search directions for the monocular case

in the other images.

While examining tracking failures during initial research, the author found many instances where the model/image correspondence was correct in one view, however poor in another view. This was the result of the combination of two possible problems. The first was that the measurement function reported a worse score after the author had manually adjusted the link's position, which motivated research into the edge measurement methods presented in Chapter 4. The second problem occurred when the measurement score improved after manually adjusting the link's position, meaning the local optimizer had failed to find a potential improvement. This second case led to the development of this local optimizer, which is designed to mimic the way in which the author checked for potential improvements to the match between the object and the image.

In the two camera problem, search directions are chosen which leave joint locations approximately constant in one image (the *current* image), while changing the location in the *other* images. These search directions are then designed to escape state space locations where a camera has locked onto edge features, *i.e.* where the hypothesized configuration looks correct from one view point. Newton based optimizers can have difficulty choosing a search direction, using the gradient and curvature information, which does not destroy the lock in one view, resulting in a search direction which does not have any potential improvements along it. There is a strong relationship between these search directions and a single camera's ambiguous depth direction when cameras are positioned orthogonally around the object.

Determining these search directions can be formulated as a constrained optimization problem, where the link's pixel displacement is minimized in one image, subject to achieving a specified displacement in another image. This can be implemented using a penalty based formulation [38], where the objective function comprises of two terms, the first a penalty for movement in the *current* image (term 1 in Equation 6.1), and the second a strict penalty term if the joint location has not moved the desired pixel distance in any of the *other* images (term 2 in Equation 6.1). Clustered joint angles $\boldsymbol{\theta} = \{\theta_1, \theta_2\}^T$ for a rotation cluster $\mathcal{G}_i^*$ are then chosen to minimize:

$$\boldsymbol{\theta}_{s1} = \arg\min_{\boldsymbol{\theta}} \left(\Delta_c + \tau \max_{i \in O} (\eta - \Delta_i)^2 \right) \tag{6.1}$$

with

$$\begin{aligned} \Delta_i &= \|I_n^i - I_o^i\| \\ I_n^i &= P_i\left(T(\mathbb{L}, \boldsymbol{\theta})\right) \\ O &= \{1, ..., c-1, c+1, ..., C\} \end{aligned}$$

where η is the desired pixel distance to move, I_o^i and $I_n^i \in \mathbb{R}^2$ are the original and new image coordinates of the end of the link $\mathbb{L}$ in image i, c is the *current* image, C is the number of images (cameras), O is the set of *other* images, $T(\mathbb{L}, \boldsymbol{\theta})$ is the transformation of link $\mathbb{L}$ by rotations $\boldsymbol{\theta}$, $P_i(.)$ is the link projection function for camera i, and $\tau \gg 0$ is a penalty scaling term.

Minimizing Equation 6.1 gives one solution, however a solution on the other side (in the image) of the original position is also desired. For each image i, a line is drawn from the link's original end position, I_o^i, to the link's end position for the solution $\boldsymbol{\theta}_{s1}$ found using Equation 6.1, $I_s^i = P_i\left(T(\mathbb{L}, \boldsymbol{\theta} = \boldsymbol{\theta}_{s1})\right)$. An extra penalty term is added to Equation 6.1 that penalizes the new solution if the link's end position projects on the positive side of the above described line:

$$\boldsymbol{\theta}_{s2} = \arg\min_{\boldsymbol{\theta}} \left(\Delta_c + \tau \max_{i \in O} \left(\left(\eta - \Delta_i\right)^2 + \alpha_i \right) \right) \tag{6.2}$$

with

$$\alpha_i \;=\; \max\left((I_s^i - I_o^i) \cdot (I_n^i - I_o^i), 0 \right)$$

where α is the projection distance along the line from the link's original end position, to the link's end position for solution $\boldsymbol{\theta}_{s1}$, and $\cdot$ denotes the dot product. The minimum error of the two equations is dependent upon both the camera geometry and the link's orientation.

In the two camera case, an approximate closed form solution to the optimization problems posed in Equations 6.1 and 6.2 can be found using Algorithm 11. In Algorithm 11, when the link 'points' towards the *current* camera i, the search direction for the link is forced to be orthogonal to the search direction used when image $j \in \{1, 2\}, j \neq c$, is the current camera. This reflects that it is impossible to move the link without significantly effecting its position in the current image. This is equivalent to the monocular case where an ambiguous search direction does not exist. The degree to which a link points at a camera is given by θ in Algorithm 11, with the threshold (θ_{thresh} in Algorithm 11), set to 20° in later experiments. In applications where the two cameras are not spaced orthogonally around the object, it may prove useful to always enforce that the search directions for each image are orthogonal.

Figure 6.2 shows the model states resulting from using Algorithm 11 to calculate points approximately ± 10 pixels along the search direction corresponding to the upper arm and the front camera. The pixel displacements were 9.99 and -9.78 in the first view and 0.23 and 1.44 in the second view, approximately what was desired. The model adjustment technique, discussed below in Section 6.4, is used so that the hand remains in its original position (the wrists are not articulated).

1) Find the lines $\mathcal{L}_i^* \in \mathbb{R}^3, i \in \{1,2\}$ from position of the link's end $\mathbb{L}_e$, to each camera's position C_i, $\mathcal{L}_i^* = C_i - \mathbb{L}_e$.

2) Find the constrained search directions $\mathcal{L}_i$ by projecting $\mathcal{L}_i^*$ onto the plane $\mathcal{P}$, defined by the link's end, $\mathbb{L}_e$, with surface normal given by the link's direction, $\mathbb{L}_d$.

3) Check the search direction is well defined for the *current* image c (the image in which it is desired to keep the link's position constant), by finding the angle θ between $\mathcal{L}_c^*$ and $\mathbb{L}_d$.

if $(\theta < \theta_{thresh})$ **then**

Force this search direction to be orthogonal to the search direction used when it is desired to move the link in this image:

3a) Set $\mathcal{L}_s$ to the line contained in $\mathcal{P}$ and orthogonal to $\mathcal{L}_o$, where o is the *other* image.

else

3a) Set $\mathcal{L}_s = \mathcal{L}_c$.

4) Find the point at a distance η_{pixels} along the line $\mathcal{L}_s$:

4a) Project $\mathcal{L}_s$ into the *other* image, giving the line $\mathcal{L}_{Im} \in \mathbb{R}^2$.

4b) Solve for the image point I_n on $\mathcal{L}_{Im}$ which is the desired image distance η_{pixels} from the projection of the link's end point $\mathbb{L}_e$.

4c) Back project I_n onto $\mathcal{L}_s$ giving the unconstrained point $R_n^* \in \mathbb{R}^3$.

5) Project R_n^* onto a sphere with radius equal to the link's length and centered at the link's origin, giving the constrained point R_n.

6) Solve for the clustered rotation angles $\boldsymbol{\theta} = \{\theta_1, \theta_2\}^T$ corresponding to R_n.

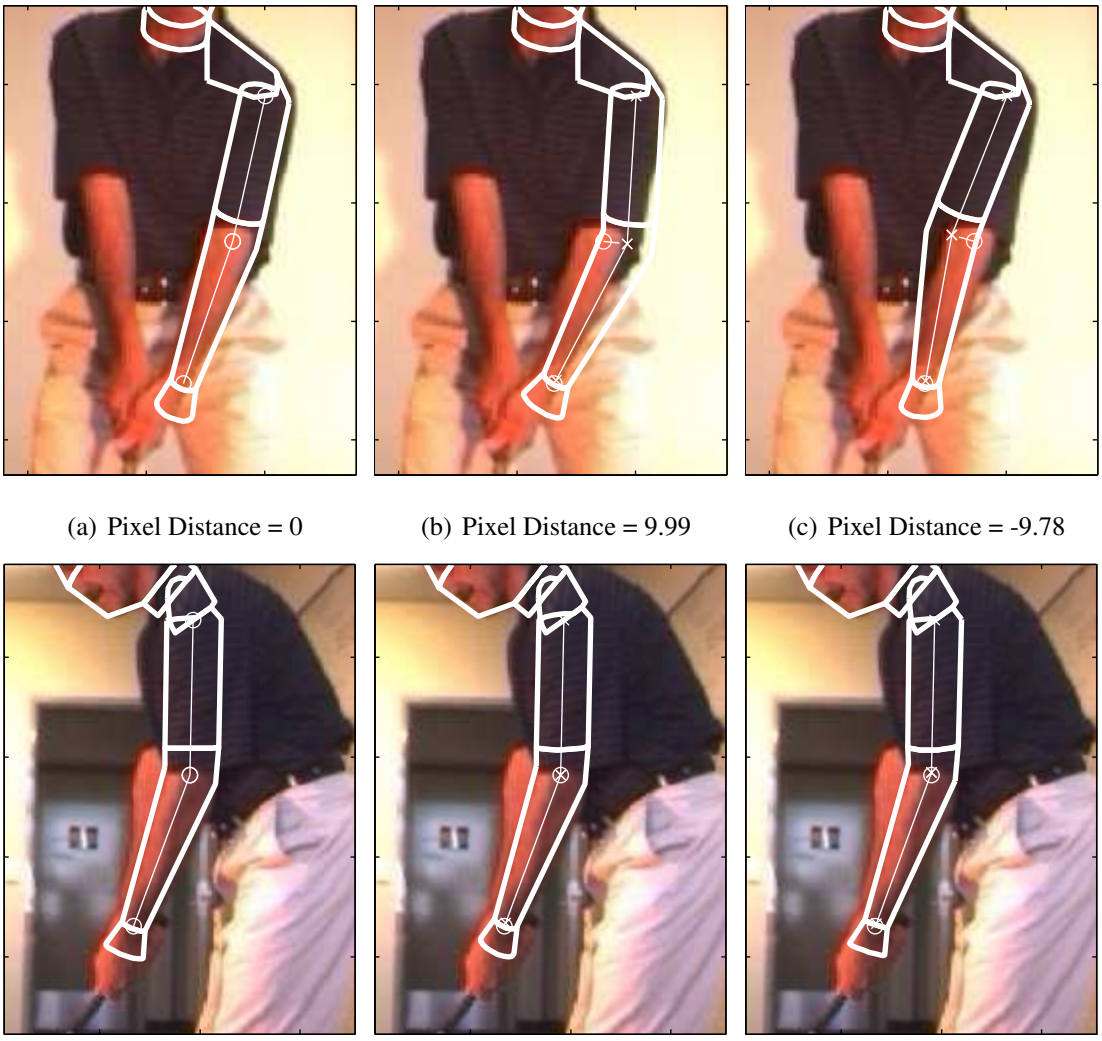
Algorithm 11: Search directions for the two camera case

6.4 Model Adjustment

To perform line searches in the search directions described above, it is necessary that links further down the kinematic chain are adjusted to remain in approximately the same position. While this naturally occurs in the Cartesian state space, extra information can be used when performing this style of local optimization on a given link to ensure a 'closer' match (or perhaps a more desirable match). In the case where a rotation cluster is being adjusted, joint angles are chosen to either:

1. Minimize the Euclidean distance between the link's end and the original position in $\mathbb{R}^3$.
2. Solve such that the link's end lies on a desired line, where the desired line is typically from the link's parent's start position to the link's original end position in $\mathbb{R}^3$.

Method one is the same method used to enforce geometric consistency when working in a Cartesian based state space, and is simply implemented by projecting the desired location of the link's end onto a sphere with radius equal to the link's length and centered at the link's new origin. In the experiments presented later in this thesis, the second method is always used for the forearms, lower legs, and the head, and additionally for any other link when the link's start has moved further away



(a) Pixel Distance = 0 (b) Pixel Distance = 9.99 (c) Pixel Distance = -9.78

(d) Pixel Distance Error = 0 (e) Pixel Distance Error = 0.23 (f) Pixel Distance Error = 1.44

FIGURE 6.2: Search direction maintaining the link's position from the second view point

from its original end position. Figure 6.2 showed an example where the hand's position was adjusted for elbow movement using the second method.

To adjust non-clustered rotations (the about axis rotation for the object model used throughout this thesis), the rotation angle is chosen to best align the link's rotated coordinate frame with its original coordinate frame. This is not be done in the standard Cartesian formulation as it relies on having a reference configuration to match.

Each of the above methods are performed using computationally inexpensive closed form solutions. The result of the articulated model adjusting to base link translations of 100mm in the y and z directions was shown in Figure 4.11. When some form of joint limits are applied to the link to be adjusted, the returned rotations are adjusted to conform to these limits. While this adjustment process could be improved by using this hierarchical solution to seed an estimator which solves for all adjusted links at once, this is not done as it is judged to be too computationally expensive.

6.5 The Local Optimization Algorithm

Starting with an initial seed position X_0 and associated measurement cost $c_{best} = -\log\left(p(Z|X)\right)$ where Z denotes the images, Algorithm 12 is run until the convergence of all rotation clusters $\mathcal{G}$.

1) Starting with an initial (seed) estimate X_0, set *converged* = *false*, $X = X_0$, and
$c_{best} = -\log(p(Z|X_0))$.
2) Loop until convergence:
while $\neg$ *converged*
3) Set *converged* = *true*
4) Loop through each rotation group in the order described in Section 6.2.1:
foreach *rotation group* $\mathcal{G}_i \in \mathcal{G}$
if $\mathcal{G}_i$ is a *rotation cluster*, $\mathcal{G}_i \in \mathcal{G}^*$ **then**
5) Perform line searches:
foreach *search direction*
5a) Calculate the search direction as per Section 6.3.
5b) Perform a 1D line search as per Section 6.5.1 seeded at X returning the new state estimate X with cost c.
5c) If an improvement was found, set *converged* = *false*.
else
5a) Perform a damped Newton optimization on the Cartesian state space variables corresponding to $\mathcal{G}_i$, returning the new state estimate X with cost c.
5b) If an improvement was found, set *converged* = *false*.
6) **return** X.

Algorithm 12: Optimization Algorithm

For primarily computational reasons, the Newton optimizations are only performed in the first and last iteration (step 2) of Algorithm 12. Due to the subproblem ordering scheme discussed in Section 6.2.1, in the first iteration these optimizations are performed after all of the subproblems searched using the camera geometry inspired search directions for the kinematic chain's branch have been calculated.

Another modification made to Algorithm 12, not shown for simplicity, is that when a link is entirely occluded from one viewpoint in the two camera case, the search directions for the applicable rotation group are calculated as per the monocular case.

6.5.1 1D Line Searches

The optimization algorithm presented in Section 6.5 is based on performing a 1D line search for each search direction for each clustered rotation group. The question then remains of how to best

perform these line searches. As discussed in Section 2.3.4, there may only be a small region where the measurement function can be locally modelled as a quadratic function, it is questionable if using the first and second derivatives of the measurement function will be helpful. Furthermore, following the discussion in Section 4.11 it is still expected that there are states where the measurement function derivatives are not well defined.

Instead of utilizing the first and second order derivatives of the measurement function, a *blind search* style is preferred for searching each direction. This blind search simply calculates potential updates at small distances in either direction along the search direction, accepting a potential update if it has lower cost than the current state estimate. The blind search algorithm is shown in Algorithm 13.

1) Starting with an initial (seed) estimate X_0, set *converged* = *false*, $X = X_0$, and
$c_{best} = -\log(p(Z|X_0))$.
2) Loop through the set of trial distances $\boldsymbol{\eta}$ from largest to smallest until convergence:
2a) Choose $\eta_i = \max(\boldsymbol{\eta})$.
while $\neg$ *converged*
2b) Using Algorithm 10 or 11, generate two trial points X_1 and X_2 at distances $\pm\eta_i$ from X.
2c) Calculate the costs c_1 and c_2 of X_1 and X_2 respectively.
if $(c_{best} - \min(c_1, c_2)) \geq$ *Function Tolerance* **then**
Update X and c_{best} with the appropriate trial point.
else
Choose the next smallest $\eta_i \in \boldsymbol{\eta}$ if available, or else set *converged* = *true*.
3) **return** X.

Algorithm 13: 1D Blind Search Algorithm

Obvious modifications can be to Algorithm 13 so that a trial point is not generated at the approximate location of any previous state estimates.

This blind search method is similar to the simulated annealing method discussed in Section 2.4.4. This approach differs from other annealing processes in that each particle location (trial point in Algorithm 13) is deterministically drawn to lie in an 'interesting location' inferred from the camera geometry, rather than drawing particles stochastically using: predetermined noise dynamics [31], particle set characteristics [32], or measurement function curvature information [92].

In the experiments presented in this chapter, for search directions measured in pixels, $\boldsymbol{\eta} = \{8, 4\}$ pixels, and for the ambiguous depth direction, $\boldsymbol{\eta} = \{4, 2\}$ millimeters. In conjunction with the ambiguous direction scaling in Algorithm 10, trial points in the ambiguous depth directions were generated at distances of up to 20 millimeters. As a rule of thumb, the author recommends using the same distances as used for central differencing gradients for the search directions that are not measured in pixels. Search directions measured in pixels can be set to give the minimum 'observable'

difference in image appearance.

6.6 Camera Geometry Optimization Evaluation

To test the effectiveness of the optimization method presented in this chapter, the experiment from Section 5.6.3 was repeated to compare this optimization method with the standard damped Newton approaches in the rotation and Cartesian spaces. In this experiment, optimizer seed position were generated by propagating a constant velocity dynamic model on user labelled training data.

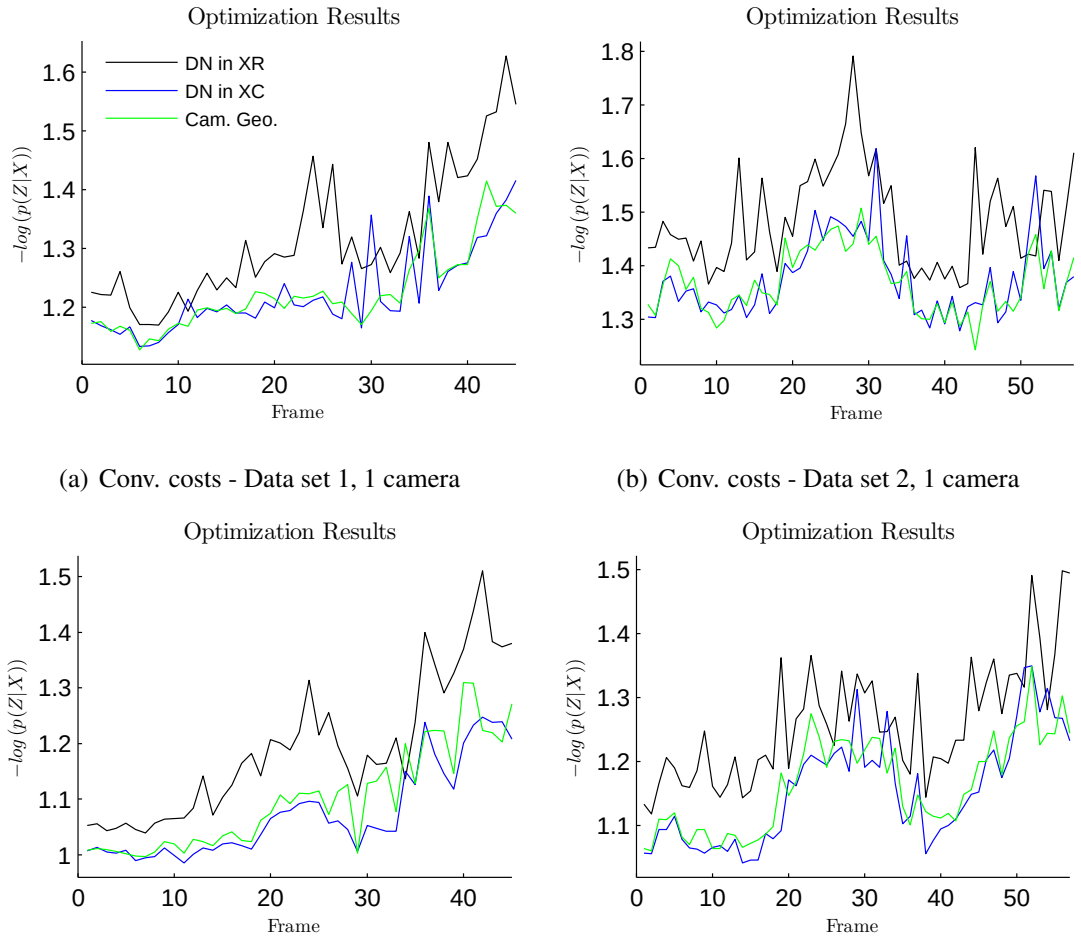
Fine Tuning Search Distances

As discussed in Section 5.6.3, the convergence tolerance for the optimization was set to 10^{-6}, which is orders of magnitude smaller than would be used during regular tracking. To compensate for this abnormally high optimization accuracy, once convergence was achieved for the proposed optimizer, the search distances used were reduced and the optimizer run again. These *fine tuning* distances were set to $\boldsymbol{\eta}_{tune} = \{2, 1\}$ pixels, and $\boldsymbol{\eta}_{tune} = \{1, \frac{1}{2}\}$ millimeters where appropriate. Figure 6.2 showed an example of a pixel displacement of 1.44 pixels, which in the author's opinion is not discernable and hence unnecessary during regular tracking.

Algorithm 12 as presented assumes that the optimal solution to each subproblem is the optimal solution to the whole problem, and so the interdependence between subproblems is only enforces by the measurement function. This is equivalent to enforcing a stricter Hessian sparsity structure for the Newton optimizers. Another modification aimed towards increasing the resolution of the optimizer is to modify Algorithm 12 to allow subproblems to be solved recursively. When a state X is found in step 5b that improves the fit of the link associated with the current subproblem, however decreases the fit of the overall model, the subproblems associated with the current link's neighbours are solved assuming the state X, giving the new potential update X_{recurse}. If X_{recurse} is an improvement on the current state estimate, it is accepted. This modification is only used in conjunction with the fine tuning distances however, as the author has found that these updates do little to change the object's image appearance, but rather just change the convergence value. This modification may prove useful however for measurement functions that do not enforce edge consistency between links.

Results

Figure 6.3 shows the optimization results in the monocular and two camera cases, comparing the proposed camera geometry approach (Cam. Geo.) with the damped Newton optimizations (DN) in



(a) Conv. costs - Data set 1, 1 camera

(b) Conv. costs - Data set 2, 1 camera

(c) Conv. costs - Data set 1, 2 cameras

(d) Conv. costs - Data set 2, 2 cameras

FIGURE 6.3: Camera Geometry Optimization Comparison

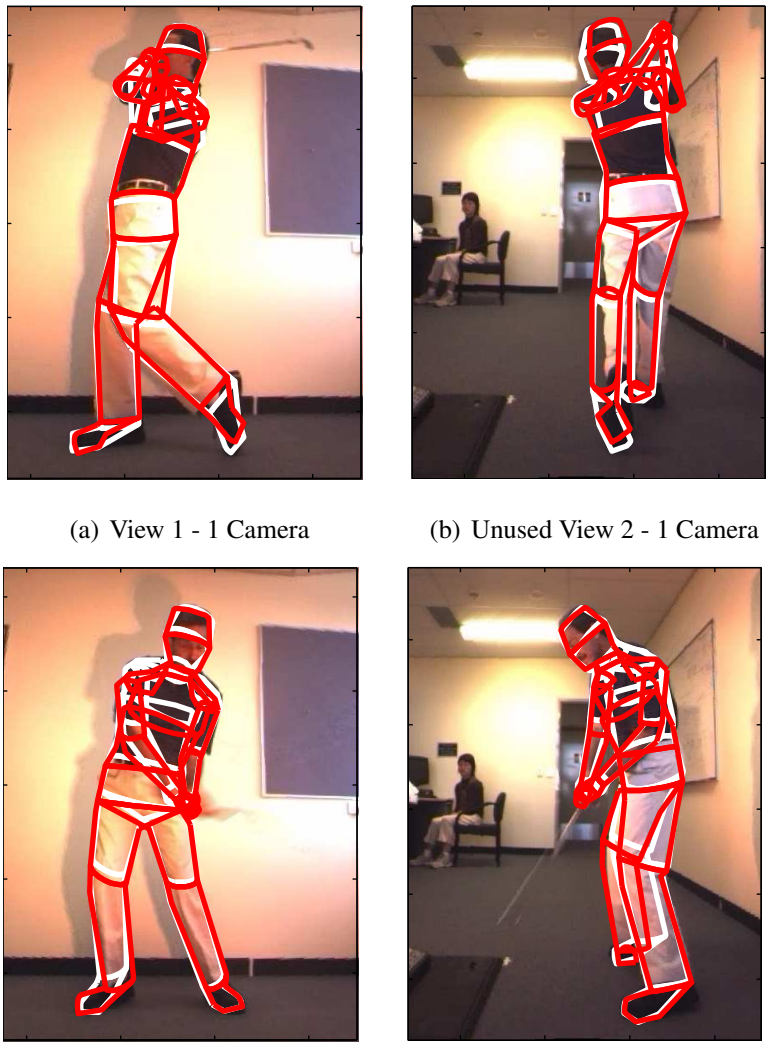
Type	Mean Cost	Std	Total Cost	Mean func. evaluations	Total func. evaluations	% Best
1 Camera						
MATLAB in $\mathbf{x}^{\mathcal{R}}$	1.4533	0.1540	148.2334	42606	4346×10^3	0%
DN in $\mathbf{x}^{\mathcal{R}}$	1.4102	0.1323	143.8413	12114	1236×10^3	2%
DN in $\mathbf{x}^{\mathcal{C}}$	1.3089	0.1042	133.5125	20736	2115×10^3	57%
Cam. Geo.	1.3054	0.0953	133.1487	5649	576×10^3	41%
2 Cameras						
MATLAB in $\mathbf{x}^{\mathcal{R}}$	1.2815	0.1285	130.7087	41911	4275×10^3	0%
DN in $\mathbf{x}^{\mathcal{R}}$	1.2286	0.1123	125.3194	12602	1285×10^3	1%
Cam. Geo.	1.1350	0.0876	115.7721	5430	554×10^3	24%
DN in $\mathbf{x}^{\mathcal{C}}$	1.1190	0.0924	114.1360	19056	1944×10^3	75%

Table 6.1: Camera Geometry Optimization Comparison

the Cartesian $\mathbf{x}^{\mathcal{C}}$ and rotation state spaces $\mathbf{x}^{\mathcal{R}}$. These graphs are summarized in Table 6.1. The camera geometry optimization scheme presented in this chapter outperforms the commonly used Newton optimization in $\mathbf{x}^{\mathcal{R}}$ in all but 5 of the 204 frames, while requiring only $\approx 40\%$ of the measurement

function evaluations. These results also show that the proposed camera geometry optimization method performed slightly worse than the damped Newton optimization in the Cartesian space however.

This decreased performance (in terms of convergence cost) of the proposed optimizer is perhaps to be expected. The convergence criteria used for the two methods is different. While both converge the measurement function to 10^{-6}, the problem decomposition used in the camera geometry method effectively means that the improvement in the 'fit' of a single link must change the measurement function by 10^{-6}, whereas in the Newton optimizers a potential update can manipulate all links in the kinematic chain.



(a) View 1 - 1 Camera (b) Unused View 2 - 1 Camera

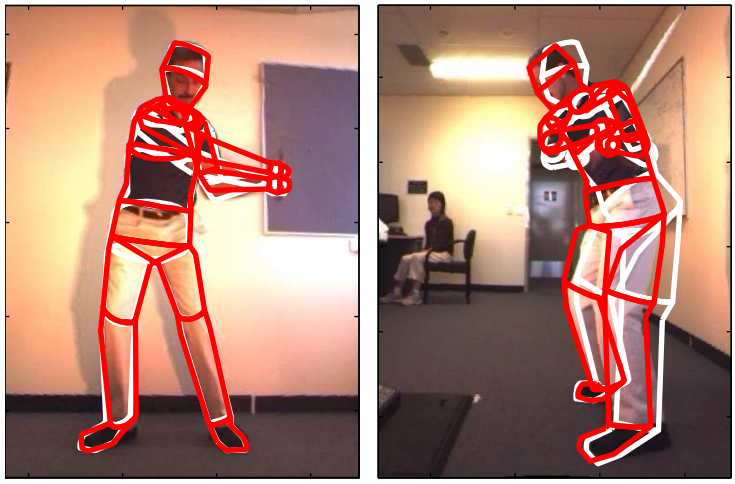
(c) View 1 - 2 Cameras (d) View 2 - 2 Cameras

FIGURE 6.4: Converged states for the camera geometry (red) and Cartesian (white) optimizations

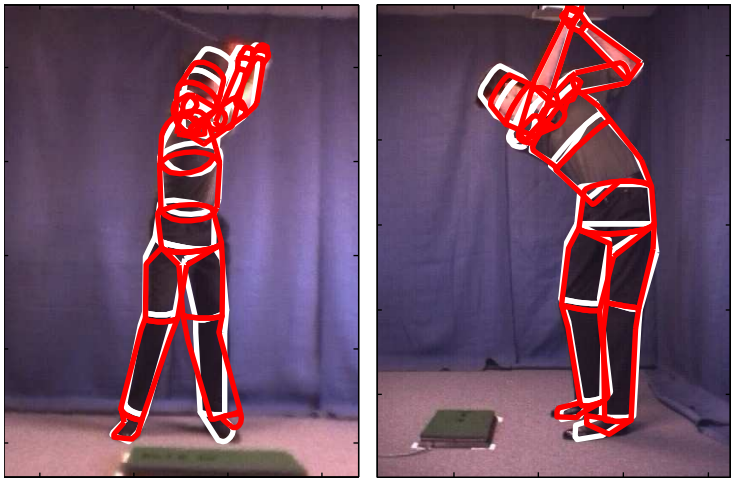
This may explain why the camera geometry optimizer recorded a lower mean but higher median score than the Newton optimizer in $\mathbf{x}^{\mathcal{C}}$. The Newton optimizers ability to make these small corrections means the Newton optimizer outperforms the camera geometry optimizer most of the time. In some cases however, the camera geometry optimizer, due to its modified ambiguous depth direction search

or because the Newton optimizer has failed to calculate the potential update direction, significantly outperformed the Newton optimizer. This is also true of the two camera optimizations, where there is no expectation that the camera geometry should ever outperform the Newton optimizer unless the Newton optimizer has failed to calculate the potential update direction.

Figure 6.4 shows the converged states at frame 42 of data set 1 in the monocular case and frame 32 of data set 1 in two camera case. These frames was chosen as they recorded the worst performance of the proposed camera geometry (red) optimization compared to the Newton optimization in Cartesian space (white) in both sequences for the monocular and two camera cases. Figure 6.4(a) shows a slight miss-estimation of the position of the left arm, while Figure 6.4(d) shows a slight miss-estimation of the position of the left shoulder in the two camera case.



(a) View 1 - 1 Camera (b) Unused View 2 - 1 Camera



(c) View 1 - 2 Cameras (d) View 2 - 2 Cameras

FIGURE 6.5: Converged states for the Cartesian (red) and camera geometry (white) optimizations

Figure 6.5 shows the converged states at frame 30 in the monocular case and frame 33 in the two camera case. These frames were chosen as they corresponded to the worst performance of the Newton

optimization in Cartesian space (now red) to the proposed camera geometry (now white) optimization. In the monocular case, Figure 6.5(a) shows a miss-estimation of the position of the right arm, while in the two camera case Figures 6.5(c) and 6.5(d) show miss-estimations of the positions of the head and left leg.

This comparison of the worst relative performance of the two optimizers indicates that the proposed camera geometry method has a better worst case performance. There was a potential issue that the problem decomposition used by the camera geometry optimizer would result in descent directions which were not the steepest descent direction for the whole problem, and so the optimizer could fall into smaller local modes. The results presented here show no indication of this. This indicates that the subproblem ordering scheme discussed in Section 6.2.1 performs as intended.

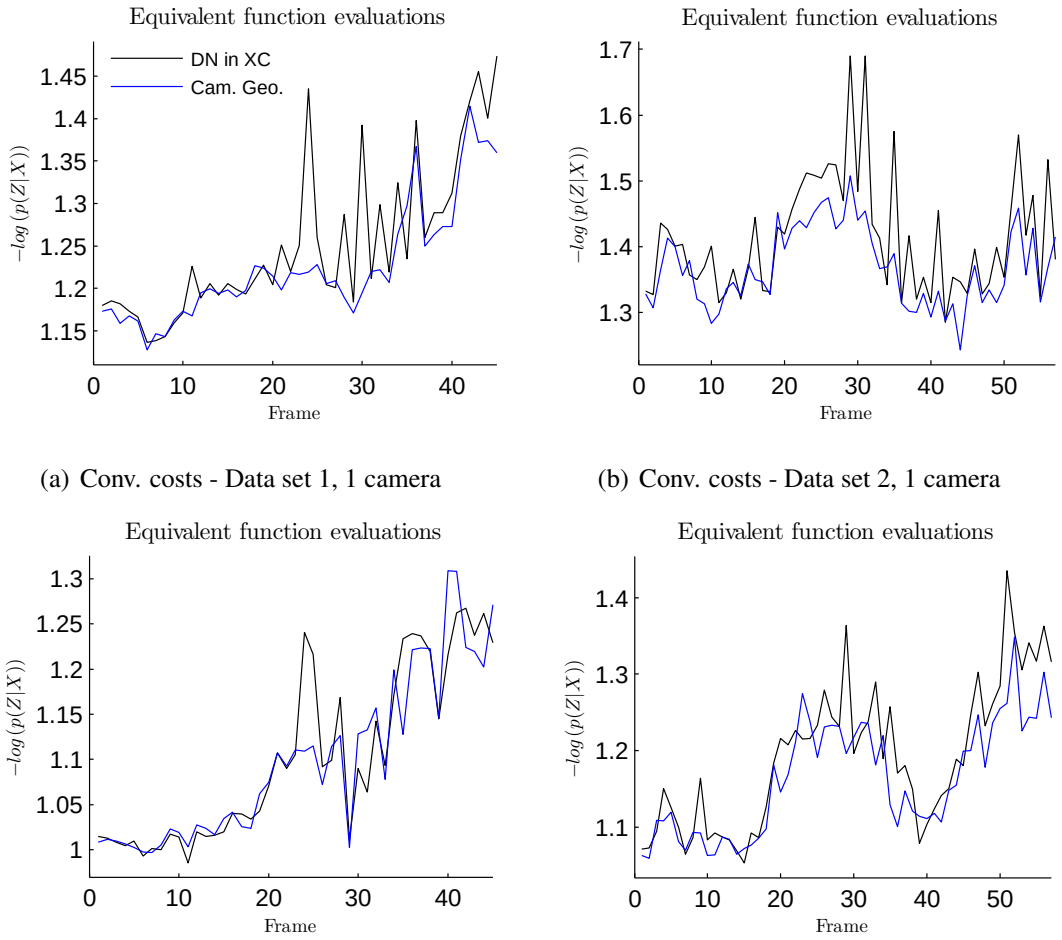
6.6.1 Convergence Rates

Given the similarity of convergence costs between the camera geometry optimizer and the Newton optimizer in Cartesian space, and the differences in convergence criteria, it is perhaps preferable to consider the convergence rates rather than the overall convergence score. Table 6.2 and Figure 6.6 show the progress of the damped Newton optimization in the Cartesian space when it is limited to the same number of measurement function evaluations as used by the camera geometry optimization method, using a linear interpolation where required.

Type	Mean Cost	Std	Total Cost	Mean func. evaluations	Total func. evaluations	% Best
1 Camera						
DN in $\mathbf{x}^{\mathcal{C}}$	1.3421	0.1208	136.8949	5649	576×10^3	22%
Cam. Geo.	1.3054	0.0953	133.1487	5649	576×10^3	78%
2 Cameras						
DN in $\mathbf{x}^{\mathcal{C}}$	1.1525	0.1039	117.5591	5430	554×10^3	41%
Cam. Geo.	1.1350	0.0876	115.7721	5430	554×10^3	59%

Table 6.2: Convergence Costs at Equal Function Evaluations

These results show that the camera geometry method does converge faster than the damped Newton method in Cartesian space. A Wilcoxon signed rank test [109] reveals a $\ll 10^{-6}\%$ and $< 10^{-2}\%$ probability that the differences between the computationally balanced convergence scores of the two methods have zero median in the monocular and two camera cases respectively. The Newton optimization in the Cartesian space was shown to converge faster than the rotation space equivalent in Chapter 5. This shows that performance, in terms of computational complexity, can be improved



(a) Conv. costs - Data set 1, 1 camera

(b) Conv. costs - Data set 2, 1 camera

(c) Conv. costs - Data set 1, 2 cameras

(d) Conv. costs - Data set 2, 2 cameras

FIGURE 6.6: Convergence Costs at Equal Function Evaluations

by using problem specific information to determine search directions, rather than the more general practice of inferring search directions from the objective function's derivatives.

6.6.2 Joint Limits

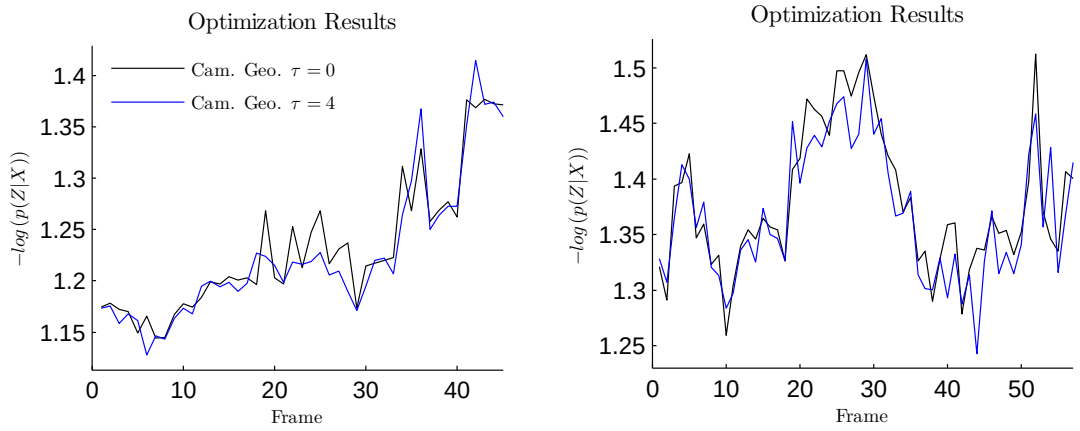
The optimization results given in the previous section were performed without any joint limit constraints imposed on the optimizer. Since these joint limits are dependent upon the interaction of the subproblems used in the proposed camera geometry based optimizer, the optimization experiment was performed using the joint limits as discussed in Section 5.6.4. Because different versions of the joint limits are used in the Cartesian and rotation spaces, the proposed camera geometry optimizer is only compared with the Newton optimizer in the Cartesian space.

Tables 6.3 shows the results of this experiment. Comparing Table 6.3 to Table 6.1, reveals improved relative performance of the proposed camera geometry optimizer. This shows that the subproblem decomposition is compatible with joint limits. Both of these optimizers still outperformed the Newton optimizer in the rotation space despite the joint limit constraints.

Type	Mean Cost	Std	Total Cost	Mean func. evaluations	Total func. evaluations	% Best
1 Camera						
DN in $\mathbf{x}^{\mathcal{C}}$	1.3339	0.1172	136.0622	15837	1615×10^3	43%
Cam. Geo.	1.3219	0.1047	134.8297	5558	567×10^3	57%
2 Cameras						
Cam. Geo.	1.1440	0.0951	116.6832	5584	570×10^3	34%
DN in $\mathbf{x}^{\mathcal{C}}$	1.1427	0.1115	116.5563	15598	1591×10^3	66%

Table 6.3: Optimization Results Enforcing Joint Limits

6.6.3 Ambiguous Direction Scaling



(a) Conv. costs - data set 1, 1 camera

(b) Conv. costs - data set 2, 1 camera

FIGURE 6.7: Camera geometry optimization, variation in τ

The ambiguous direction scaling term, τ from Algorithm 10, controls the distance at which a measurable difference in the image appearance occurs in an ambiguous depth direction relative to an observable direction. To determine if there is an advantage in scaling the distance along the ambiguous direction, the camera geometry optimizer was run without this scaling term, $\tau = 0$. Figure 6.7 shows the performance differences between setting $\tau = 0$ and $\tau = 4$ (the default), which are summarized in Table 6.4.

Type	Median cost	Mean cost	Std	Total Cost	% Best
Cam. Geo. $\tau = 0$	1.3300	1.3142	0.0974	134.0438	33%
Cam. Geo. $\tau = 4$	1.3149	1.3054	0.0953	133.1487	67%

Table 6.4: Optimization comparison for different τ, 1 camera(s)

Not utilizing the ambiguous direction scaling term resulted in worse optimizer performance in terms of the mean and median convergence costs. A Wilcoxon signed rank test [109] reveals a $\approx$

0.01% probability that the difference between the convergence scores of the two methods has zero median, showing the ambiguous direction scaling term has a statistically significant effect. Without this scaling term, the camera geometry optimization method is outperformed by the Newton optimizer in Cartesian space in terms of both mean and median. No study has been performed to determine an optimal value for τ, as it scales another free parameter (η_{mm} from Algorithm 10) which has also not been studied.

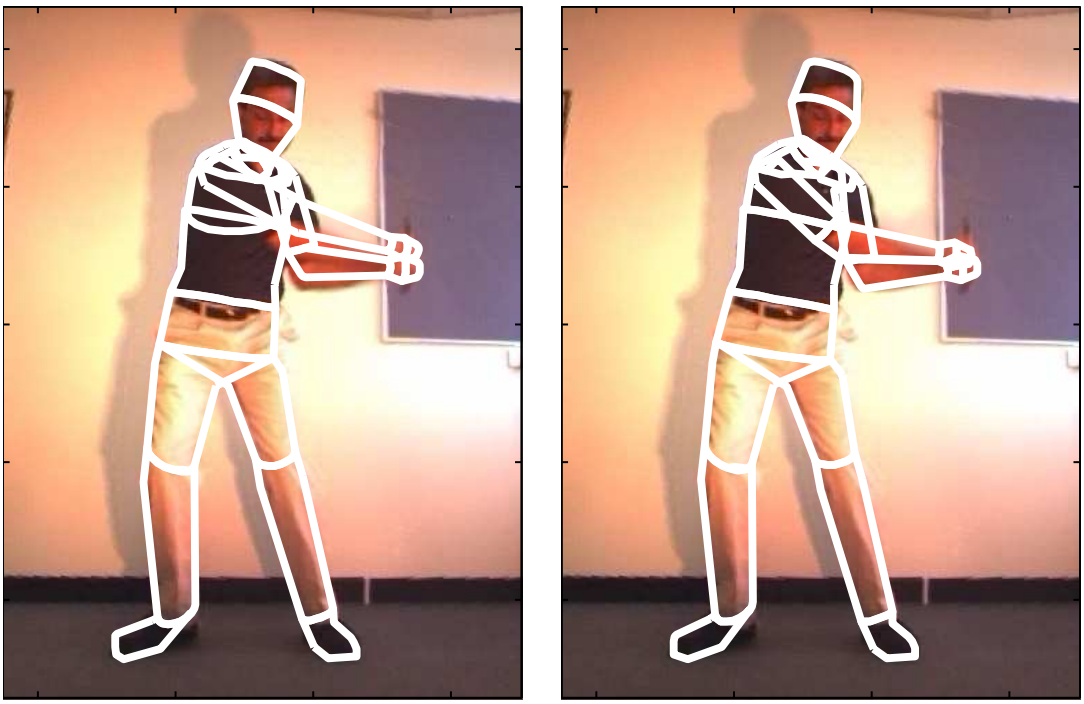
6.7 Escaping local modes

Only slight modifications are required to extend the camera geometry optimization method presented in this chapter beyond a truly local optimizer. In the results presented in Section 6.6, special smaller *fine tuning* distances were used to increase the resolution of the optimizer. To escape local modes, special larger *mode breaking* distances may be used. These distances are dynamically selected to attempt to escape local modes in two ways.

The first method pertains only to monocular tracking, where the mode breaking search distance along the ambiguous depth direction is dynamically chosen to place the link in the other of its forwards–backwards ambiguous poses, as discussed in Section 6.3.1. This is similar to the kinematic jump process proposed by Sminchisescu and Triggs [91], except in this case the ambiguity is being searched at a lower 'level' of a multiple hypothesis style global search algorithm [21]. Furthermore this ambiguity is only considered for a single link, whereas the kinematic jump process considers the ambiguities for chains of links. This technique is also used in two camera tracking when a link is unable to be observed in an image due to self occlusions.

The second method is the result of the distances along all search directions, excepting the ambiguous depth directions, being measured in pixels. The mode breaking search distances in this case are chosen to be the same as the width of the link in an image. This gives the optimizer some power to escape local modes caused by mismatched edge assignments, *i.e.* where the edge matched by the measurement function to one side of the link actually corresponds to the other side of the link. Figure 6.8 shows an example of escaping a local mode in this fashion. Figure 6.8(a) shows the converged state for the Newton optimizer in Cartesian space at frame 30, which recorded the worst performance compared with the camera geometry optimizer in the monocular case. Figure 6.8(b) shows the result of using this state to seed the camera geometry optimizer utilizing mode breaking search distances. The optimizer escaped the local mode by using the upper arm's image displacement maximizing search direction. All subsequent updates were the result of the standard local optimization procedure.

These mode breaking distances were not used in the results presented in Section 6.6 because, as



(a) Pre mode break search (b) Post mode break search

FIGURE 6.8: Escaping a local mode

shown in Figure 6.8, this mode breaking approach can be applied to the converged state of any of the optimizers. These mode breaking distances are incorporated into the tracking results presented in Chapter 7 however.

6.8 Discussion

In this chapter, a novel local optimization method has been proposed for performing local optimizations for visual tracking problems. This method is based upon choosing search directions based upon camera geometry, rather than inferring search directions from localized measurements function gradient and curvature information. The proposed camera geometry optimizer outperformed a standard damped Newton optimization in the rotation space. In the monocular case, the camera geometry optimization is roughly equivalent to the damped Newton optimization in the Cartesian space proposed in Chapter 5, with perhaps a small improvement in computational performance. In the two camera case, there seems little reason to prefer the camera geometry optimizer to the Newton optimizer formulated in the Cartesian space. The extension of the proposed optimizer to include special 'mode breaking' search distances showed that it can be used in conjunction with any optimizer however.

Anecdotal evidence suggests that due to the derivative free nature of the camera geometry optimizer, its relative performance improves when a less computationally intensive measurement function, *i.e.*, one with more local maxima, is used. In this case the search directions chosen by the camera geometry optimizer often have far fewer local maxima than the search directions chosen by a Newton optimizer. In all cases the camera geometry optimizer finds local maxima larger than the local maxima corresponding to ground truth state more often than the other methods, highlighting undesirable measurement function behaivour.

The discussion on computational costs of the competing methods is predicated upon using full numerical methods to calculate the measurement function's gradient and curvature in the Newton style optimizers. While an approximation to these was used at the optimizer level in Algorithm 9, an approximation at the measurement function level is often available, such as was derived for the cost path edge measurement method in Section 4.9. While this style of approximation was not trialled due to the difficulties in deriving an approximation for the region consistency measure discussed in Section 3.5.2, it has been used by other authors such as Sminchisecu and Triggs [90]. These approximations are typically built upon the assumption that the set of image features (and occlusions) do not change when the model is perturbed slightly, meaning the measurement function's derivatives are well defined at all points in the state space. As discussed in Section 2.3.4, there is potentially a dense set of states where this assumption does not hold when discrete line searches are as used. The JCP measurement function presented in Chapter 4, was proposed because it reduces the variability of found image features between 'close' state space points, and as such is better suited to these assumptions.

The author knows of no reason why calculating an approximated Jacobian and Hessian would improve the optimizer's performance in terms of convergence costs. It is still an open question however as to whether approximating the Jacobian and Hessian in this manner could decrease the effective computational cost of the Newton optimization in Cartesian space by enough to give the Newton optimizer in Cartesian space computational parity with the proposed camera geometry based optimization method, particularly given the Newton optimizer already utilizes a BFGS style Hessian approximation.

7

Tracking Evaluation

In this Chapter tracking is performed on a challenging image sequence, where the methods proposed in Chapter 4–6 are integrated into an existing tracking algorithm. The image sequence to be tracked is of a golfer performing a golf swing. Tracking results assume a known object location in the first frame, and that the object model was known apriori (no colour / texture information), as learning object model parameters is beyond the scope of this thesis as discussed in Section 1.3.

The object location and shape parametrization are assumed to be known in the first frame as the author considers first frame initialisation to be a pose estimation problem, which is distinct from the tracking problem investigated in this thesis. While some tracking methodologies also handle the pose estimation, this view is supported by a review performed by Sigal, Balan, and Black [84] which found that just five of the thirty five reviewed trackers perform both pose estimation and tracking.

A golf swing presents a challenging motion because there is a significant about axis rotation for the hips and spine during the swing. These rotations are difficult to infer in a causal fashion even for a human operator, which was the reason for the raw (causal) and smoothed (non-causal) training sets discussed in Section 5.4. Furthermore, the highly dynamic nature of a golf swing gives localized object speeds of approximately 45 kph, corresponding to inter frame pixel distances of greater than 100 pixels. Of note no golf swing specific information has been built into the tracker,

where it could be incorporated into the system dynamics (e.g. Urtasun, Fleet, and Fua [99]), and / or into the measurement model (e.g. Lepetit, Shahrokni, and Fua [61]).

The image sequence itself is challenging due to the combination of high and low contrast edges, where the low contrast edges of the pants closely match the background. This makes the task of interpreting the about axis rotation of the hips even more problematic. Furthermore, the pants' pockets make the occluding contour in this region highly deformable increasing the difficulty of inferring the about axis rotation. The cameras used are not specialized high speed cameras, which combined with the highly dynamic motion resulted in poor quality images, which was the reason optical flow data has not been used as discussed in Section 3.5.1.

7.1 Tracking Algorithm

The tracking algorithm is based on the covariance scaled multiple hypothesis tracker (CSS) as proposed by Sminchisescu and Triggs [90], as the modular nature of this tracker allows the work presented in this thesis to be readily incorporated into the tracker. The covariance scaled sampling aspect of the CSS algorithm algorithm is also of particular use for inferring the about axis rotations of the hip and spine.

7.1.1 Measurement Function

Following the work presented in this thesis, the following modifications have been made to the measurement function of the CSS algorithm:

- Use of the joined cost path (JCP) edge measurement approach from Chapter 4

These update was required to make the tracker successful enough for an analysis of tracking results to be meaningful. The complete measurement function also includes the region consistency score from Section 3.5.2, the occlusion likelihood score from Section 3.1.3, and a joint limit penalty function. These joint limits were quite liberal due to the expected motion, with the strictest being the upper back's 'backwards' bend limited to $30°$. No body part interpenetration avoidance or 'soft' joint priors were used.

7.1.2 Temporal Propagation

The tracking results used below propagated 10 Gaussian mixtures to represent the posterior distribution, which were propagated using a constant velocity dynamic model, and 1000 new samples drawn

to represent the new prior distribution. The covariance matrix was scaled before sampling by 2×10^{-2} and 4×10^{-2} in the monocular and two camera case respectively. This is notable lower than used by Sminchisescu and Triggs [90], however should not be directly compared because the measurement function used in these tracking results was developed in Chapter 4, and so is different to the measurement function used by Sminchisescu and Triggs.

The tracking results are somewhat insensitive to these choices. When a particle is initially selected for optimization, it was compared to all other particles which were already optimized in the current frame. If this particle is too 'close' to another optimized particle, it is not optimized and the next most probable particle is selected. Similarly, as advocated by Sminchisescu and Triggs, an optimized particle (mixture in this case) is discarded if it is too close to another previously optimized particle. The evaluation of the 'closeness' of particles was performed using a modified infinity norm based approach, where a particle was judged too close if none of the angular state dimensions were further apart than 10°, and if no limbs had Cartesian differences of more than 5cm. A such, choosing a small covariance scaling simply results in many particles being discarded before the optimization process, *i.e.* inefficient prediction.

Equivalently to the measurement function, prior probabilities for each dimension (in a space where each dimension of the Gaussian mixture is independent) were combined using a sum of squared differences approach. Using p_i, $i \in \{1, \ldots, d\}$ to denote the probability for each independent dimension, then the prior $p(\mathbf{x}^t|\mathcal{N}(\mu^*, \Sigma^*))$ is given by:

$$\log\left(p(\mathbf{x}^t|\mathcal{N}(\mu^*, \Sigma^*))\right) = -\frac{1}{d}\sum_{i=1}^{d}(1 - p_i)^2 \tag{7.1}$$

with $\mu^* = 2\mu_i^{t-1} - \mu_j^{t-2}$, and $\Sigma^* = 4\Sigma_i^{t-1} + \Sigma_j^{t-2}$ as per the constant velocity model, and previous Gaussian mixtures, $\mathcal{N}(\mu_i^{t-1}, \Sigma_i^{t-1})$, $\mathcal{N}(\mu_j^{t-2}, \Sigma_j^{t-2})$ were selected using the Bhattacharyya distance as advocated by Sminchisescu and Triggs.

7.2 Datasets

Tracking evaluation was performed on the data set 1, show in Figure 1.1, as it provided a difficult set of observational features, however a range of tracking alternatives were "broadly" successful on it. Data set 2 proved too challenging to allow reasonable interpretation of the differences in tracking errors produces by the different trackers.

While it would be preferable to present results on a public data set such as the HumanEva dataset, a review conducted by the data set's creators [84] concludes that successful tracking of even a simple walking motion is beyond the abilities of present algorithms for one and two camera tracking. This

unfortunately means that the work presented in this thesis cannot be directly compared to existing work in the literature, and highlights the difficulty in performing a quantitative evaluation of the combination of the work presented in this thesis. As a result of these difficulties only a single sequence collected by the author has been used to produce the quantitative results shown below, and so care should be taken when interpreting the results.

7.3 Tracking Results

The following optimization methods used within the CSS [90] tracking framework are analysed in this section:

- Damped Newton optimization in Rotation space (DN in XR). This is the default method used by the CSS algorithm.
- Damped Newton optimization in the Cartesian space presented in Chapter 5 (DN in XC).
- Camera Geometry optimization in the Cartesian space presented in Chapter 6 (Monocular and Cam. Geo. for the one and two camera cases).

The camera geometry method is the only method where results are provided for the one camera tracking case, as it was the only method that produced results successful enough to be compared to the two camera methods. Results are also provided for monocular method where errors in the "ambiguous direction" (the direction from the camera to the link's center) have been removed from the error calculation, which are labelled Monocular*. This is done to provide a fairer comparison of the errors between the one and two camera results.

Table 7.1 provides a summary of the tracking errors for the location of the person's hips, head, hands, and feet, as well as the total tracking error across all locations. This shows that reformulating the state space to Cartesian coordinates, rather than the standard space of joint angles as presented in Chapter 5 of this thesis, improves tracker performance. The most successful optimization scheme was the camera geometry method proposed in Chapter 6, which is to be expected as the optimizer uses information specific to the visual tracking problem. Figure 7.1 shows the limb positional errors for each frame in the data sequences.

The 'mpf' row of Table 7.1 indicates the number of minutes required to process a single frame of the sequence, with the experiments performed in the MATLAB®environment, on a 64 bit 2.5Ghz Intel Xeon machine running Microsoft Windows Enterprise edition, with a variable number of users

	GT (std)	Monocular	Monocular*	DN in XR	DN in XC	Cam. Geo
frames	43	42	42	43	43	43
mpf		1.34	1.34	4.87	7.35	2.12
hip	31.87	47.21 ±28.42	18.78 ±9.46	22.64 ±11.21	19.72 ±5.87	19.74 ±8.18
head	22.66	85.89 ±83.60	17.94 ±30.25	17.34 ±12.23	10.81 ±4.46	10.89 ±4.20
left hand	136.12	126.53 ±160.13	84.22 ±154.70	66.46 ±63.74	51.48 ±42.29	54.43 ±44.04
right hand	121.15	226.90 ±152.69	49.31 ±57.35	45.80 ±41.82	50.35 ±43.72	45.31 ±34.45
left foot	19.43	189.20 ±191.29	61.40 ±61.15	37.28 ±44.26	31.13 ±21.60	22.59 ±18.37
right foot	25.83	138.90 ±122.46	58.20 ±117.28	38.29 ±55.89	25.20 ±11.85	25.29 ±16.10
all		**135.77 ±145.98**	**48.31 ±89.61**	**37.97 ±45.63**	**31.45 ±30.67**	**29.71 ±29.20**

Table 7.1: Tracking Errors (mean mm ± std)

logged on during the experiments. As such these timings should be viewed as indicative and are perhaps being viewed relatively rather than absolutely.

The following subsections provide a discussion of the performance of each optimization method, as well as images from the tracking results.

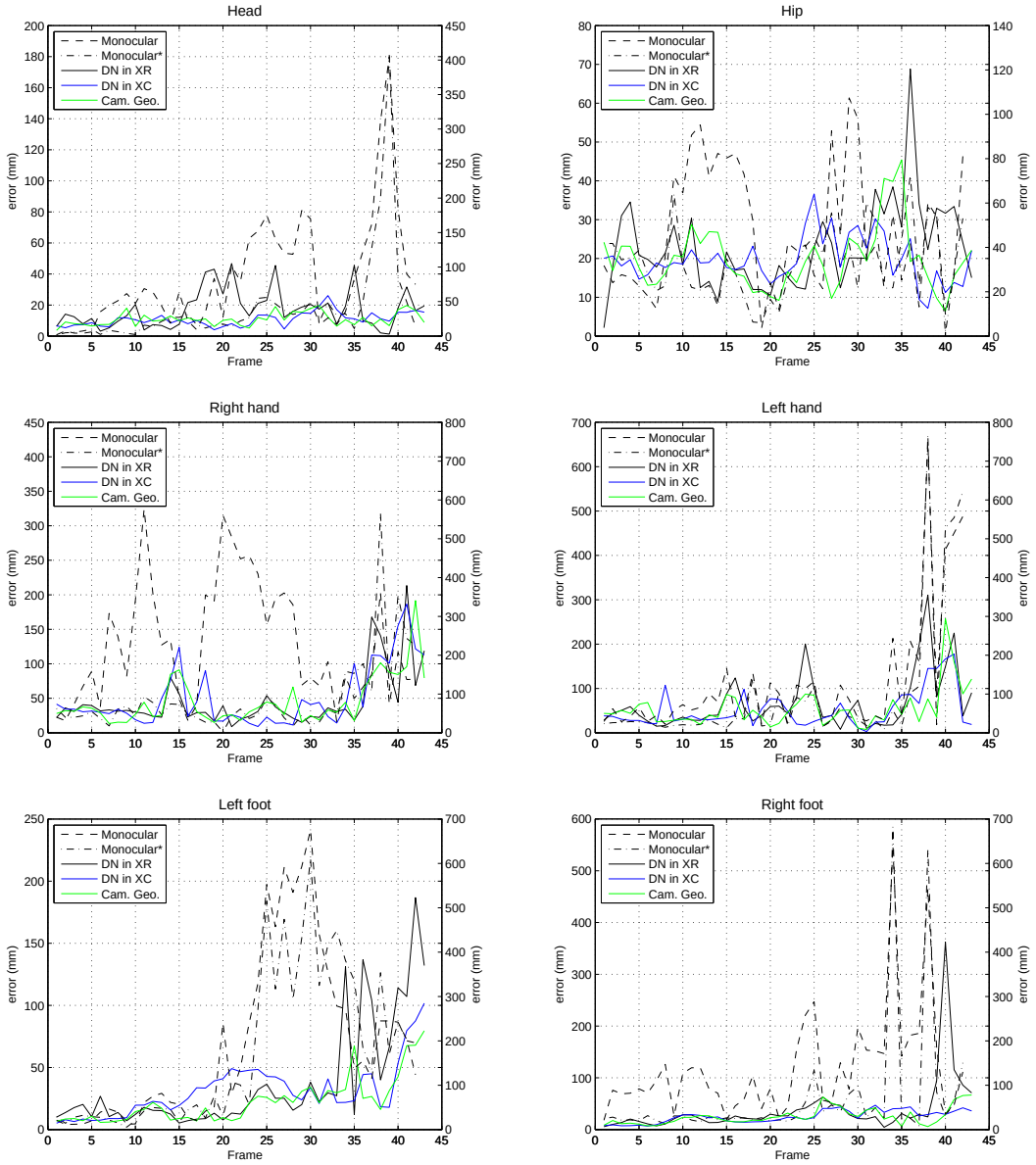
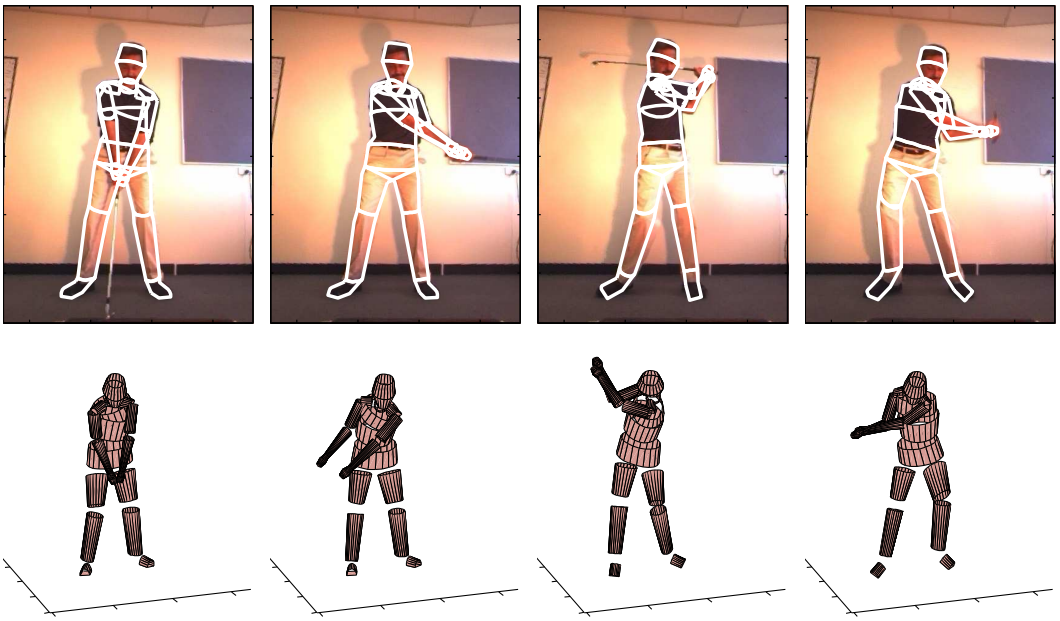


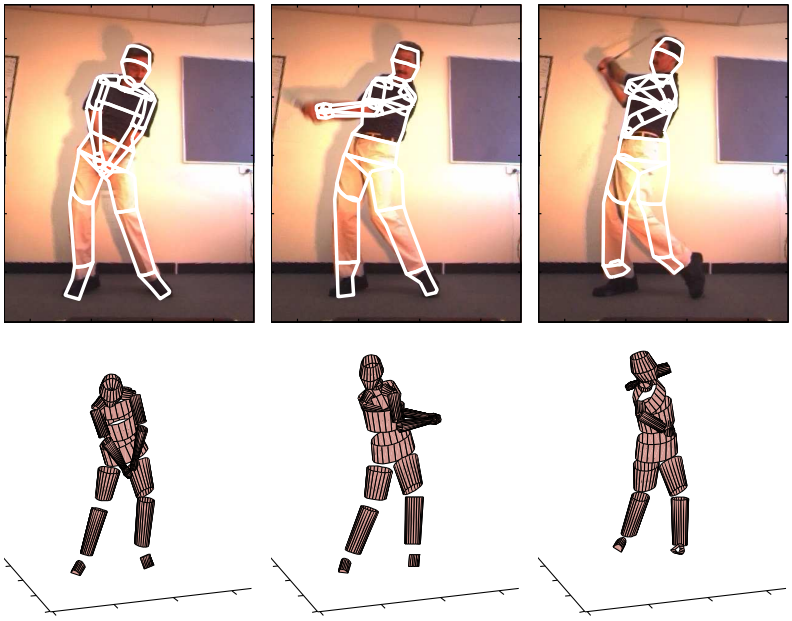
FIGURE 7.1: Limb Tracking Errors

7.3.1 Monocular Tracking

Figure 7.2 shows the image sequence being successfully (mostly) tracked using the camera geometry optimization method presented in Chapter 6. The maximum likelihood estimate is plotted in each frame, and a 3D representation of the golfer's posture is also shown. Note that the image sequence is mirrored, so that a screen in front of the golfer acts like a mirror, which is why the 3D representation appears to move in the opposite direction. The first 37 frames were tracked quite successfully, however there is a dramatic tracking failure by frame 39. This was largely due to incorrectly assigning which elbow was closer to the camera when the arms occluded each other. Propagating an increased number of states between frames could potentially solve this problem. An alternative approach would



(a) Frame 1 (b) Frame 12 (c) Frame 25 (d) Frame 30



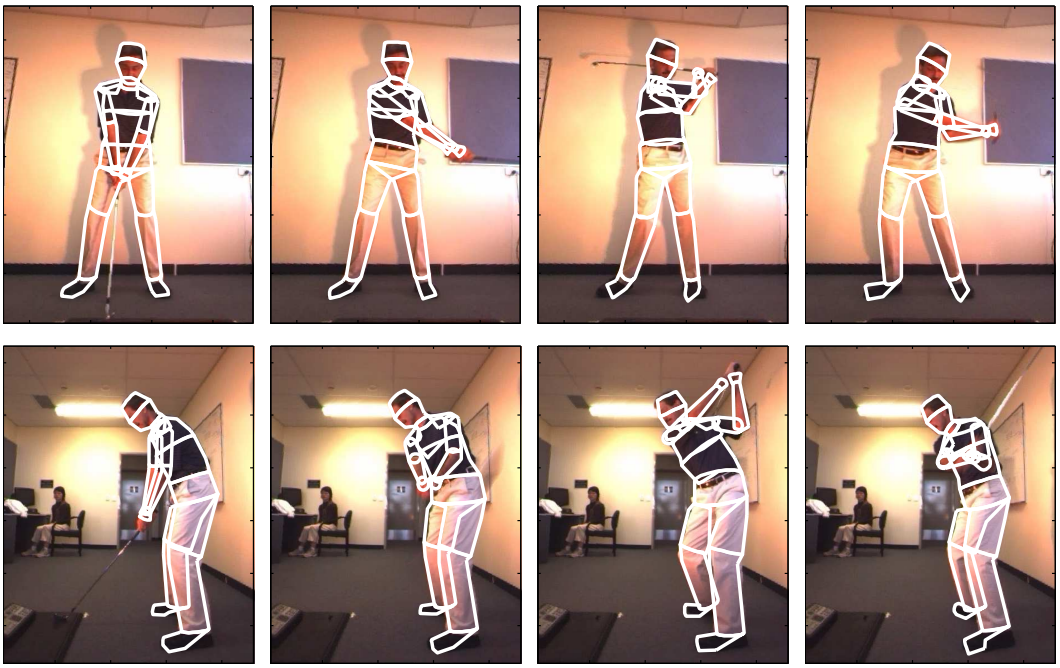
(e) Frame 33 (f) Frame 36 (g) Frame 39

FIGURE 7.2: Monocular tracking

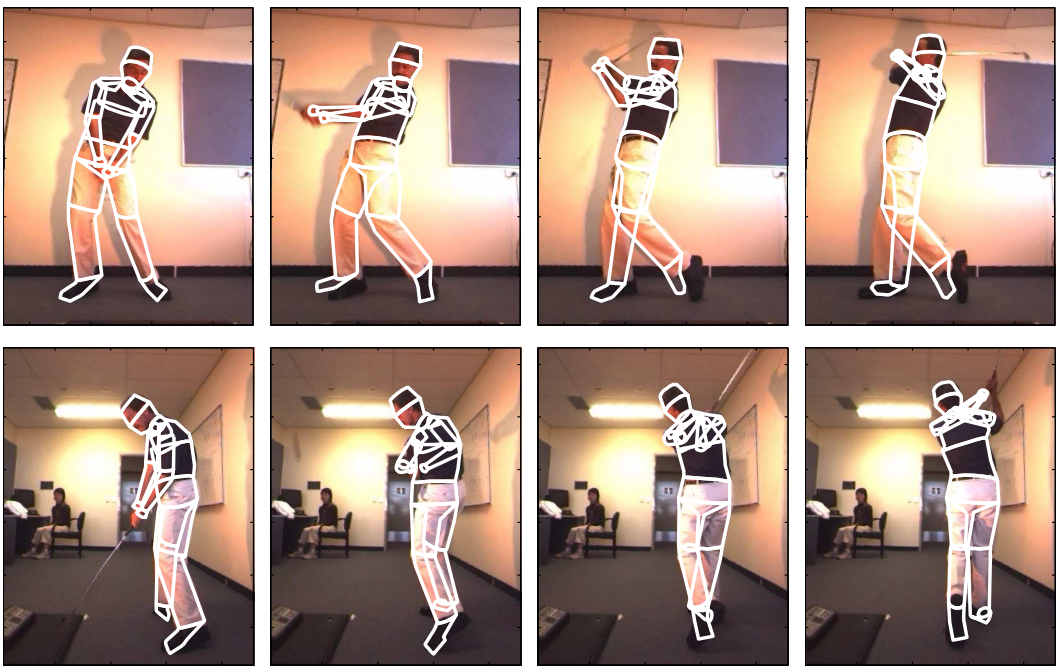
be to further improve the camera geometry optimization method, by allowing local minimas to be escaped using the known camera location to check states where the occlusion order is reversed.

7.3.2 Newton Optimizer in Rotation Space (two cameras)

Figure 7.3 shows the results for the tracker using the Newton optimizer in the rotation space - the approach presented by Sminchisescu and Triggs for the CSS algorithm [90]. This figure shows the tracker failing to maintain lock as the arms become occluded by the body in the second view, while occluding each other from the first view.



(a) Frame 1 (b) Frame 12 (c) Frame 25 (d) Frame 30

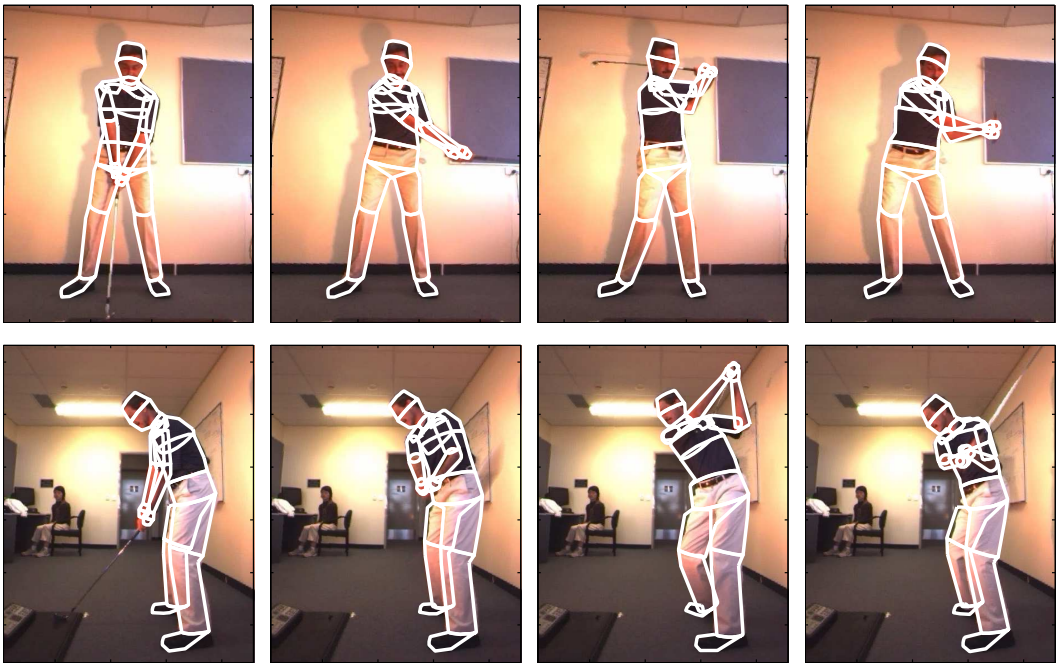


(e) Frame 33 (f) Frame 36 (g) Frame 39 (h) Frame 43

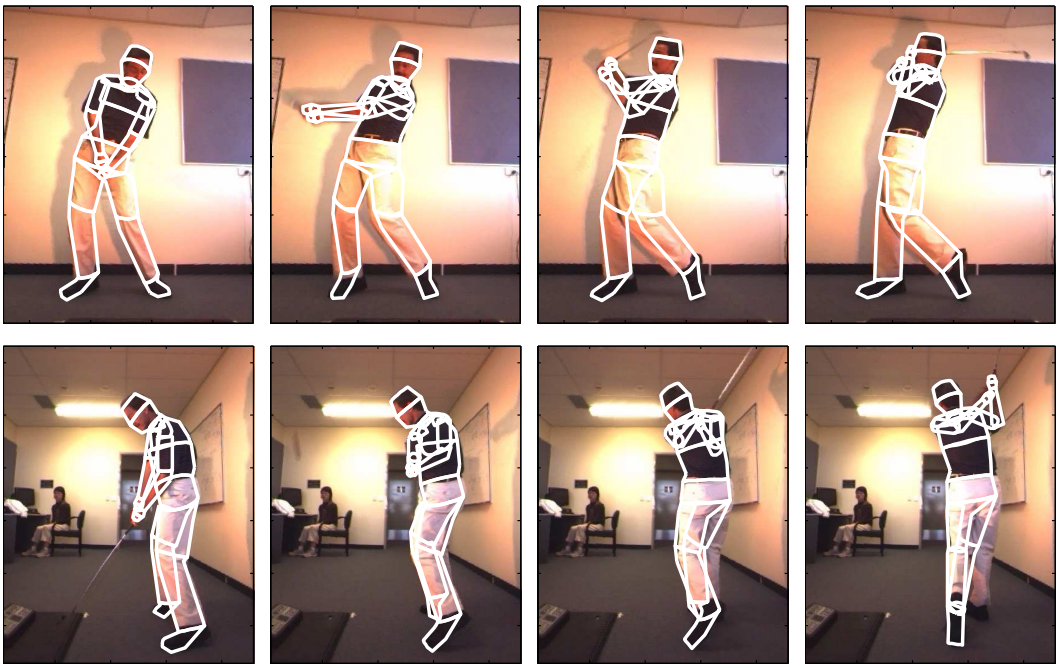
FIGURE 7.3: Two camera tracking - Newton optimizer in rotation space

7.3.3 Newton Optimizer in Cartesian Space (two cameras)

Figure 7.4 shows the results for the tracker using the Newton optimizer in Cartesian space, as discussed in Chapter 5. This figure shows that tracking is mostly successful, however the lock on the right leg is lost in the later frames. This loss of lock of the leg stemmed from the ambiguity in the direction of the vertical axis rotation of the hips, wither the tracker failing to propagate the correct direction of rotation.



(a) Frame 1 (b) Frame 12 (c) Frame 25 (d) Frame 30

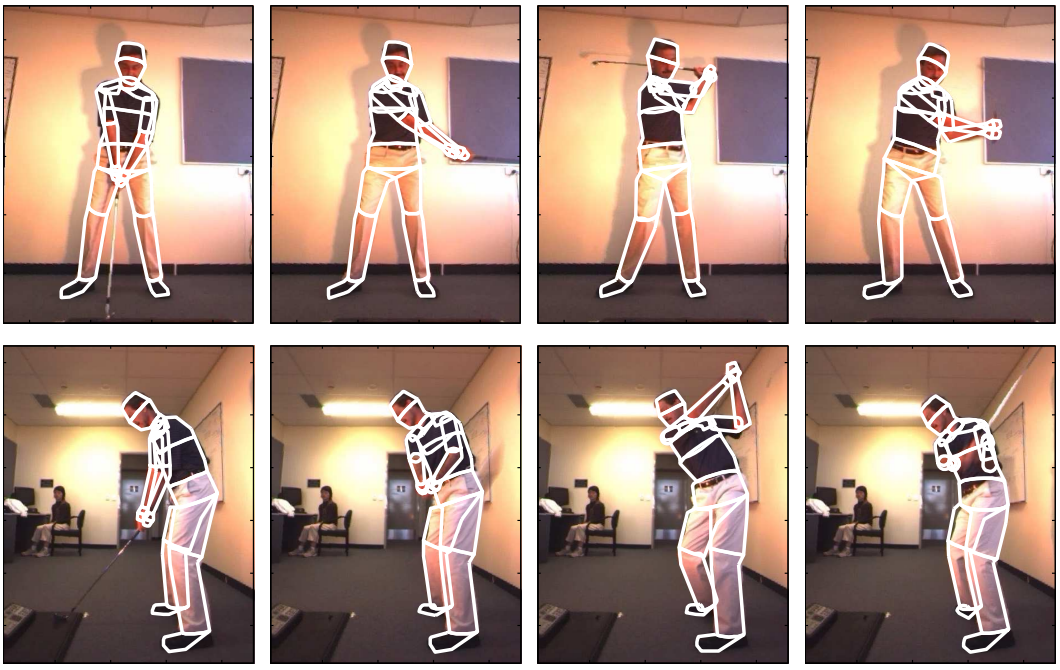


(e) Frame 33 (f) Frame 36 (g) Frame 39 (h) Frame 43

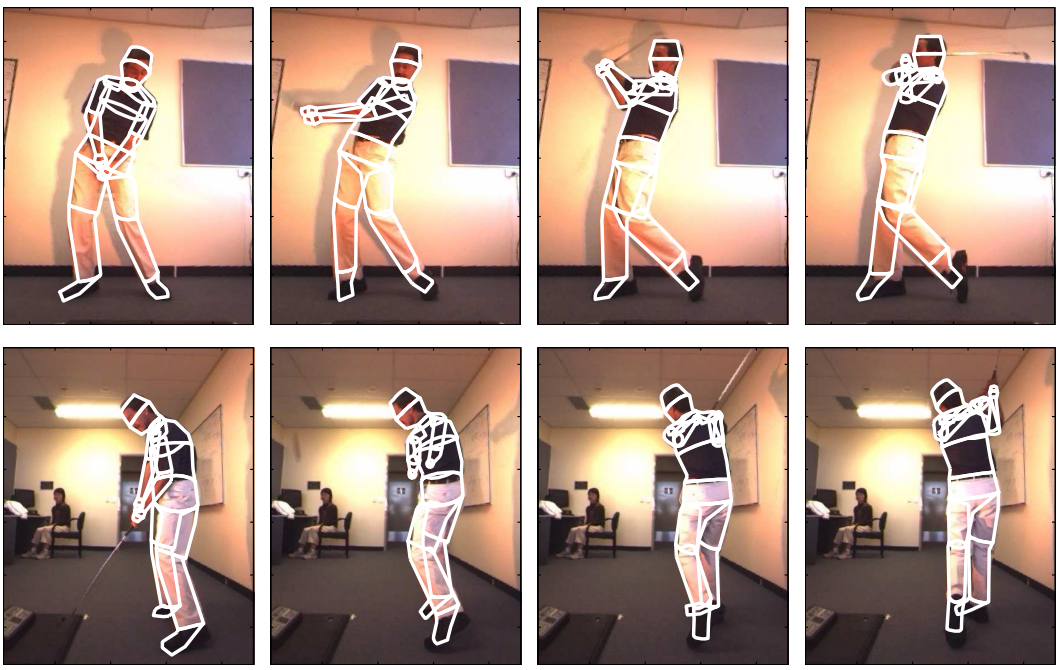
FIGURE 7.4: Two camera tracking - Newton optimizer in cartesian space

7.3.4 Camera Geometry Optimizer in Cartesian Space (two cameras)

Figure 7.5 shows the image sequence being successfully tracked using camera geometry optimization method presented in Chapter 6. All tracker settings were identical to those used in Section 7.3.3, with the exception that the camera geometry optimizer was used to perform the local optimizations. The additional observational information provided by the second camera allowed this method to track successfully, while the monocular version lost lock around frame 37.



(a) Frame 1 (b) Frame 12 (c) Frame 25 (d) Frame 30



(e) Frame 33 (f) Frame 36 (g) Frame 39 (h) Frame 43

FIGURE 7.5: Two camera tracking - camera geometry optimization

There are some inaccuracies in these tracking results however, such as the position of the hands in frame 30. The position of the front foot also slips from the true position before frame 43. This was the result of the measurement function judging the plotted state as more likely than the true foot location, which was discussed earlier in Section 4.11.

8

Conclusions and Future Work

This thesis has proposed modifications to the 'building blocks' of visual tracking algorithms, which were individually shown to improve the performance of each specific module. In Chapter 7 it was shown that the combination of these modifications significantly improved the performance of a covariance scaled sampling styler multiple hypothesis tracker.

8.1 Conclusions

Edge Measurement Function In Chapter 4 a novel edge measurement scheme was described, utilizing a graph based approach to edge measurement. Unlike other methods, this graph based method enforces a consistent edge measurement along each object edge, due to the edge connectedness of the graph. This method is not reliant upon finding a set of edge features for each image, but rather operates directly on a smooth function of the image derivatives. This method was extended so that a single graph was used for each occluding contour of the object.

Unlike in the usual case of calculating the shortest path across a graph, self occlusions mean that the weights of some graph vertices are unknown. Several treatments were presented and evaluated, and methods for the computation of the optimal shortest path for each treatment were also given. It

was determined that the biased occlusions method was the most successful. The biased occlusion method was built on the intuition that the object's surface models are not particularly accurate, and so biasing the shortest path towards possible occlusions limits the variability of the measurement function in these occluded areas.

It is shown that the proposed graph based edge measurement approach produces observational likelihoods with significantly fewer modes than the edge measurement schemes previously used in tracking problems. The worst performance in the two dimensional experiments was a 95% reduction in the number of modes. While this approach is computationally more expensive, it is argued that this is offset by the reduced computational expense of the global search procedure.

State space selection In Chapter 5 the choice of state space in which the tracking problem is formulated was examined. A state spaced comprising primarily of the Cartesian coordinates of links in the kinematic chain was proposed as an alternative to the state space based on the joint angles of the human body as used by other authors. This state space was proposed because of the much more linear relationship between state variables in this space and the 3D locations of points on the object model. While similar state spaces have been used for representations of full body kinematic chains, to the author's knowledge this Cartesian based formulation has not been used in conjunction with a 'pure' kinematic model.

This Cartesian based space has greater dimensionality than the rotation based space, and perhaps has been avoided due to the well known "curse of dimensionality". The region of the state space where the object is in a geometrically consistent state (*i.e.*, limb lengths are preserved) has the same content as the original rotation space however. A simple method is given to project an implausible state onto the constraint surface, as well as a measurement function curvature dampening scheme, which is used to restrict variability in directions orthogonal to the constraint surface.

It was shown that the more linear relationship between state variables and the location of points on the object model gave increased dynamic predication accuracy for a range of different dynamic models, and also reduced the number of dimensions required to capture the motion in a reduced state space. Furthermore, the increased linearity also caused means the location of a point on the object in the image is also a more linear function of the state variables. It was shown that this increased local optimizer performance, particularly for optimization methods which use localized quadratic approximations to the measurement function.

Local Optimization Method In Chapter 6, a novel local optimization method was proposed. Newton like local optimizations are often used as a component of global search strategies as they good

general methods. However, as discussed earlier in this thesis, the 1^{st} and 2^{nd} partial derivatives of the measurement function are not defined for all (interesting) points in the state space. The proposed local optimization method is specifically tailored to the visual tracking problem.

The proposed local optimizer used camera geometry to infer interesting search directions, rather than by inferring search directions directly from noisy image data. Variations for choosing these search direction are given for the monocular and two camera tracking cases. In the monocular case, the search directions of interest are chosen as the ambiguous depth direction and the image displacement maximizing direction, while in the two camera case search directions were chosen to only alter the object's appearance in one image. The proposed method also uses a problem decomposition to reduce the computational cost of the optimizer.

The proposed method is shown to outperform Newton based optimizations in a rotation based state space, and gives at worst equivalent results to a Newton optimizer in a Cartesian state space, but at a significantly reduced computational cost. Extensions are given to allow the optimizer to escape local modes cause by mismatched edge assignments, and kinematic forwards–backwards ambiguities.

8.1.1 Original Contributions

In the author's opinion, the significant contributions of this thesis are:

- A graph based edge measurement approach for tracking problems.
- The formulation of a method to calculate a consistent edge measurement around an entire occluding contour.
- Treatments for calculating the shortest path across a graph with occluded vertices.
- The formulation of the tracking problem in a Cartesian based state space, including methods to project implausible configurations onto the constraint surface and dampening curvature in implausible directions.
- Demonstrated improved dynamic model prediction using the Cartesian space.
- Demonstrated improved local (Newton) optimization performance in the Cartesian space.
- The formulation of a novel local optimization method using camera geometry to select interesting search directions.
- Applied extensions to the proposed local optimizer to escape local modes commonly 'fallen' into.

8.2 Future Work

This thesis concludes with a brief discussion for avenues of future research in this area. Despite the successful tracking results shown in Chapter 7, there remains several avenues in which further improvements to the 'building blocks' are possible.

The graph formulated around an occluding contour of the object in Chapter 4 uses a strict rectangular trellis formulation. This is not ideal as it was noted that the number of interior rows must be kept small so that consecutive link trellises may be joined to form the occluding contour's trellis. In some states this joining process can not be performed, which may cause local modes in the observational likelihood. Furthermore, a trellis row may have significantly different pixel spacings between vertices in different regions. This may lead to a bias towards states which have a high concentration of vertices on strong edges, and a low concentrations along weaker edges. Using more general graphs formed around the occluding contour could alleviate these problems. As a starting point, varying the number of rows used in a local section of the occluding contour's trellis could help eliminate cases where consecutive link trellises can not be joined. The use of oriented metrics to determine the vertex weights should also be considered, where the only component of the image derivatives orthogonal to the hypothesized edge direction is considered.

In the comparison of state spaces in Chapter 5, the Newton based optimizers did not use (measurement function level) approximations to the Jacobian and Hessian of the measurement function as used by some other authors. An evaluation of the optimizers' performances when these are used should be performed. An analytic approximation to the position of a point on the object in the image as a function of the state variables could also be considered. The synthetic annealing experiment in Chapter 5 also suggested that sampling in the Cartesian space generated more interesting samples than sampling in the rotation space. The extent to which this is true should also be investigated with further experimentation.

The proposed local optimizer presented in Chapter 6 assumes each link in the kinematic chain being able to be optimized independently. It is believed that the proposed edge measurement scheme greatly strengthened this assumption. In some cases however, trial points are generated which improve the 'fit' of the current link, however reduce the fit of the overall model. In these case a 'depth' search could be performed recursively, where the state is used to seed the optimizer acting only upon connected link's in the kinematic chain. This may give the kinematic ambiguity mode breaking search the ability to act on a local region of the kinematic chain. It may also give the optimizer the ability to resolve mismatched edge assignments between two links, so for example the left and right hand positions may be 'swapped'.

References

[1] "EyeToy®," [Online] Available at http://www.eyetoy.com/index.asp, last accessed 15/2/2007.

[2] "Dragon Fly at Point Grey Research," [Online] Available at http://www.ptgrey.com/products/dragonfly/dragonfly.pdf, last accessed 25/8/2004.

[3] I. Andricioaei, J. E. Straub, and A. F. Voter, "Smart Darting Monte Carlo," *Journal of Chemical Physics*, vol. 114, no. 16, pp. 6994–7000, 2001.

[4] B. Appleton and C. Sun, "Circular Shortest Paths by Branch and Bound," *Pattern Recognition*, vol. 36, no. 11, pp. 2513–2520, 2003.

[5] B. Appleton and H. Talbot, "Globally Optimal Surfaces by Continous Maximal Flows," in *Digital Image Computing: Techniques and Applications*, vol. 2, 2003, pp. 987–996.

[6] B. Appleton and H. Talbot, "Globally optimal geodesic active contours," *Journal of Mathematical Imaging and Vision*, vol. 23, no. 1, pp. 67–86, 2005.

[7] B. Appleton and H. Talbot, "Globally Minimal Surfaces by Continuous Maximal Flows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 1, pp. 106–118, 2006.

[8] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for Online Non-linear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[9] A. Azarbayejani, T. Darrell, A. P. Pentland, and C. Wern, "Pfinder: Real-time tracking of the human body," in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 19(7), 1997, pp. 780–785.

[10] A. Balan, L. Sigal, and M. Black, "A quantitative evaluation of video-based 3d person tracking," in *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005, pp. 349–356.

[11] P. Bamford and B. Lovell, "Unsupervised cell nucleus segmentation with active contours," in *Signal Processing (Special Issue: Deformable models and techniques for image and signal processing)*, vol. 71, no. 2, 1998, pp. 203–213.

[12] M. Black and P. Anandan, "The Robust Estimation of Multiple Motions: Parametric and Piecewise Smooth Flow Fields," *Computer Vision and Image Understanding*, vol. 6, no. 1, pp. 57–92, 1996.

[13] A. Blake and M. Isard, *Active Contours*. London: Springer-Verlag, 1998.

[14] Y. Boykov and V. Kolmogorov, "Computing geodesics and minimal surfaces via graph cuts," in *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, vol. 1. Washington, DC, USA: IEEE Computer Society, 2003, pp. 26 – 33.

[15] H. Bozdogan, "Model Selection and Akaikes Information Criterion (AIC): The General Theory and its Analytical Extensions," in *Psychometrica*, vol. 52(3), 1987, pp. 345–370.

[16] M. Bray, E. Koller-Meier, N. N. Schraudolph, and L. Van Gool, "Stochastic Meta-Descent for Tracking Articulated Structures," *IEEE Workshop on Articulated and Nonrigid Motion, Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004.

[17] M. Bray, E. Koller-Meier, N. N. Schraudolph, and L. Van Gool, "Fast Stochastic Optimization for Articulated Structure Tracking," *Image and Vision Computing*, vol. 25, no. 3, pp. 352–364, 2007.

[18] C. Bregler and J. Malik, "Tracking People with Twists and Expodential Maps," in *IEEE International Connference of Computer Vision and Pattern Recognition*, 1998, pp. 8–15.

[19] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic Active Contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1997.

[20] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert, "Minimal Surfaces Based Object Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 394–398, 1997.

[21] T. Cham and J. M. Rehg, "A Multiple Hypothesis Approach to Figure Tracking," in *IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 2, 1999, pp. 239–245.

[22] B. V. Cherkassky, A. Goldberg, and T. Radzik, "Shortest path algorithms: theory and experimental evaluation," *Math Programming*, vol. 73(2), pp. 129–174, 1996.

[23] G. K. M. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler, "A real time system for robust 3D voxel reconstruction of human motions," in *Proceedings of Computer Vision and Pattern Recognition Conference*, vol. 2, 2000, pp. 714–720.

[24] K. Choo and D. Fleet, "People Tracking Using Hybrid Monte Carlo Filtering," *IEEE International Conference on Computer Vision*, vol. 2, pp. 321–328, 2001.

[25] L. Cooper and D. SteinBerg, *Introduction to Methods of Optimization*. W. B. Saunders Company, 1970.

[26] S. Corazza, E. Gambaretto, L. Mundermann, and T. P. Andriacchi, "Automatic generation of a subject-specific model for accurate markerless motion capture and biomechanical applications." *IEEE Trans. Biomed. Engineering*, vol. 57, no. 4, pp. 806–812, 2010.

[27] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley-Interscience, 1991.

[28] G. Cross and A. Zisserman, "Quadric Reconstruction from Dual-Space Geometry," in *Proceedings of the Sixth International Conference on Computer Vision*, 1998, pp. 25–31.

[29] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, ser. CVPR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 886–893. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2005.177

[30] J. Davis and A. Bobick, *Virtual PAT: a virtual personal aerobics trainer*, 436th ed., MIT Media Lab Perceptual Computing Group Technical Report, 1998.

[31] J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *Proceedings of Computer Vision and Pattern Recognition Conference*, vol. 2, 2000, pp. 126–133.

[32] J. Deutscher, A. Davison, and I. Reid, "Automatic Partitioning of High Dimensional Search Spaces associated with Articulated Body Motion Capture," in *IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 2, 2001, pp. 669–676.

[33] J. Deutscher and I. Reid, "Articulated Body Motion Capture by Stochastic Search," *International Journal of Computer Vision*, vol. 61, no. 2, pp. 185–205, 2004.

[34] C. Dyer, *Foundations of Image Understanding*. Kluwer, 2001, eds. L.S. Davis.

[35] A. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric Model for Background Subtraction," in *Proceedings of the Sixth European Conference on Computer Vision*, vol. 1843, 2000, pp. 751 – 767.

[36] F. Evans, D. Alder, and J. deSilva, "Determining the Number of Clusters in a Mixture by Iterative Model Space Refinement - with Application to Free-swimming Fish Detection," in *Digital Image Computing: Techniques and Applications*, 2003, pp. 79–88.

[37] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient matching of pictorial structures," in *Proceedings of Computer Vision and Pattern Recognition*, 2000, pp. 66–73.

[38] R. Fletcher, *Practical Methods of Optimization*, 3rd ed. John Wiley and Sons, 1987.

[39] D. M. Gavrila and L. S. Davis, "3-D model based tracking of humans in action: a multi-view approach," in *IEEE Computer Vision and Pattern Recognition*, 1996, pp. 73–80.

[40] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, *Markov Chain Monte Carlo in practice*. Chapman and Hall, 1996.

[41] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky, "Fast Geodesic Active Contours," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1467–75, 2001.

[42] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.

[43] R. D. Green and L. Guan, "Tracking human movement patterns using particle filtering," in *Acoustics, Speech, and Signal Processing*, vol. 3, 2003, pp. 25–28.

[44] F. Guo and G. Qian, "Learning and Inference of 3D Human Poses from Gaussian Mixture Modeled Silhouettes," in *Proceedings of the Eighteenth International Conference on Pattern Recognition*, vol. 2, 2006, pp. 43–47.

[45] R. Hartley and A. Zisserman, *Multiple View Geometry*. Cambridge: Cambridge University Press, 2000.

[46] T. Heap and D. Hogg, "Wormholes in Shape Space: Tracking Through Discontinuous Changes in Shape," in *Proceedings of the Sixth IEEE International Conference on Computer Vision and Pattern Recognition*, 1998, pp. 344–349.

[47] Y. C. Ho and R. C. K. Lee, "A Bayesian Approach to Problems in Stochastic Estimation and Control," in *IEEE Transactions on Automation and Control*, vol. 9, 1964, pp. 333–339.

[48] M. Isard and A. Blake, "Visual tracking by stochastic propagation of conditional density," in *Proceedings of the Forth European Conference on Computer Vision*, 1996, pp. 343–356.

[49] M. Isard and A. Blake, "ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework," *Lecture Notes in Computer Science*, vol. 1406, pp. 893–908, 1998.

[50] J. E. Jackson, *A User's Guide to Principal Components*. Wiley and Sons, Inc., 1991.

[51] H. T. Jongen, K. Meer, and E. Triesch, *Optimization Theory*. Kluwer Academic Publishers, 2004.

[52] S. X. Ju, M. J. Black, and Y. Yacoob, "Cardboard people: A parameterized model of articulated image motion," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, 1996, pp. 38–44.

[53] S. J. Julier and J. K. Uhlmann, "Unscented Filtering and Nonlinear Estimation," in *Proceedings of the IEEE*, vol. 92(3), 2004, pp. 401–422.

[54] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A New Approach for Filtering Nonlinear Systems," in *Proceedings of the American Control Conference*, 1995, pp. 1628–1632.

[55] P. KaewTraKulPong and R. Bowden, "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow detection," in *Proceedings of the Second European Workshop on Advanced Video Based Surveillance Systems*, 2001.

[56] E. R. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[57] R. Kehl, M. Bray, and L. V. Gool, "Full body tracking from multiple views using stochastic sampling," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 129–136.

[58] O. King and D. A. Forsyth, "How Does CONDENSATION Behave with a Finite Number of Samples?" in *Proceedings of the Sixth European Conference on Computer Vision*, vol. 1842, 2000, pp. 695–709.

[59] S. Kirkpatrick, C. D. Gellatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[60] A. Land and A. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, pp. 497–520, 1960.

[61] V. Lepetit, A. Shahrokni, and P. Fua, "Robust Data Association For Online Applications," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2003, pp. 281–288.

[62] J. P. Luck, C. Debrunner, Q. H. W. Hoff, and D. E. Small, "Development and analysis of a real-time human motion tracking system," in *WACV*, 2002, pp. 196–202.

[63] J. P. Luck, W. Hoff, D. Small, and C. Little., "Real-Time Markerless Human Motion Tracking using Linked Kinematic Chains," in *WACV*, 2002, pp. 849–854.

[64] J. MacCormick, *Stochastic Algorithms for Visual Tracking*. London: Springer-Verlag, 2002.

[65] J. MacCormick and A. Blake, "A probabilistic contour discriminant for object localisation," in *Proceedings of the Sixth International Conference on Computer Vision*, 1998, pp. 390–395.

[66] J. MacCormick and A. Blake, "Partitioned sampling, articulated objects and interface quality hand tracking," in *Proceedings of the Sixth European Conference on Computer Vision*, vol. 2, 2000, pp. 3–19.

[67] W. G. Macready and D. H. Wolpert, "What Makes An Optimization Problem Hard?" Santa Fe Institute, Santa Fe, NM, Tech. Rep. SFI-TR-95-05-046, 1995.

[68] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London Series B*, vol. 207, no. 1167, pp. 187–217, 1980.

[69] R. Merve, A. Doucet, N. Freitas, and E. Wan, "The Unscented Particle Filter," Cambridge University, Tech. Rep., 2000, technical Report CUED/F-INFENG/TR380.

[70] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman, "Human Body Model Acquisition and Tracking Using Voxel Data," *International Journal of Computer Vision*, vol. 53(3), pp. 199–233, 2003.

[71] H. Moon and R. Chellappa, "3d shape-encoded particle filter for object tracking and its application to human body tracking." *EURASIP J. Image and Video Processing*, vol. 2008, 2008. [Online]. Available: http://dblp.uni-trier.de/db/journals/ejivp/ejivp2008.html

[72] D. Morris and J. M. Rehg, "Singularity analysis for articulated object tracking," in *Computer Vision and Pattern Recognition*, 1998, pp. 289–296.

[73] R. M. Neal, "Annealed importance sampling," *Statistics and Computing*, vol. 11, no. 2, pp. 125–139, 2001.

[74] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd ed. New York: McGraw-Hill, 1984.

[75] M. Pitt and N. Shephard, "Filtering via Simulation: Auxiliary Particle Filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–630, 1997.

[76] D. Ramanan, D. Forsyth, and A. Zisserman, "Strike a pose: Tracking people by finding stylized poses," in *Proceedings of Computer Vision and Pattern Recognition Conference*, 2005, pp. I: 271–278.

[77] R. Rosales and S. Sclaroff, "Learning body pose via specialized maps," in *Neural Information Processing Systems*, 2002.

[78] Roy Leipnik, "The Extended Entropy Uncertainty Principle," *Information and Control*, vol. 3, no. 1, pp. 18–25, 1960.

[79] G. A. F. Seber, *Multivariate Observations*. Wiley, 1984.

[80] H. Senderowitz, F. Guarnieri, and W. C. Still, "A Smart Monte Carlo Technique for Free Energy Simulations of Multiconformal Molecules. Direct Calculation of the Conformational Population of Organic Molecules," *Journal of the American Chemical Society*, vol. 117, no. 31, pp. 8211–8219, 1995.

[81] Y. Sheikh, A. Datta, and T. Kanade, "On the sustained tracking of human motion." in *Proceedings of the 8th IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE Computer Society, 2008, pp. 1–7.

[82] H. Sidenbladh, M. J. Black, and D. J. Fleet, "Stochastic Tracking of 3D Human Figures Using 2D Image Motion," *Proceedings of the Sixth European Conference on Computer Vision*, vol. 2, pp. 702–718, 2000.

[83] L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard, "Tracking Loose-limbed People," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. 421–428.

[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," *Int. J. Comput. Vision*, vol. 87, no. 1-2, pp. 4–27, Mar. 2010. [Online]. Available: http://dx.doi.org/10.1007/s11263-009-0273-6

[85] D. E. Small, "Real Time Shape from Silhouette," Masters Thesis, Colorado Schools of Mines, University of Maryland, 2001.

[86] C. Sminchiesescu, "ESTIMATION ALGORITHMS FOR AMBIGUOUS VISUAL MODELS - Three Dimensional Human Modeling and Motion Reconstruction in Monocular Video Sequences," Ph.D. dissertation, Institute National Politechnique de Grenoble (INRIA), 2002.

[87] C. Sminchisescu, "Consistency and Coupling in Human Model Likelihoods," in *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, vol. 1, 2003, pp. 69–76.

[88] C. Sminchisescu, A. Kanaujia, L. Zhiguo, and D. Metaxas, "Discriminative Density Propagation for 3D Human Motion Estimation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 390–397.

[89] C. Sminchisescu and B. Triggs, "Building Roadmaps of Local Minima of Visual Modes," in *Proceedings of the Seventh European Conference on Computer Vision*, vol. 1, 2002, pp. 566–582.

[90] C. Sminchisescu and B. Triggs, "Estimating Articulated Human Motion with Covariance Scaled Sampling," *International Journal of Robotics Research*, vol. 22(6), pp. 371–393, 2003.

[91] C. Sminchisescu and B. Triggs, "Kinematic Jump Process for Monocular 3D Human Tracking," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 1, 2003, pp. 69–76.

[92] C. Sminchisescu and B. Triggs, "Fast Mixing Hyperdynamic Sampling," *Image and Vision Computing, Special Issue on Outstanding Papers from ECCV 2002 Conference*, vol. 24, no. 3, pp. 279–289, 2005.

[93] C. Sminchisescu and B. Triggs, "Mapping Minima and Transitions in Visual Models," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 81–101, 2005.

[94] C. Sminchisescu, M. Welling, and G. Hinton, "A Mode-Hopping MCMC Sampler," University of Toronto, Tech. Rep. Technical Report CSRG-478, 2003.

[95] A. W. B. Smith and B. C. Lovell, "Measurement Function Design for Visual Tracking Applications," in *Proceedings of the Eighteenth International Conference on Pattern Recognition*, vol. 1, 2006, pp. 789–792.

[96] A. Sundaresan and R. Chellappa, "Model driven segmentation of articulating humans in laplacian eigenspace," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1771–1785, 2008.

[97] A. Sundaresan and R. Chellappa, "Multi-camera tracking of articulated human motion using motion and shape cues," *IEEE Transactions on Image Processing*, vol. 18, no. 9, pp. 2114–2126, 2009.

[98] R. Szeliski and D. Terzopoulos, "Tracking with kalman snakes," in *Active Vision*, 1992, pp. 3–20.

[99] R. Urtasun, D. J. Fleet, and P. Fua, "Monocular 3-D Tracking of the Golf Swing," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 932–938.

[100] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.

[101] A. F. Voter, "A Method for Accelerating the Molecular Dynamics Simulation of Infrequent Events," *Journal of Chemical Physics*, vol. 106, no. 11, pp. 4665–4677, 1997.

[102] A. F. Voter, "Hyperdynamics: Accelerated Molecular Dynamics of Infrequent Events," *Physical Review Letters*, vol. 78, no. 20, pp. 3908–3911, 1997.

[103] S. Wachter and H. Nagel, "Tracking Persons in Monocular Image Sequences," in *Computer Vision and Image Understanding*, vol. 74(3), 1999, pp. 174–192.

[104] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Department of Computer Science, University of North Carolina, Tech. Rep., 1995.

[105] P. H. Winston, *Artificial Intelligence*, 3rd ed. Addison–Wesley Publishing Company, 1992, ch. Nets and Optimal Search, pp. 81–100.

[106] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Search," Santa Fe Institute, Santa Fe, NM, Tech. Rep. SFI-TR-95-02-010, 1995.

[107] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.

[108] Y. Yacob and L. Davis, "Learned Temporal Models of Image Motion," in *Sixth International Conference on Computer Vision*, 1998, pp. 446–453.

[109] J. H. Zar, *Biostatistical Analysis*, 4th ed. Prentice Hall, 1999.

[110] X. Zhao and Y. Liu, "Generative tracking of 3D human motion by hierarchical annealed genetic algorithm," *Pattern Recognition*, vol. 41, no. 8, pp. 2470–2483, 2008.

Symbols used throughout this thesis

Symbol	Description
$\star$	The convolution operator
$\cdot$	The dot product of two vectors
$\langle \times \rangle_z$	The z component of the vector cross product, *i.e.* $\langle a \times b \rangle_z = a_x b_y - a_y b_x$
α	The particle survival rate
A	The number of annealing layers
$\mathcal{A}(.)$	Arc length

Table 1: Symbols used throughout this thesis A

Symbol	Description
β_a	Term used to control the smoothness of the weighting function **w**(.)
$c(i,j)$	Vertex weight for vertex $V(i,j)$
C	The number of cameras used in the tracking problem
$\mathcal{C}(n\|s)$	The probability of finding n features on a search line with length s.
d	The dimensionality of the state space
$\mathcal{D}$	Perceptual state space distance
η	Distance, typically along a vector **V**
e	An eigenvalue
$\mathcal{E}$	The edge feature set for a sampled point on the object's boundary
$\mathbf{f}_m(.)$	The measurement process
$\mathbf{f}_s$(.)	The system process
f(.)	An objective function
f_b	Re-sampling weight bias
$\mathcal{G}$	A rotation group
$\mathcal{G}^*$	A clustered rotation group, $\mathcal{G}^* \in \mathcal{G}$
H(.)	The Hessian
h_b	Re-sampling bias strength
J(.)	The Jacobian
λ	The scaling factor for the weighting matrix **W**
L	The length of a projected link edge
$\mathbb{L}_d$	The direction the link runs in
$\mathbb{L}_e$	The Cartesian coordinates of the end of the link
$\mathbb{L}_l$	The link's length
$\mathcal{L}$	A line in $\mathbb{R}^2$ or $\mathbb{R}^3$
$\mathbb{L}_s$	The Cartesian coordinates of the start of the link

Table 2: Symbols used throughout this thesis B

Symbol	Description
μ	The mean value of a distribution
$M(.)$	Cost metric function
n	The number of...
$\mathbf{n}$	A unit vector orthogonal to the object's projected boundary
$\mathbb{N}$	The set of natural numbers
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean μ and covariance Σ
$\mathbf{O}(.)$	Occlusion function
$p(.)$	Probability density function
P	A projection operator
$\mathcal{P}$	The set of all paths across a trellis. Alternatively it is used to describe a plane
$\mathbf{q}_{01}$	The non detection rates associated with the expected feature locations
Q	A queue of nodes
$\mathbf{r}$	The image coordinates of a sampled point on the occluding contour
$\mathbf{r}(.)$	A function to calculate the image coordinates of a sampled point on the occluding contour
$\hat{\mathbf{r}}$	The set of expected feature locations on a measurement line
$\bar{\mathbf{r}}$	The set of pixels evaluated by a measurement function
$\mathbf{R}$	The 3D location of a sampled point on the occluding contour
$\mathbf{R}(.)$	A function to calculate the 3D location of a sampled point on the occluding contour
$\mathbb{R}$	The set of real numbers
σ	A node in a queue Q
Σ	Covariance
S	Source points for shortest path algorithms

Table 3: Symbols used throughout this thesis C

Symbol	Description
$\mathcal{S}$	Denotes a set of particles, or a sphere where appropriate
t	Time dimension
T	Sink points for shortest path algorithms
τ	A tunable algorithm parameter
u	the number of rows in a rectangular trellis
u	An independently identically distributed (i.i.d) noise vector
U(.)	Function controlling the update step for Newtonian optimization
ς	Sign variable, $\varsigma \in \{-1, 1\}$
v	The number of columns in a trellis
v	An independently identically distributed (i.i.d) noise vector
V	The set of graph vertices
$V(i, j)$	Graph vertex (i, j)
V	The eigenvectors of a matrix, or a volumetric hull where appropriate
$\mathcal{V}$	A set of graphs, $V \in \mathcal{V}$
w(.)	A weighting function
W	A symmetric positive definite weighting matrix
x	X axis direction
x	The object's state space, the space of all configurations that the object can take
X	The object's state, a point in the state space **x**. A vector of length d.
y	Y axis direction
$\mathbf{z}^t$	The set of all measurements at time t
$\mathbf{Z}^t$	The set of all measurements up to time t
Z	The set of measurements for the object in state X
$\mathbb{Z}$	The set of integers

Table 4: Symbols used throughout this thesis D