3D Hand Tracking in a Stochastic Approximation Framework

Desmond Chun Fung Chik

June 2009

A thesis submitted for the degree of Doctor of Philosophy
of the Australian National University

THE AUSTRALIAN NATIONAL UNIVERSITY

Dedicated to the interested readers.
Mum and Dad included.

Declaration

I hereby declare that the work in this thesis is my own, or partially in collaboration with others, except where otherwise stated and that, to the best of my knowledge, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning. My contribution to the results of the collaborative work presented in this thesis is at least 80%.

Desmond C.F. Chik

Acknowledgements

First and foremost I would like to thank my main supervisor, Dr Jochen Trumpf, for his constructive support and supervision, for the intellectually stimulating discussions, for challenging me to think differently, and for having faith in me through the most trying of times. I would also like to thank Dr Nicol N. Schraudolph for his valuable advice and making himself available when help was needed. I am grateful to Prof. Luc Van Gool for hosting my lab visit at the BIWI Computer Vision lab at ETH Zürich. My visit to the BIWI lab was truly an eye-opening experience. Many thanks to Roland Kehl for imparting his experiences and tricks-of-the-trade in body tracking and for making my stay at BIWI enjoyable. I would also like to thank NICTA for providing a vibrant research environment and financial assistance for my PhD.

I would like to extend my thanks to Daniel Heckenberg for his time and the many delightful conversations (both work and non-work related) and for lending me a mocap data glove. Thanks also to my friend and colleague, Hendra Nurdin, for his highly entertaining company and advice. Other honourable mentions from RSISE who have influenced me in one way or another during my PhD study include Robby Tan, Arvin Dehghani, Wynita Griggs and Anika Schumann. Many thanks to them.

I must also use this opportunity to thank all my friends and family, whose support and jovial company have helped me to get to where I am today. Thanks to Mum, Dad and V for their unconditional support and understanding, and for being such a forgiving bunch when I forget to ring home for weeks on end! Special thanks to Annie, Evelyn and our weekly Saturday-night-dinner posse Ivana, Hannah, Michele, Patrice, Anthie and David for keeping me (and each other) sane for the past 4 years. Thanks to my cool running buddies Aditi, Morten and Kate for coercing me to run insane distances as a means of escape (in both the mental and physical sense!). Thanks also to Chloe, Steph, Priscilla, Fred, Jacq, Srinivas, Esther and Suv. Last but certainly not least, thanks to my

friends back home, Raja, Gordon, Yong and Jun, for channeling their support across the ditch.

Abstract

A markerless gradient-based 3D hand tracking system that uses colour intensity and silhouette cues from a pair of video cameras is proposed. The hand tracking system has been designed with the intention of future use as a gesture interface for freeform modelling. The tracking system follows a model-based approach and is comparatively fast, without a heavy compromise to the level of tracking accuracy that is required for such a gesture interface. This is achieved by using a sparse point cloud instead of a complete surface hand model for tracking. However, the point cloud approximation introduces sampling noise into the system. Noise also arises from the video images themselves. It is not immediately clear whether such a tracking system can converge to the optimal pose.

The main contribution of this thesis is to show, using stochastic approximation theory, that it is possible for the tracking system to locally converge to the optimal pose despite the presence of noise. Key results from this analysis are that the ideal cost function of the tracking system has a unique global minimum at the optimal pose for almost all pose configurations and that the Hessian of the ideal cost function at the unique global minimum is positive definite. To the best of the author's knowledge, such an attempt at a convergence analysis for a hand tracking system has not been undertaken in the literature before.

Also proposed in this thesis is an online adaptive vector-autoregressive motion (RVAR) predictor. The RVAR predictor augments the tracking system by initialising a suitable starting point for the tracker's optimisation routine, based on both the tracker's past motion history and the accuracy of the predictor's past predictions. This motion predictor substantially improves tracking performance especially for sequences exhibiting heavy self-occlusion and large finger movements. Experiments also show that tracking in our stochastic setup with the RVAR predictor is much better than tracking with the naïve deceleration motion predictor or an auto-regressive model that is trained with motion capture data obtained from a data glove.

Contents

Terminology

HCI	Human-computer interface
DOF	Degrees of freedom
fps	Frames per second
AR	Auto-regressive
VAR	Vector auto-regressive
RVAR	Robust vector auto-regressive model
PCA	Principal component analysis
ICA	Independent component analysis
SGD	Stochastic gradient descent
SMD	Stochastic meta-descent
BFGS	Broyden-Fletcher-Goldfarb-Shanno method
oBFGS	Online BFGS
LM	Levenberg-Marquardt method
GN	Gauss-Newton method

Chapter 1

Introduction

Marker-less tracking of articulated structures, be it the human body or the human hand, has garnered significant interest in the recent decade in the computer vision community. Research in articulated body tracking has been spurred on by a variety of real-life applications ranging from motion capture in the visual effects industry to human-computer interfaces (HCI). For humans, using hand gestures is an intuitive mode by which to communicate ideas amongst peers [61]. Computer technology has become powerful enough that adopting hand gestures as an alternative interface between humans and desktop computers is fast becoming a realisable possibility. Examples where this mode of interaction is perceived helpful include multimodal user interfaces for traffic control [20], freeform modelling [48, 67], automatic sign language recognition [59, 32] and entertainment consoles *e.g.* the Sony EyeToy and more recently, Microsoft's Project Natal.

The objective of this thesis is to develop a system that tracks the 3D pose of a moving hand in video sequences. The scope of the tracking work is primarily focused towards hand gestures that serve as interface commands for freeform modelling on the average desktop computer.

The inner workings of a gesture-based interface can be generally described as follows. An initialisation step sets up the tracking system for user input, and involves aspects such as camera and model calibration, or finding the initial tracking pose. Next is the extraction of visual cues that are required for pose estimation, *e.g.* silhouette information. Given these visual cues, a pose estimation of the articulated body can be performed on each frame of the video feed. The pose estimates are subsequently mapped to interface commands in the form of gesture recognition. This thesis examines the tracking/pose estimation aspect of the gesture-based interface.

There are many open problems in hand tracking, a selection of which is listed below (see Section 1.1 for our proposed approach to overcome these problems) :

- **Fast finger movements** - The small-angle assumption, which states that the change in pose between time t and $t+1$ is sufficiently small, is frequently made in tracking [26]. This assumption is violated more often in hand tracking than body tracking, as finger flexions are known to reach high angular speeds. Peak angular velocities of the finger joints typically reach up to 6 rad/s [45], or equivalently, up to a peak angle change of 10 degrees per frame for a video captured at 30 frames per second (fps). This issue can be addressed by video capturing at a higher frame rate ($\geq$ 30 fps). However, this results in more computations, as more frames are required for processing. The increased computational load is problematic for online applications. Alternatively, a motion prediction model can be employed [93, 42, 7, 33].

- **Self-occlusion** - Self-occlusion is a common occurrence in hand tracking, *e.g.* when a finger movement partially occludes neighbouring fingers or the palm. Resolving self-occluding poses is not helped by the hand's weak and indistinct texture. Having multiple cameras helps *e.g.* in [28]. Rehg and Kanade [64] use a layered template model combined with a kinematic model to track partially occluded fingers. In Gorce *et al.*'s work [26], texture gradients at the self-occluding contours are analytically derived to improve tracking.

- **Computational speed** - For real-time interactive applications, the computational time spent in tracking needs to be very short. Real-time tracking systems have been achieved [63, 93, 72, 33, 55, 91] albeit with caveats. Many real-time systems can only handle a highly restrictive set of motion [63, 93, 91]. Tracking solutions that do capture a range of complex motions in real-time require multiple processors to handle the computational load [72, 33, 55].

- **Tracking analysis** - An area in the hand/body tracking literature that is underdeveloped is the analysis of tracking systems, in particular, convergence analysis. Whether a tracker can inherently converge to the ideal pose has largely been empirically verified [30]. Generally, the squared sum of differences between the actual and predicted positions of virtual markers on the articulated body is used as an empirical measure for convergence *e.g.*

[7, 92]. To the author's knowledge, there is no theoretical convergence analysis for any of the hand/body tracking systems discussed in the literature, with the exception of the author's previous work [21].

1.1 The Approach

In light of these problems, a model-based 3D hand tracking system has been implemented in a stochastic approximation framework. Two cameras are used to recover 3D pose and help mitigate the effects of self-occlusion. The tracker follows a gradient-based approach - the goodness of fit for a pose estimate is evaluated by projecting the 3D hand model into the model image plane which is compared against the real camera image. Cost evaluation is based on silhouette information and colour intensity. Mismatch errors in the image space are propagated as gradients back to the parameter space of the hand model. An optimisation algorithm is used to refine the pose based on the error gradients.

Rendering the full 3D hand model to the image plane for the evaluation of the cost function is computationally expensive. To address this issue, random samples on the surface of the hand model are selected to approximate the ideal cost instead. Stochastic approximation theory, like the Robbins-Monro method, states that such an approximation is valid under unbiased sampling and certain conditions on the cost function. The ideal cost function of this proposed hand tracking system is shown analytically to have a well-behaved unique global minimum in most scenarios. This property allows the Robbins-Monro method to be applied and therefore show that the tracker still exhibits, in theory, local convergence under point sampling. Experimental results on various hand tracking sequences support this empirically.

To address the issue of fast finger movements and self-occlusion, an online auto-regressive model is introduced as a motion predictor. Many trackers use the constant velocity motion predictor [41, 7, 80, 77] which is often simple to implement but primitive. Second-order dynamic models such as [93, 33] should encapsulate the motion of articulated joints better. These models are trained offline with ground-truth data obtained from a motion capture device *e.g.* a data glove. However offline learning is only effective if the training data is unbiased and rich enough, and may not generalise well to motions not observed in the training data. Thus, an online dynamic model is an attractive alternative. An online Robust Vector-Autoregressive (RVAR) model is presented in this thesis, modeling

the accelerations of the joint parameters as auto-regressive processes. This model is also adaptive in that future predictions are dependent on the reliability of the recent past predictions. Experimental results indicate that the RVAR model is better than the constant velocity motion predictor or the auto-regressive-based motion model that is trained offline using motion capture data obtained from a data glove.

1.2 Thesis Outline

The main contributions of this thesis are:

- A convergence analysis for a gradient-based hand tracking system, using stochastic approximation theory. Parts of this work have been published in the International Conference on Computer Vision (ICCV'07) Workshop on Human Motion [21] and an updated version will be submitted to the IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI).

- An adaptive online dynamic model for the prediction of fast finger motions. Parts of this work have been published in the International Conference on Automatic Face and Gesture Recognition (FG'08) [22].

The thesis is structured as follows:

Chapter 2 A literature review of the current state-of-the-art research in hand tracking is given. Similar research in body tracking is also examined. Both appearance and model-based approaches are explored, with an emphasis on the latter. Topics that pertain specifically to the model-based approach are also investigated, *e.g.* dimensionality reduction.

Chapter 3 The proposed 3D hand tracking system is described. Each aspect of the tracking pipeline is explored in detail, from the hand model, to the projection pipeline and the cost function that is based on silhouette and colour information. Briefly mentioned are the different optimisation algorithms implemented in the tracker and individually tested in the experiments.

Chapter 4 Stochastic approximation theory, in particular the Robbins-Monro method, is introduced. The proposed tracking system is shown analytically to locally satisfy the conditions required by the Robbins-Monro method. These include aspects such as the existence of a well-behaved unique global minimum in the cost function for most scenarios and what it means to achieve unbiased sampling.

Chapter 5 Practical implementation issues and initial experimental results are presented. A discussion is given on choosing the appropriate gestures to evaluate the hand tracking system for the purpose of freeform modelling. Experimental results on synthetic and real video sequences are given.

Chapter 6 In this chapter, an adaptive online VAR model is proposed to model the dynamics of hand movements and thus allow the tracker to handle fast finger motions. A robust version of the VAR motion prediction model is given, which is compared against the constant-velocity model and a VAR model trained off-line with data obtained from a data glove.

Supplementary Videos Video clips of the tracking results for the real sequences can be found in the accompanying CD-ROM. The best tracking results achieved with this tracking system can be found under the directory /Chapter6/10000/Seq-All/.

1.3 Publication List

This thesis is partly based on the following papers that have been published in peer-reviewed conferences/workshops:

- D. Chik, J. Trumpf, and N. N. Schraudolph, Using an adaptive VAR model for motion prediction in 3D Hand Tracking. In *8th Intl. Conf. Automatic Face and Gesture Recognition*, Amsterdam, Netherlands, September 2008.
- D. Chik, J. Trumpf, and N. N. Schraudolph, 3D hand tracking in a stochastic approximation setting. In *2nd Human Motion Workshop - Understanding Modeling, Capture and Animation*, Rio de Janiero, Brazil, October 2007.

The contents of the following related paper do not form part of this thesis:

- D. Chik. Using optical flow for step size initialisation in hand tracking by stochastic optimisation. In *HCSNet Workshop on the Use of Vision in HCI (VisHCI)*, volume 56 of *Conference in Resarch and Practice in Information Technology (CRPIT)*, Canberra, Australia, November 2006.

Chapter 2

Prior Work

We begin at the fundamental level by examining the visual cues used in state-of-the-art tracking systems in the literature. Next, the many approaches to hand tracking are explored. These approaches are broadly categorised as appearance-based and model-based, with an emphasis on the recent advancements in model-based tracking. Works in body tracking are also examined, as the techniques used in tracking the articulated human body are usually applicable to the hand tracking scenario.

2.1 Visual Cues

Camera images are the "eyes" of the tracking system. The questions that immediately arise in designing a tracking system are: 1) what types of visual information can be gleaned from the images to aid in tracking, and 2) how many camera viewpoints are required.

The silhouette of the articulated object is a standard cue used in hand and body tracking [18, 2, 84, 81, 74, 37]. It constrains the region in the image that the articulated object can occupy. Most approaches explicitly extract the silhouette of the articulated object [18, 2, 84, 81, 74] although in Gorce *et al.*'s work [26], the silhouette of the hand is implicitly extracted by incorporating a model of the background in the cost function of the tracking system. Methods for silhouette extraction include background subtraction [41] or segmentation via a learned probabilistic colour model of the articulated body, such as skin colour detection in [11] and [77]. Some systems pre-process multiple silhouettes from different viewpoints together to perform volume reconstruction [42, 53, 55, 86, 16]. The reconstructed volume of the body is then used for pose estimation directly. Menier

et al. [53] takes this further by extracting the skeleton from the visual hull and using it to match that of the body model.

Edges of the articulated body are also used by many [42, 5, 40, 91, 7, 77, 63, 49] as they are generally invariant to lighting. Edge features are typically extracted via filters such as the standard Canny edge detector [17]. However, tracking with edges can become problematic in cluttered scenes. Stray internal edges on the articulated body (*e.g.* creases of the subject's shirt, or a body that is rich in texture) or edges from background objects can be a distraction for the tracker. Priors derived from the structure of the articulated body can be used to weed out unlikely edges [5]. Some systems pre-process the edge map to generate a contour map of the articulated body based on similarities in the neighbouring edge directions. These contour maps are then used instead for tracking [27].

Colour intensity is a cue that is sometimes used [42, 26, 76, 49]. Typically the system has an inherent knowledge of the object's coloured texture, *e.g.* via a learned colour model, which is used to match to the texture seen in the real images [42, 26]. In Lu *et al.*'s work [49], shading from lighting in the scene is used for tracking. In [11], structured lighting is used to recover depth information that is then used for pose estimation. In [77] a probabilistic colour model is used to define the likelihood region in which the hand lies.

Motion fields, *e.g.* optical flow, are yet another type of feature used for tracking and provide temporal information. Lu *et al.* [49] for example, account for optical flow in pose estimation. In Theobalt *et al.*'s work [84], the reconstruction of a 3D motion field is used to help with tracking. In [81], motion and spatial cues from images are used to recover displacements of the body segments in the parameter space.

The number of cameras used is dependent on the intended application for the tracking system. Monocular view approaches have the disadvantage of depth ambiguity in pose estimation. Nevertheless, using a single camera might be enough for certain applications such as a gesture recognition interface for a desktop computer. In this situation, working out the exact 3D pose is not so important, as long as the tracking system can distinguish between the discrete set of command gestures. For applications that require more precise pose reconstruction where the exact 3D pose is important, *e.g.* motion capture or freeform modelling, multiple cameras are essential.

2.2 The Appearance-Based Approach

In this approach, the hand state is estimated directly by comparing known 2D images of the hand with the observed image. These include database-search methods [72, 5, 76, 77] and learned mappings [66] that map the observed appearance directly to the hand pose space.

Shimada *et al.* [72] use template matching to determine the pose of the hand. A database containing over 16,000 silhouette contours of the hand in various poses is constructed from a 3D hand model. The silhouette contour of the hand in the input image is compared against this database. An adjacency map is used to optimise the search, where poses with similar joint angles are connected to each other. The search for the pose in the next frame is then limited to the neighbourhood of the current estimate.

Athitsos and Sclaroff [5] follow a similar path to Shimada *et al.* [72], using a database of synthetic images for pose estimation, albeit for a restrictive set of hand poses. Edges generated by the contours of the hand are used for matching hand shapes. By exploiting the fact that the edge segments of each digit form a set of connected 'nearly straight' line segments, stray edges can be identified and removed in a probabilistic framework. This allows the tracking system to handle images with a cluttered background. Processing time for each frame is about 15 seconds on a 1.2 GHz computer. The authors state that although the retrieval rate is not reliable enough for the system to work as a stand-alone hand tracker, the tracking system could be useful for tracking initialisation or error recovery.

In Rosales *et al.*'s work [66] a silhouette database is generated from rendered viewpoints of various hand poses obtained from a data glove. These rendered silhouettes are represented by their visual feature vectors based on the Hu moments [34]. A set of specialised mappings, generated via supervised learning, maps the feature vector to multiple hypotheses in hand pose space. To estimate a particular hand pose, its feature vector is extracted, from which multiple hypotheses in the hand pose space are obtained via the set of specialised mappings. The best candidate is then chosen based on the similarity of the candidate's feature vector to the observed feature vector.

Stenger *et al.* [76, 77] draw on the ideas of grid-based filtering in the search for the optimal hand pose. The state space, represented by hand appearance templates, is split into sets of partitions at different resolution levels. The coarsest partition set forms the starting level of a database tree, which branches into a partition set of a finer resolution as one traverses down a level in the tree. Each level

of the tree is seen as a grid approximation of the posterior distribution. At any given level, partitions that pass a fixed likelihood threshold are explored further down in the next level, thereby refining the pose of the likely candidates. Later work in [77] introduces learned dynamics into the discrete filtering framework to increase robustness.

2.3 The Model-Based Approach

For model-based approaches, the tracking system uses an articulated 3D model of the hand or the body for matching. This can be matching in 2D space, where the projection of the 3D model (evaluated online) is matched against the observed 2D image of the hand [26, 62, 49, 47, 91, 80]. Alternatively, model matching can be performed in "$2\frac{1}{2}$D" or 3D space [42, 11, 53, 55]. Model-based techniques in the literature for finding the optimal pose can generally be categorised as either gradient-based or particle filter-based. Dimensionality reduction and dynamic models are also useful methods for improving efficiency and robustness.

2.3.1 The Gradient-Based Approach

Gradient-based approaches generate gradients from the mismatch error in a cost function. Via an optimisation algorithm, these gradients guide the tracking system to a better pose estimate. The advantage of gradient-based approaches is that they tend to be computationally fast as the gradient information drastically narrows the search space. However, convergence to the global optimum is not always guaranteed, unless the cost function exhibits sufficient properties such as strict convexity, which is rare in practice. At worst, gradient-based approaches can converge to local optima. Nevertheless, one can attempt to avoid local minima by running multiple instances of the optimisation algorithm at different initial states, or by using simulated annealing techniques. This is analogous to the idea of maintaining multiple hypotheses in particle filtering.

The work of Rehg and Kanade [62, 63, 64] represents some of the earliest research in model based 3D hand tracking. The *DigitEyes* tracking system in [63] has two cameras, and uses fingertip points and occlusion edges for model alignment. Levenberg-Marquardt is the optimisation algorithm used. Only non-occluding hand movements are examined in [63], although the issue of self-occlusion is addressed with the use of edge templates and a kinematic model in the subsequent work [64].

In the work of Bray *et al.* [11] the 3D hand pose is estimated from a depth map obtained from structured lighting. The cost function utilises depth information as well as surface normals. A stochastic sampling approach is introduced where instead of generating the entire synthetic depth map of the hand model for cost evaluation, sample points on the hand surface are used to evaluate an approximate of the cost function. The stochastic optimisation algorithm SMD (stochastic meta-descent)[68] is used.

In the work of Kehl *et al.* [42] a voxel-based fitting technique is applied, whereby the model is fitted to a volume reconstruction of the subject in 3D space. The volume is generated from multiple silhouette views. In addition, 2D contour edges on the subject's body are used to make the tracking more robust. A Gaussian mixture model of the subject's colour profile is also used as a prior in the overall cost function to resolve difficult poses where the forearms of the subject are in front of the torso. Like in Bray's work, the objective function is minimised in a stochastic optimisation setting. The sampling approach allows for a speed up of the processing time to 1 second per frame on average.

In Lu *et al.*'s work [49] the hand is modelled as a dynamic system influenced by forces due to edges and shading in the image space. Edge forces are generated from mismatches between the model edges and the observed edges. Forces due to intensity change are determined by solving a constraint equation that incorporates optical flow, the directional change in illumination at a pixel due to motion, and shading flow, the change in illumination intensity of a point on the hand surface due to rotational movements. These forces are propagated as gradients back to the parameter space and are used to recursively update the hand pose.

Kaimakis and Lasenby [37] have used silhouette information from multiple cameras for hand tracking. Each segment of the hand model is represented by ellipsoids. The projections of these ellipsoids form conic fields. Correcting gradients from the conic fields are analytically derived and are used to correct mismatch errors. The Gauss-Newton method is the optimisation algorithm used.

Recently, in Gorce *et al.*'s work [26] the 3D hand pose from a single view is recovered using a dynamically estimated texture of the hand and estimated light sources in the scene. Given the background image, a synthetic image is generated for a particular hand pose estimate. A cost function penalises differences in pixel values between the synthetic and the observed image. Sequential quadratic programming with the BFGS (Broyden-Fletcher-Goldfarb-Shanno) approximation of the Hessian is used as the optimisation routine. Image gradients at the self-occluded boundaries are analytically derived for better accuracy. Af-

ter the optimised pose is found, the texture model of the hand is updated to minimise noise introduced by the background. Spatial and temporal smoothness constraints are applied in the texture update routine for robustness. Cast shadows are not addressed in this work.

2.3.2 The Particle Filter-Based Approach

Particle filtering is a popular technique employed in tracking [91, 47, 33, 7] as it can approximate prior probability density functions that exhibit multiple modes of likelihood. The effect of this translates to maintaining multiple hypotheses about the pose of the articulated body. Keeping track of multiple hypotheses gives the tracker a better chance of escaping lesser modes of likelihood (or equivalently a local minimum) when the modes of likelihood evolve with time.

Having a smart particle resampling scheme often dictates how well a particle filter performs, in particular when the dimensionality is high. For example a resampling scheme that exploits prior information *e.g.* human motion dynamics [91, 33] means that less samples (and consequently, less computational time) are required to search for the important modes in the probability density function. Bray *et al.* [12] propose the use of SMD to propagate the particles to likely minima. Partition sampling [51], annealed particle filtering [24] and non-parametric belief propagation [80] are further examples of schemes aimed to improve sampling efficiency.

In Lin *et al*'s work [47] a combination of particle filtering and a direct search method is proposed. Firstly the hand configuration space is approximated by a discrete set of poses captured with a data glove. Then in a two-stage approach, the Nelder-Mead (NM) search method is used to search for the closest pose estimate within this discrete set before refining the search in the continuous neighbourhood of this pose estimate. To address the issue of local minima, particle filtering is employed to maintain multiple hypotheses for the hand pose. The NM method is also used to locate the modes of likelihood in the tracking prior.

In Sudderth *et al.*'s work [80] a graphical model is used to represent the hand. Each segment of the hand is treated as a set of rigid bodies. Kinematic, structural and temporal constraints are described as statistical dependencies between the rigid segments. The estimation problem is posed in a graphical model framework using Nonparameteric Belief Propagation (NBP).

In Chang *et al.*'s [19] work a modified form of particle filtering is introduced. Instead of the traditional Bayesian formulation where the current state is only

conditionally dependent on the previous state, the current state in the new formulation is also conditioned on a set of "attractors". These attractors represent key hand pose states whose observations are known. The attractors aid in pose transitions by guiding the particle filter to resample near more probable regions in the state space.

2.3.3 Dimensionality Reduction

Pose estimation is often directly evaluated in the joint angle space of the articulated model [26, 11, 42]. This space is generally of a high dimensionality (>25 degrees of freedom), unless the set of articulated movements is artificially restricted such as in [76]. The high dimensionality can be problematic, particularly so for particle filter based trackers where, at worst the number of samples required for tracking scales exponentially with an increase in dimension [91]. Fortunately, general human motion has been shown to lie in a lower dimensional subspace or submanifold of the high dimensional joint angle space [91, 44, 77]. This is due not only to the structural constraints of the articulated body but also the kinetic constraints *e.g.* tendon forces that effect the simultaneous movement of adjacent digits [85]. Therefore, dimensionality reduction can be applied via techniques such as Principal Component Analysis (PCA) [40, 3, 77] and a learned manifold representation of the pose space [91, 44].

In the work by Kato *et al.* [40] the dimensionality is reduced by applying PCA on hand motion data captured by a data glove. Independent Component Analysis (ICA) is then applied to the reduced space to extract feature vectors of hand motion, in particular, finger articulation. This transforms the problem to a reduced space of 11 dimensions. Particle filtering on this reduced space is implemented for hand tracking. Experimental results limited to motions of the hand in the frontal view show this to be a promising approach.

In Zhou *et al.*'s work [93] dimensionality is reduced via PCA to a 6D space. The authors assert that 99.79% of the variance in motion is preserved, although the type of motion explored represents a very restricted subset of natural articulated hand movements.

In Wu *et al.*'s work [90, 91] PCA is initially applied to reduce the dimension of the state space to 7, preserving 95% of the variance. For the specific set of hand gestures investigated, it is observed that the hand poses lie on linear manifolds in this reduced space. The importance function of their particle filtering framework is learned from this set of linear manifolds. Experimental results show

that using such an importance function outperforms the naïve random search and the classical condensation [35] approach. However, the authors note that the system has difficulty handling severe out of plane motion and motion with scale changes.

2.3.4 Dynamics

Modelling of dynamics allows for motion prediction. With motion prediction, the pose estimate can be initialised closer to the optimal pose based on the motion trajectory observed in the past frames. This is especially useful for fast finger/limb motion and for resolving ambiguities when the fingers are moving simultaneously. Starting the pose search closer to the optimal value via motion prediction generally means that the tracker has less local minima to overcome in order to reach the optimal pose. Motion prediction models of varying complexities have been used, from first order predictors, *e.g.* zero acceleration/constant velocity predictors in [42, 7, 80] to motion predictors of higher orders *e.g.* [2, 33].

Constant velocity predictors are popular and tend to be straightforward to implement. Many assume parameter independence in the prediction model [42, 80, 7], and models are often tempered with a heuristic damping factor to avoid overshooting [42, 7]. In Bayesian frameworks the velocity of each pose parameter is typically modelled as an independent Gaussian stationary process [80, 7]. In Stenger *et al.* [77] the initial state for the next frame is predicted via a first order Markov state transition matrix. This transition matrix is learned via training data from a data glove.

Zhou *et al.* [93] present a more sophisticated dynamic model. A linear dynamic system is initially trained to model the evolution of the pose estimate in a reduced PCA space. To improve robustness, "eigen-dynamics analysis" is used to introduce structural constraints relating to finger dynamics. In essence, the transition matrix of the linear dynamic system is solved as five de-coupled second order subsystems. Each subsystem is independently trained and represents the hand dynamics for a set of movements where the motion of one particular hand digit dominates.

2.4 Summary

A review of the current research in hand and body tracking has been given in this chapter. Silhouettes, edges, colour and motion field are all common visual

cues used for tracking. Both appearance-based and model-based methods are discussed, although it appears that the latter class of methods has gained popularity in recent years. The class of model-based methods can be further subdivided into the gradient-based approach and the particle filtering-based approach. Both have their advantages and drawbacks. Gradient-based approaches achieve rapid convergence, albeit locally. Particle filtering on the other hand maintains multiple hypotheses but scales badly to an increase in dimension and requires sensible resampling schemes, which invariably require offline training. Research into more robust and computationally efficient forms of tracking include dimensionality reduction and incorporating dynamics in the pose estimation process.

Chapter 3

The Tracking System

The tracking system follows a gradient based approach where mismatch errors between the projected pose of the 3D hand model and the observed pose are back-propagated as gradients in the hand pose space. An optimisation algorithm uses these gradients to refine the hand pose estimate. A cost function based on silhouette and colour information from a stereo pair of cameras (see Figure 3.1) provides a measure for the goodness of fit.

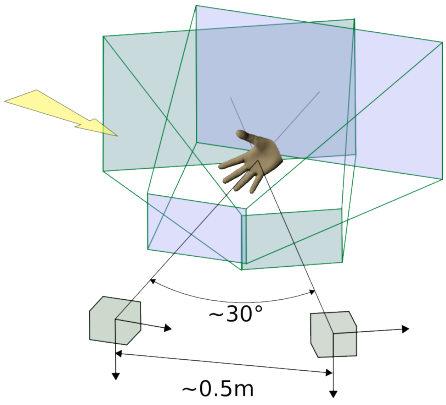

Figure 3.1: Diagram of the tracking system setup. A pair of cameras (square boxes) is placed in a convergent setup facing the hand. A strong light source (yellow arrow) that points towards the hand from above and behind one of the cameras illuminates the scene.

Evaluating the cost (thereby generating the error gradients) by projecting the full 3D model pose onto the image plane for comparison is computationally expensive. This is equivalent to projecting a sufficiently dense set of sample points on the hand model surface that faithfully represents the full model pose

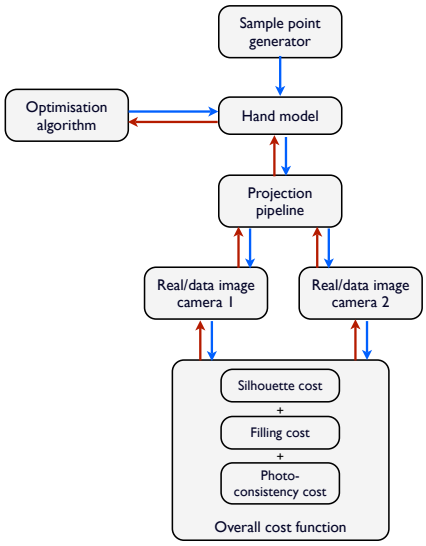

Figure 3.2: Flow diagram of the main components in the tracking system. Blue arrows indicate the forward mode for cost evaluation while the red arrows indicate the back-propagation of error gradients.

projection. Instead, our tracking system only uses a much smaller subset of sample points on the hand model surface to evaluate an approximation of the true cost. This approximate cost is used to refine our pose estimate. One can show via the Robbins-Monro method [65] in stochastic approximation theory (see Chapter 4) that running an iterative optimisation scheme on the approximate cost instead of the true cost is valid under certain conditions.

Figure 3.2 is a flow diagram of the tracking system. With the exception of the sample point generator, each of the steps depicted in the flow diagram will be examined in this chapter. The inner workings of the sample point generator will be dealt with in Chapter 4 as they relate to choosing an unbiased sampling scheme required by the Robbins-Monro method.

3.1 The Hand Model

Different types of hand models with varying complexity have been used in the tracking literature. One popular class of methods is to model the body as chains of discrete rigid bodies linked together. These rigid bodies are typically represented in a parametric manner using quadrics such as 3D ellipsoids [42] or truncated cones [77]. Modelling the hand as a collection of rigid parametric bodies has

the advantage of simplifying (and consequently speeding up) certain calculations such as model projections in 2D space and contour extractions.

Alternatively, one can use a deformable mesh driven by an underlying skeleton to represent the articulated body [13, 86, 4]. Although more complex, deformable mesh models are ideal as they are generally more accurate representations of the articulated body and consequently allow for more refined pose estimates. A deformable mesh model is used in our tracking system for better accuracy. The increased computational cost associated with the more complex model is manageable and not prohibitive due to the reduced load from sparse point sampling.

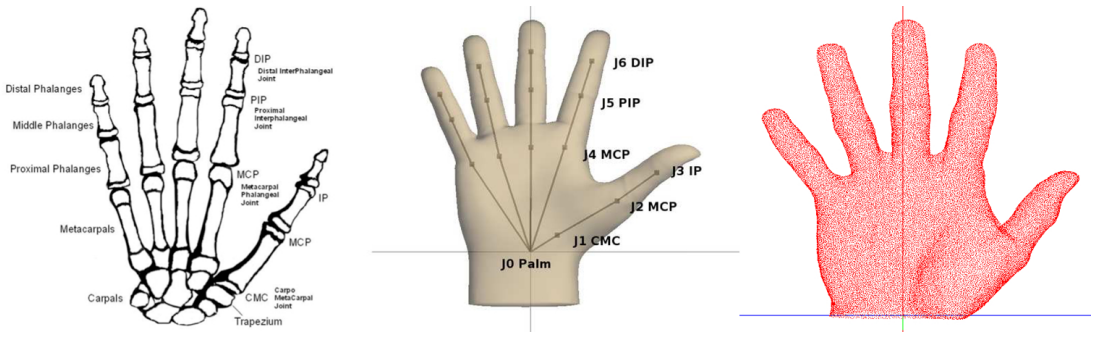

Figure 3.3: Left: Skeletal structure of the hand. (Image taken from McDonald *et al.* [52]) Middle: The hand model used for hand tracking. Right: The deformable mesh of the hand model.

We choose to model the human hand as a deformable mesh driven by an underlying skeleton with 16 joints, totalling 26 degrees of freedom (DOF) [38]. 6 DOFs at the palm joint defines the global rotation and translation of the hand. The MCP joint of each finger has 2 rotational DOFs to encapsulate the abduction and flexion movements at the joint. The DIP and PIP joints of the fingers each have 1 rotational DOF for joint flexion. Finally, the thumb has 2 DOFs for the CMC and 1 DOF for each of the MCP and PIP joints. Note that the x-rotation at the CMC joint is modelled as a rotation at an imaginary joint (see Figure 3.3) further down the kinematic chain to mimic thumb abduction better.

The rotations at all joints are parameterised using Euler angles. The gimbal lock problem is often associated with the Euler angle parameterisation. This degeneracy only applies to the palm joint as the other joints have two or less DOFs. Even so, the set of angles where the degeneracy occurs is finite. Hence our Euler angle representation for a 3D rotation at the palm is unique almost everywhere. We find the Euler angle parameterisation to be adequate for our

purposes in practice.

The dense mesh models the skin surface and is acquired from the 3D scanning of a real hand. It is bound to the underlying skeleton via linear skin blending [46]. Linear skin blending allows sample points taken near the joint regions to deform in a more realistic manner when the joint is bent. The movement of each vertex in the triangulated mesh is influenced by up to 3 different joints. Let h_k be a sample point/vertex on the hand model surface expressed in the local coordinates of joint k and $w_k \in [0, 1]$ be the weighted influence of joint k on h_k. Also let T_k^0 be the rigid transformation from the local coordinate system of joint k to the world coordinate system. Then p, the overall position of the sample point in the world coordinates after linear blending is

$$p = \sum_k w_k T_k^0 h_k, \tag{3.1}$$

where

$$\sum_k w_k = 1. \tag{3.2}$$

In an application scenario, the hand model must be able to generalise to hands of varying sizes in order for the tracker to correctly estimate the pose of any observed hand. The aspect of hand model initialisation is discussed later in Chapter 5.

3.1.1 Hand Model Constraints

The constraints on the hand model include both static and kinematic-based constraints. Table 3.1 shows the static range for each joint of the hand model.

We follow the work of McDonald *et al.* [52] to model the inherent twist that occurs at the MCP joint of the fingers during flexion. This twist prevents the fingers from colliding when a fist pose is formed. It is modelled as being linearly dependent on the y-rotation at the MCP joints. The y-rotation is responsible for finger abduction/adduction. The slight twisting of the thumb that occurs when it is brought in front of the palm is also modelled as a linear dependence on the x-rotation twist of the CMC joint.

3.2 The Projection Pipeline

The projection pipeline projects the ith sample point p_i on the hand model surface to the camera model image plane. Assume that p_i are given in homogenous

Table 3.1: The range of joint angles

Digit	Joint	Angle range (°)	Digit	Joint	Angle range (°)
Thumb	CMC (x-rot)	-80 — -5	Ring	MCP (y-rot)	-21— 34
	CMC (y-rot)	-70 — -20		MCP (z-rot)	-10 — 90
	MCP	0 — 90		PIP	-10— 115
	PIP	-15 — 90		DIP	0—90
Index	MCP (y-rot)	-33 — 22	Little	MCP (y-rot)	-11— 44
	MCP (z-rot)	-10 — 90		MCP (z-rot)	-10— 90
	PIP	-10 — 115		PIP	-10— 115
	DIP	-0— 90		DIP	0— 90
Middle	MCP (y-rot)	-25 — 30			
	MCP (z-rot)	-10 — 90			
	PIP	-10 — 115			
	DIP	0 — 90			

coordinates in the world coordinate frame. Let T_0^j be the rigid transformation that takes a point in the world coordinates and transforms it to the jth camera coordinates. It has the general form of

$$T : \mathbb{R}^4 \to \mathbb{R}^4, \quad p \mapsto \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} p, \tag{3.3}$$

where $R \in SO(3)$ is a rotation matrix and $t \in \mathbb{R}^3$ is a translation vector. Let K_j be the projection matrix for the jth camera. K_j has the general form of the calibration matrix of a CCD camera [29] *i.e.*,

$$K := \begin{pmatrix} \alpha_x & 0 & o_x & 0 \\ 0 & \alpha_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{3.4}$$

where α_x, α_y are the camera focal lengths with respect to the x and y directions in the image plane and (o_x, o_y) is the camera's principal point. The cameras are assumed to be pre-calibrated prior to tracking. Lastly, let D_i be the depth normalisation function *i.e.*

$$D_i : \mathbb{R}^3 \to \mathbb{R}^2, \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{1}{z}x \\ \frac{1}{z}y \end{pmatrix}. \tag{3.5}$$

Then $s_{i,j}$, the projection of the ith sample point on the jth model image plane, is given as

$$s_{i,j} = D_i K_j T_0^j p_i. \tag{3.6}$$

Note that $s_{i,j}$ is dependent on $\mathbf{x} \in \mathbb{R}^{26}$, the vector of hand parameters that defines the hand pose. This is because the positions of the sample points p_i are dependent on $\mathbf{x}$.

3.3 The Cost Function

Given a set of sample point projections on the model image plane, one can look at the corresponding pixel coordinates in the real and data images, respectively, to evaluate the cost*. The overall cost function uses silhouette and colour intensity to determine the goodness of fit. It is the weighted sum of a silhouette cost function C_s, a filling cost function C_f and a photo-consistency cost function C_p.

Let $\mathbf{x}^*$ be the vector of optimal hand parameters, corresponding to a perfect fit of the model to the actual pose. The overall cost function $C_{\mathbf{x}^*}$ is given as

$$C_{\mathbf{x}^*}(\mathbf{x}) = \alpha C_s(\mathbf{x}) + \beta C_f(\mathbf{x}) + C_p(\mathbf{x}), \tag{3.7}$$

evaluated over a sufficiently dense set of sample points chosen from the hand model surface. α and β are scalar weights. It is shown in Chapter 4 that this overall cost function is well-behaved at its unique global minimum.

3.3.1 The Silhouette Cost Function C_s

The silhouette cost function penalises the tracker when the projected hand model does not lie within the silhouette of the real hand in the camera images. The cost function uses a distance map obtained from applying a chamfer distance transform [9] over the silhouette image extracted from the real images. It assigns

*Unless stated otherwise, the word *cost* is henceforth taken to mean the *true* cost, i.e. the cost evaluated from a sufficiently dense set of sample points. One should note its distinction from the *approximate* cost, *i.e.* the cost value obtained from sparse point sampling.

a distance value V to each pixel based on the pixel's proximity to the closest pixel that belongs to the hand silhouette. Thus, the silhouette cost function over j camera views is given by

$$C_s(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{2} V(s_{i,j})^2, \tag{3.8}$$

where N is the cardinality of a sufficiently dense set of sample points. Note that V is dependent on the set of optimal hand parameters $\mathbf{x}^*$ in the sense that $\mathbf{x}^*$ determines the hand pose seen in the real camera images.

3.3.2 The Filling Cost Function C_f

The filling cost function C_f penalises the tracker when the projection of the hand model does not fill the hand silhouette completely. A dense set of sample pixels is randomly chosen from within the *silhouette* to evaluate C_f. Let $\hat{s}_{i,j}$ be the pixel coordinate of the ith sample pixel chosen inside the silhouette in the jth camera view. Let $h_{i,j}(\mathbf{x})$ be the pixel coordinate of the point on the hand model projection that is closest to $\hat{s}_{i,j}$. Then the filling cost function over two camera views is given as

$$C_f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{2}\frac{1}{2}||\hat{s}_{i,j} - h_{i,j}(\mathbf{x})||^2, \tag{3.9}$$

where M is the cardinality of the dense set of sample points. Note from (3.9) that sample pixels which are covered by the projection of the hand model do not contribute to the filling cost since $\hat{s}_{i,j} = h_{i,j}(\mathbf{x})$ in such a situation.

3.3.3 The Photo-Consistency Cost Function C_p

The photo-consistency cost function is used for local fine-tuning by resolving pose ambiguities in the silhouette information. We first assume the hand surface to be well approximated by the Lambertian surface model [25]. As such, one expects to observe the same YUV colour intensity for a given point on the hand surface regardless of the camera viewpoint. Thus, the photo-consistency cost function penalises the tracker when the observed YUV values of the projections of a sample point in the two camera views are not equal. The rationale is that differing YUV values suggest that the projections do not correspond to the same point on the hand surface, implying that the pose estimate is incorrect.

Let I be the function that maps a given pixel coordinate to a YUV value. I is also dependent on $\mathbf{x}^*$ since $\mathbf{x}^*$ determines the YUV values of the pixels in the real camera images. For two camera views, the photo-consistency cost function is given as

$$C_p(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}||I(s_{i,1}) - I(s_{i,2})||^2, \tag{3.10}$$

where N is the cardinality of a sufficiently dense set of sample points.

3.4 Gradients

The chain rule is used to propagate the error gradients in the camera projective space to the hand parameter space. As an example, suppose we wish to determine the change in cost C with respect to a change in the world coordinate space, *i.e.* $\frac{dC}{dp}$. Given that the error gradient in the image space $\frac{dC}{dD}$ is known, we apply the chain rule to get

$$\frac{dC}{dp} = \frac{dC}{dD}\cdot\frac{dD}{dK}\cdot\frac{dK}{dT}\cdot\frac{dT}{dp} \tag{3.11}$$

$$= R^{\mathrm{T}}K^{\mathrm{T}}\frac{dD}{dK}^{\mathrm{T}}\frac{dC}{dD}. \tag{3.12}$$

The chain rule is continually applied until the hand parameter space is reached.

So far we have assumed that each module in the derivative chain is analytically differentiable. This is not the case for the distance map function $V(s_j)$ used by the silhouette cost function and the colour intensity mapping function $I(s_j)$ used by the photo-consistency cost function. In practice, the image gradients $\frac{dV}{ds_{i,j}}$ and $\frac{dI}{ds_{i,j}}$ are determined empirically by running a derivative filter over the data images to obtain the image gradients. Implementation details of this can be found in Chapter 5.

3.5 Optimisation Algorithms

Gradients generated by the tracking system pipeline are used by the optimisation algorithm to refine the hand pose. More precisely we wish to find the optimal hand pose, $\mathbf{x}^*$,

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} C_{\mathbf{x}^*}(\mathbf{x}), \tag{3.13}$$

$$\text{subject to } \mathbf{x} \in \varphi, \tag{3.14}$$

where φ is the set of allowable hand poses governed by the static constraints of the hand model.

In practice, we cannot observe the true cost $C_{\mathbf{x}^*}(\mathbf{x})$ but we do observe the approximate cost $C_{\mathbf{x}^*}(\mathbf{x}, \Phi)$. Here, the approximate cost is dependent on the random variable Φ which represents the effects of noise due to sparse point sampling and also camera noise. At best, we can only seek to minimise the expectation of the approximate cost,

$$\mathbf{x}^*_\phi = \arg\min_{\mathbf{x}} \mathrm{E}_\Phi[C_{\mathbf{x}^*}(\mathbf{x}, \Phi)], \tag{3.15}$$

$$\text{subject to } \mathbf{x} \in \varphi. \tag{3.16}$$

Stochastic approximation theory states that under suitable conditions, $\mathbf{x}^*_\Phi \rightarrow \mathbf{x}^*, a.s.$ at the limit. This will be elaborated further in the next chapter. For now one should observe that the true gradients $\nabla C_{\mathbf{x}^*}(\mathbf{x}_t)$ are not used in the optimisation algorithm, but rather the gradient estimates $\nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t{}^\dagger)$.

The choice of optimisation algorithm to use in the tracking system is up to the user. Several optimisation algorithms are tried out in the experiments for comparison. These are explored in the following subsections. Note that the Robbins-Monro method (see chapter 4) used to prove the theoretical local convergence of the tracker at the limit requires a bound on the optimisation algorithm's step size. SGD and oBFGS can be proven to satisfy this bound on step size.

3.5.1 Stochastic Gradient Descent

The iterate update equation for SGD can be expressed as

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_0 \frac{\tau}{\tau + t} \nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t), \tag{3.17}$$

where $\eta_0 > 0$ is the initial step size and $\tau > 0$ is a tuning parameter. $\mathbf{x}_{t+1}$ is checked at each iteration to see if it lies within the set of hand poses bounded by the static hand constraints (Table 3.1). Any outlying parameters are mapped back onto the boundary of the feasible region.

3.5.2 Stochastic Meta-Descent

Stochastic meta-descent (SMD) [68] has been used in both hand [11, 13] and body tracking [42] and is empirically shown to achieve fast convergence in stochastic

$^\dagger\phi_t$ is the realisation of Φ at time t.

approximation settings. The iterative update equation for SMD differs from SGD in that it follows an adaptive scheme for choosing an appropriate step size vector $\eta_t \in \mathbb{R}^{26}$. The step size adaptation takes into account the past history of step sizes as well as the curvature information of the cost function via the Hessian H_t. The iterative update equations are given as

$$\eta_{t+1} = \eta_t \cdot max(\frac{1}{2}, 1 - \mu \nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t) \cdot v_t), \tag{3.18}$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_{t+1} \cdot \nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t), \tag{3.19}$$

$$v_{t+1} = \lambda v_t - \eta_{t+1} \cdot (\nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t) + \lambda H_t v_t), \tag{3.20}$$

where $\cdot$ denotes the Hadamard product and $\mu \geq 0$ and $\lambda \in [0, 1]$ are tuning parameters. The initial value of v_0 is set to be the zero vector.

Since our cost function is non-convex, H_t will not be positive definite everywhere. This is undesirable in practice as it can lead to divergence. An alternative is to use the extended Gauss-Newton matrix G_t [69] to approximate H_t. Note that there are fast $O(n)$ methods available for calculating the Gauss-Newton vector product $G_t v_t$ [69].

To incorporate hand constraints, outlying parameters are mapped back onto the boundary of the feasible region just like in SGD. Let $\mathbf{x}^c_{t+1}$ be the parameter vector that has been mapped back to the feasible region. For consistency, this change must be reflected in the v_{t+1} update, which consequently affects the next step size evaluation η_{t+2}. This is done by calculating the hypothetical constrained gradient $\nabla^c C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t)$, as outlined in [11] as

$$\nabla^c C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t) = \frac{\mathbf{x}_t - \mathbf{x}^c_{t+1}}{\eta_{t+1}}. \tag{3.21}$$

This constrained gradient replaces $\nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t)$ in the v_{t+1} update in Equation (3.20).

3.5.3 Online BFGS

The online Broyden-Fletcher-Goldfarb-Shanno method (oBFGS) [71] is an adaptation of the traditional BFGS method [57] to the stochastic approximation setting. oBFGS differs from BFGS in that the update of the inverse Hessian estimate B_t is slightly different and that the line search routine is absent in oBFGS. The iterative update equation for oBFGS is given as

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{\eta_t}{c} B_t \cdot \nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t), \tag{3.22}$$

where $0 < c \leq 1$ is a tuning parameter and n_t is a real sequence of the following form

$$\eta_t = \eta_0 \frac{\tau}{\tau + t}. \tag{3.23}$$

The starting inverse Hessian estimate is taken to be $B_0 = \epsilon I, \epsilon > 0$. B_t is updated as follows; let

$$s_t = -\frac{\eta_t}{c} B_t \cdot \nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t) \tag{3.24}$$

and

$$y_t = \nabla C_{\mathbf{x}^*}(\mathbf{x}_{t+1}, \phi_t) - \nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t) + \lambda s_t, \tag{3.25}$$

where $\lambda \geq 0$ is the trust region factor. Also let

$$\varrho_t = (s_t^{\mathrm{T}} y_t)^{-1}. \tag{3.26}$$

Then the update equation for B_t is given as

$$B_{t+1} = (I - \varrho_t s_t y_t^{\mathrm{T}}) B_t (I - \varrho_t y_t s_t^{\mathrm{T}}) + c \varrho_t s_t s_t^{\mathrm{T}}. \tag{3.27}$$

One should note that at $t = 0$, B_0 in the B_{t+1} update Equation (3.27) is not ϵI but $\frac{s_t^{\mathrm{T}} y_t}{y_t^{\mathrm{T}} y_t} I$. See [71] for further details.

Both oBFGS and BFGS require $s_t^{\mathrm{T}} y_t > 0$ for all t so that B_t is positive definite. This is not guaranteed in our situation since the cost function is non-convex. To ensure that $s_t^{\mathrm{T}} y_t > 0$ is satisfied, Equation (3.25) can be changed to

$$y_t = G_t s_t, \tag{3.28}$$

where G_t is the extended Gauss-Newton matrix described in [69].

To incoporate hand constraints, an outlying $\mathbf{x}_{t+1}$ is mapped back to $\mathbf{x}_{t+1}^c$. The constrained s_t^c

$$s_t^c = \mathbf{x}_{t+1}^c - \mathbf{x}_t, \tag{3.29}$$

replaces s_t in the B_t update to ensure consistency.

3.6 Summary

An overview of the model-based tracking system has been presented. A deformable mesh model is chosen over parametric hand models for accuracy. A sparse set of sample points on the model surface are used to evaluate an approximate cost. The tracker's overall cost function uses silhouette and colour intensity information as an indication of the goodness of fit. Error gradients from the approximate cost function are back-propagated to the parameter space of the hand

model by applying the chain rule. These gradients are noisy estimates of the true gradients from the true cost function. The gradient estimates are used by one of several stochastic optimisation algorithms to find the optimal pose.

Chapter 4

Stochastic Approximation

Measurement noise is an unavoidable phenomenon in any tracking system. Sources of noise include camera calibration noise, colour calibration noise and interestingly, noise due to point sampling. Point sampling noise occurs when the tracking system samples sparsely on the hand model surface to reduce computational load. As a result, the true cost (see Section 3.3) can never be evaluated by the tracking system in practice. Only a noisy approximate of the true cost can be measured.

Stochastic approximation theory [65, 8, 43, 87] provides a class of techniques for finding a root of a function $f(\mathbf{x})$ where only noise-corrupted measurements of the function values are available. This can be applied to an optimisation setting like the one in the tracker if we take $f(\mathbf{x})$ to be the gradient, $\nabla C_{\mathbf{x}^*}(\mathbf{x})$, of the tracker's true cost function $C_{\mathbf{x}^*}$. Then finding a root of f equates to finding a critical point (a minimum) of $C_{\mathbf{x}^*}$.

One can prove that $C_{\mathbf{x}^*}$ has a unique global minimum for most hand poses (see Section 4.5.1) and that the Hessian at the unique global minimum is positive definite (see Section 4.5.2). Furthermore the tracking system is shown to fit in the stochastic approximation framework. The tracking system locally satisfies the conditions of the Robbins-Monro method [65, 8]. The implication of this is that it is possible for the tracker estimate to converge locally to the optimal pose with probability 1 despite using gradient measurements that are noise-corrupted. The local convergence property is conditional on the noise statistics and the behaviour of the cost function. These conditions, and the extent to which they are satisfied, are explored in this chapter. Note that parts of Section 4.5.1 in this chapter have been published [21]. Section 4.5.2 contains new results that complete the proof for a well-behaved unique global minimum.

4.1 The Robbins-Monro Method

Recall from Section 3.5 in the last chapter that $\nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t)$ is the *gradient estimate* used by the optimisation algorithm to find the optimal pose $\mathbf{x}^*$ and is corrupted by noise, modelled as a random variable Φ. $\nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \phi_t)$ is modelled as the realisation of the random variable $\nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \Phi)$. Let a_t be the step size in our optimisation procedure. Then one possible iterative scheme for stochastic approximation is

$$\mathbf{X}_{t+1} = \mathbf{X}_t - a_t \nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \Phi), \tag{4.1}$$

where $\mathbf{X}_t$ is a random variable and $\mathbf{x}_t$ is the actual event of $\mathbf{X}_t$ at time t. The Robbins-Monro theory [65, 8] states that $\mathbf{X}_t$ converges to the optimal pose $\mathbf{x}^*$ in mean square and with probability 1, *i.e.*

$$\lim_{t\to\infty} E[(\mathbf{X}_t - \mathbf{x}^*)^2] = 0 \quad \text{and} \quad P(\lim_{t\to\infty} \mathbf{X}_t = \mathbf{x}^*) = 1, \tag{4.2}$$

if the following conditions are met:

1. A bound on the step size a_t, namely

$$\sum_{t=1}^{\infty} a_t = \infty, \sum_{t=1}^{\infty} a_t^2 < \infty. \tag{4.3}$$

2. $\nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \Phi)$ is unbiased, *i.e.*

$$E_\Phi[\nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \Phi)] = \nabla C_{\mathbf{x}^*}(\mathbf{x}_t). \tag{4.4}$$

3. $\nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \Phi)$ has uniformly bounded variance in the sense that

$$\sup\{Var(\nabla C_{\mathbf{x}^*}(\mathbf{x}_t, \Phi)) : \mathbf{x}_t \in \mathbb{R}^K\} < \infty. \tag{4.5}$$

4. $\nabla C_{\mathbf{x}^*}(\mathbf{x}_t)$ is well-behaved around $\mathbf{x}^*$ in the sense that

$$\inf\{(\mathbf{x} - \mathbf{x}^*)^{\mathrm{T}} \nabla C_{\mathbf{x}^*}(\mathbf{x}) : \epsilon < ||\mathbf{x} - \mathbf{x}^*|| < \epsilon^{-1}\} > 0, \tag{4.6}$$

for all $\epsilon \in \mathbb{R}, 0 < \epsilon < 1$.

The following sections demonstrate the extent to which the hand tracking system meets these requirements.

4.2 A Bound on the Step-size

Conceptually, the controlled decay of a_t serves to average out the noise as the iterative scheme progresses. The evolution of the step-sizes for the optimisation algorithms SGD (Section 3.5.1) and oBFGS (Section 3.5.3) can be proven to satisfy these bounds [87, 82].

The use of SGD in practice may be undesirable due to its slow convergence speed. Instead one can use oBFGS or SMD (Section 3.5.2), both of which have a faster rate of convergence. Although there is no convergence proof available yet for SMD, it has been shown empirically to converge well under noisy conditions in many practical situations. Prior applications of SMD to body/hand tracking work include [11, 42].

Note that the arguments for satisfying the remaining conditions (4.4), (4.5), (4.6) are independent of the chosen optimisation algorithm.

4.3 An Unbiased Gradient Estimate

Noise enters the tracking system via the real image gradients and the sparse point sampling scheme, resulting in noisy gradient measurements. Noise in the image gradients is largely uncontrollable. It derives from lens distortion, differing colour profiles between the two cameras, imperfect silhouette and the discretized approximation of the image gradients. The effects of lens distortion and differing colour profiles are minimised and assumed to be unbiased in practice by performing camera calibration with a checkerboard and a colour palette. Noise in the extracted silhouette is reduced via dilation and erosion techniques [75]. As for the approximation of the image gradients, this is done by applying a discrete Gaussian derivative filter [25] over the camera image. Noise due to this gradient approximation is unbiased since the filter is symmetric.

Sparse point sampling also introduces noise into the tracking system. However, this noise will be unbiased as the surface points are chosen via random sampling. So far, the cost function defined in Chapter 3 is ambiguous in the sense that the "ideal" distribution of surface points on the hand model has not been set. The distribution of surface points is important as it shapes the cost function*. The ideal distribution of surface points is taken to be the one that is most effective for the purposes of tracking. In practice, the sample points are

*Note however that a redistribution of sample points does not change the position of the unique global mininum in the cost function.

chosen from the vertices of the mesh. The distribution of the vertices on the mesh is taken to be an unbiased estimate of the ideal point distribution. This ideal point distribution is empricially determined. Chapter 5 details the effects of varying the distribution of vertices on the mesh.

4.4 A Uniformly Bounded Variance

The gradient estimate has a uniformly bounded variance: the variance in the Chamfer distance estimation process [9] for the noisy silhouette images is uniformly bounded by the size of the image. Therefore the variance in the gradient estimates generated by the silhouette and filling cost functions via finite differences will also be uniformly bounded. The variance due to the image gradients derived from the photo-consistency cost function is also uniformly bounded since the range of YUV values at each pixel is uniformly bounded.

4.5 A Well-behaved Minimum

Having a unique global minimum and a positive definite Hessian of $C_{\mathbf{x}^*}$ at $\mathbf{x}^*$ is sufficient to satisfy condition (4.6) locally. We first show that a unique global minimum exists at $\mathbf{x}^*$ for almost all hand poses. Exceptions to this will be highlighted. We will then prove that the Hessian at the unique global mininum is positive definite. The general assumptions made to facilitate both parts of the analysis are:

1. The y-axis of the two camera coordinate frames are parallel to each other. See Figure 3.1.

2. There is only one directional light source illuminating the scene from the front. See Figure 3.1.

3. The hand model has a Lambertian surface and has a uniform texture. Hence the YUV value of a point on the hand is completely determined by the surface normal and the light direction. Of course, a real hand exhibits additional texture derived from veins and pigments beneath the skin etc. But we argue that this additional structure only makes the choice of sample points for achieving a unique global minimum (Proposition 4.5.1) and a positive definite Hessian (Proposition 4.5.2) at $\mathbf{x}^*$ easier. Therefore the additional structure can be ignored in the analysis.

4. Cast shadows from the digit segments are ignored. Any surface neighbourhood that lies inside the cast shadow will have the same YUV value (namely black) under a single light source and therefore will be ineffective for realising Propositions 4.5.1 and 4.5.2. However, we argue that the boundary between the shadowed and the lighted region adds structure to the hand and will be informative enough to compensate for the lack of structure in the shadowed region. We also argue that the observable shadow-free regions of the hand are rich enough to realise Propositions 4.5.1 and 4.5.2.

5. For each segment of the hand, one can find at least a small region on the surface that is visible in both camera views. This assumption is used to exclude ambiguous poses for which a unique global minimum will never exist due to certain digit segments being completely hidden from the camera views. One example is a fisted hand with the back of the palm facing the cameras. In a practical HCI context, such occurrences are rare as most hand poses involve the palm facing towards the cameras.

Additional assumptions that are specific to the first part of the analysis only (showing a unique global minimum at $\mathbf{x}^*$ in Section 4.5.1) are

6. The palm is modelled by a rectangular cuboid and the digits of the hand are modelled by chains of cylinders. Our proof of Proposition 4.5.1 for a unique global minimum will rely on a suitable choice of sample points. We argue that this choice only becomes easier for a hand model that has a more structured or irregular surface. Hence the proof applies *a fortiori* to our hand model used in tracking (Figure 3.3).

7. The baseline of the camera pair is longer than the length of any given digit segment.

Additional assumptions that are specific to the second part of the analysis only (showing a positive definite Hessian at $\mathbf{x}^*$ in Section 4.5.2) are

8. Without loss of generality, we restrict the analysis of the photo-consistency cost function in the proof to just the Y channel for convenience, ignoring the U and V channels. This is valid since C_p can be decomposed as a linear combination of costs from the Y, U and V channels respectively. The inclusion of U and V channels will only make it easier to achieve a positive definite Hessian at $\mathbf{x}^*$ (*cf.* Section 4.5.2).

9. Assume that both cameras share the same calibration/projection matrix K which has the form

$$K := \begin{pmatrix} \alpha_x & 0 & o_x \\ 0 & \alpha_y & o_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.7}$$

Without loss of generality, we assume for convenience that K is the identity, *i.e.* K has a unity focal length for both image axes and its principal point at (0,0). Having a non-unity focal length merely scales the problem and does not affect the overall proof. Setting the principal point to (0,0) is valid as the principal point is just an arbitrary shift in the coordinate frame of the image plane.

10. We assume that the hand surface can be approximated well by sections of quadric C^2 surfaces stitched together *e.g.* spherical surfaces for the knuckles and fingertips of the hand, and ellipsoids for sides of the palm, and cylindrical surfaces for the digit segments. Note that this assumption is less restrictive than Assumption 6 that is used for proving a unique global minimum.

4.5.1 A Unique Global Minimum of $C_{\mathbf{x}^*}$ at $\mathbf{x}^*$

A unique global minimum for $C_{\mathbf{x}^*}$ does not exist for all $\mathbf{x}^*$, *i.e.* every possible hand pose observed in the pair of real camera images. Nevertheless, we will show that for a substantial subset of the possible $\mathbf{x}^*$, there is always a unique global minimum at $\mathbf{x}^*$. We use the term 'substantial' to mean that the exceptions can be described by a finite set of (not necessarily polynomial) equations.

Proposition 4.5.1. *The cost at the optimal position* $C_{\mathbf{x}^*}(\mathbf{x}^*) = 0$. *Under Assumptions 1 to 7, perturbing* $\mathbf{x}^*$ *to* $\mathbf{x} \neq \mathbf{x}^*$ *strictly increases* $C_{\mathbf{x}^*}(\mathbf{x})$.[†]

The first part of the Proposition is obvious, because at the optimal position all the sample points lie within the silhouette and for each point on the Lambertian surface of the hand model, the observed YUV value from the different camera views will be the same. In the latter part, perturbing $\mathbf{x}^*$ will cause certain parts of the hand to move. We denote the points on the hand surface affected by this

[†]Note that Proposition 4.5.1 does not preclude the existence of other stationary points in the cost function.

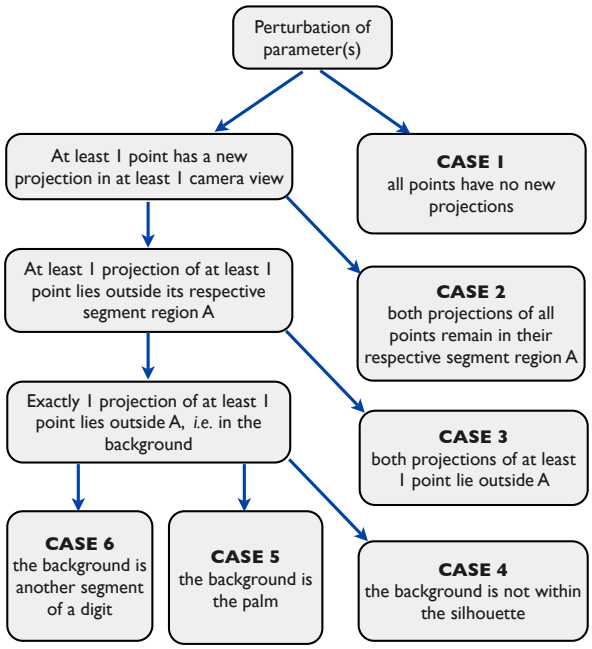

Figure 4.1: All the possible scenarios that can occur for the active points under a perturbation of parameters. Note that A denotes the projected area of the segment (at the optimal hand pose $\mathbf{x}^*$) that an active point belongs to.

perturbation as 'active points'. For Proposition 4.5.1 to be true at least one of these active points must cause the cost to increase.

In the analysis the hand will be divided into 16 segments based on the hand joints, *i.e.* 1 segment for the palm and 3 segments for each digit. Each active point belongs to a segment. Let A_j^i be the 'original' projected area (in the jth camera view) of the segment that the ith active point belongs to, at the optimal hand pose $\mathbf{x}^*$. Then Figure 4.1 shows the tree of possibilities that can occur to the projections of the active points. We can classify the set of active points into subsets based on these six cases in the tree. Each subset will be individually examined as to whether it will cause $C_{\mathbf{x}^*}$ to increase after a perturbation. Exceptions where the subset does not cause the cost to increase are highlighted at the end of each case. Note that Proposition 4.5.1 fails to hold only if *none* of the six subsets increase $C_{\mathbf{x}^*}$ upon a perturbation.

Case 1 (All points have no new projections)

Eight points that lie in a non-degenerate configuration in Euclidean space uniquely define the epipolar geometry of the camera pair [29]. Conversely, a known epipo-

lar geometry uniquely defines the projections of eight points that belong to a non-degenerate configuration. Let γ be a set of eight points in a non-degenerate configuration, chosen from the set of active points on a rigid segment of the hand. This is always possible as the set of active points is dense. Then, a perturbation will move at least one of the eight points in γ. Thus Case 1 cannot occur.

Case 2 (Both projections of all points remain on their original segment)

We ignore the trivial example of a cylindrical segment rotating around the main cylindrical axis as this type of movement is not possible for the digits of the hand without making the palm rotate, which in turn causes other digits to move outside their original positions.

For a cylindrical segment to lie inside the original region of a given camera view after perturbation, it can only move in a conic region of the plane spanned by the end points of the cylinder and the camera's optical centre. Given that there are two cameras (see Figure 4.2), the intersection of the two conic planes is the only region where movement is allowed.

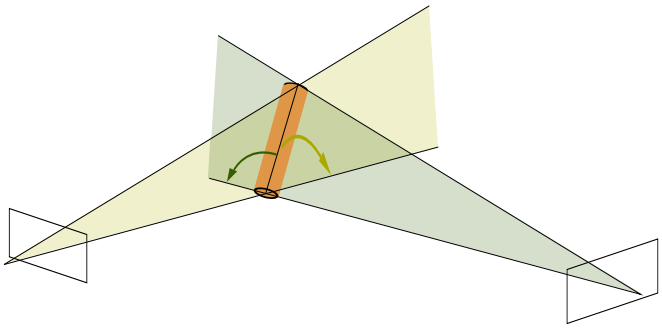

Figure 4.2: Foreshortening of a cylindrical segment when the main axis does not lie on the epipolar plane.

This intersection specifies the position of the cylinder uniquely unless the conic regions lie on the same plane, namely the epipolar plane spanned by one end of the cylinder (see Figure 4.3, right). If the cylinder's main axis lies on this plane, then movements on the plane can cause the resulting projection to lie within the original segment for both cameras. For convenience, we shall denote this set of movements as κ.

Pure translational movements on the plane belong to κ (see Figure 4.3, left) only if the projection of the cylindrical segment is longer than the baseline of the camera pair. The baseline is much longer than any of the segments of the

hand in our setup (as per Assumption 7), so pure translational movements can be ignored.

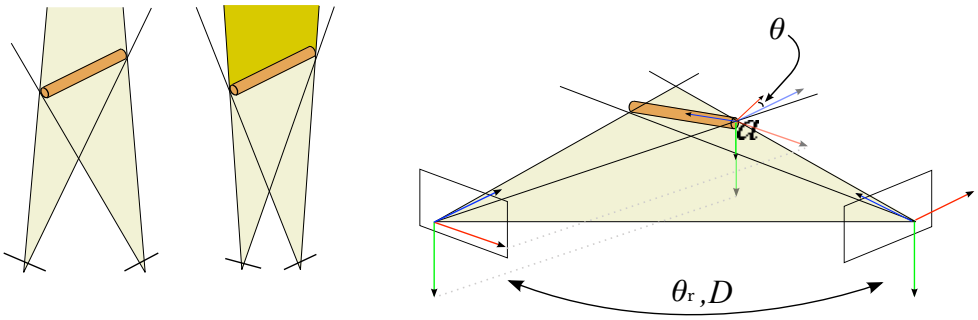

Figure 4.3: Left: A cylinder undergoing a pure translation violates the condition for Case 2, unless the camera baseline is shorter than the length of the cylinder (*e.g.* 2nd diagram from the left). The dark yellow area indicates the region where the cylinder can translate to. Right: Foreshortening in both cameras when the cylindrical segment rotates on the epipolar plane spanned by a, the centre of rotation.

A combination of rotational and translational movements on the plane belong to κ if the projection of the cylinder's main axis to the camera image plane is shorter in both cameras after the rotational movement, and the translation movement only moves the perturbed segment within the original region.

Without loss of generality, we take the epipolar plane to be the plane spanned by the x and z axis in the world coordinates. Let θ be the original angle of rotation prior to the perturbation (see Figure 4.3). Then one can show that for $C_{\mathbf{x}^*}$ not to increase after a perturbation,

$$\theta = tan^{-1}(-\frac{D_z}{D_x}) - \theta_r, \tag{4.8}$$

where θ_r, the rotation angle, and D, the translation vector, are the transformation parameters that convert points from the local coordinates of camera 1 to camera 2 (See Appendix A for details). Hence (4.8) is the only choice for θ (and consequently $\mathbf{x}^*$) that might not cause $C_{\mathbf{x}^*}$ to increase.

The same argument can be applied to the palm, as the palm is attached to the digits, which are cylindrical chains. However, this ambiguity for the palm can only occur if additionally a) the palm and the digits all lie on the epipolar plane or b) all digits of the hand are touching their adjacent digits to form a convex shape. Condition b) ensures that there are no gaps between the fingers that would otherwise lead to Case 4 when movement occurs on the epipolar plane.

Case 3 (Both projections lie outside of the original segment)

Using a simple continuity argument, one can show that it is impossible to move a hand segment such that its new projection lies completely outside of its original projection at $\mathbf{x}^*$ without causing other segments along the kinematic chain to partially leave their original projections or to leave the silhouette. Therefore one can use the arguments in Cases 4, 5 or 6 for the active points on the adjoining segments to show that $C_{\mathbf{x}^*}$ increases.

Case 4 (One of the projections lies outside of the silhouette)

If one of the active point projections lies outside the original segment and falls outside the silhouette region, then $C_{\mathbf{x}^*}$ increases due to the silhouette cost function C_s.

Case 5 (One of the projections lies on the palm)

Let p_1, p_2 be the projections of an active point p in the two camera views. p_1 is projected to a surface point s_c on the original cylindrical segment while p_2 is projected onto a surface point s_p on the palm. Suppose the YUV value at p_1 is the same as in p_2, which implies that the surface normals n_p and n_c at s_p and s_c respectively are equidistant to the light direction l. Then this point will not increase $C_{\mathbf{x}^*}$.

However note that a reasonably-sized neighbourhood of p will also be projected onto the palm. This statement is valid since the set of all sample points on a segment of the hand surface is closed and thus its projection into the image plane is also closed with a non-trivial interior. We can always choose a p' from this neighbourhood such that the surface normal at s'_c and s_c are not equidistant to the light direction. Since the palm is locally planar, it has a constant surface normal. Therefore the colour intensity at p'_1 will be different to p'_2, and so p' increases $C_{\mathbf{x}^*}$.

Ambiguity occurs when the YUV values at p_1 and p_2 are completely black *i.e.* when both $n_p \cdot l \geq 0$ and $n_c \cdot l \geq 0$. In this situation the neighbourhood of s_p (on the palm) will be completely black. Similarly there will be a closed set of points with different surface normals at the neighbourhood of s_c (on the cylinder) that is completely black, due to the inequality in the lighting equation $n_c \cdot l \geq 0$. Therefore it becomes unclear whether the active points in the neighbourhood of p will increase $C_{\mathbf{x}^*}$.

Let γ be the entire set of active points belonging to Case 5. In light of the above ambiguity, γ will not increase $C_{\mathbf{x}^*}$ only if the YUV values at the projections of *all* active points in γ are black. This will not occur if we have a frontal light source, as per Assumption 2.

Case 6 (One of the projections lies on another digit segment)

For almost all $\mathbf{x}^*$, the set of active points that belong to this case will increase $C_{\mathbf{x}^*}$ upon a perturbation. Let p_1, p_2 be the projections of an active point p in the camera views 1 and 2 respectively. p_1 is projected to a surface point s_c on the original cylindrical segment C while p_2 is projected onto a surface point s_d of another cylindrical segment D (see Figure 4.4, left). Both projections of p have the same YUV value, which we will denote as η.

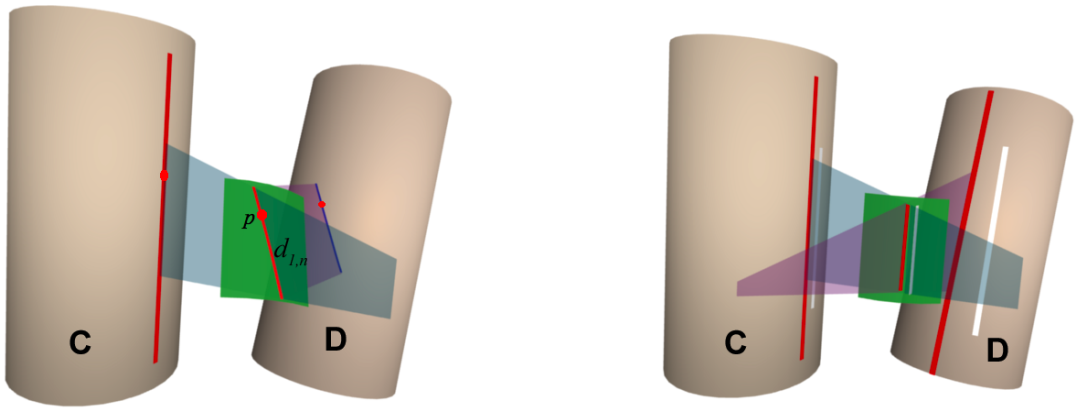

Figure 4.4: Left: The green surface is a patch of the hand segment that has been perturbated from $\mathbf{x}^*$. An active point p lies on a trajectory $d_{1,\eta}$ (marked in red). The projection of $d_{1,\eta}$ onto cylinder C lies on a line with a constant YUV value. The projection of $d_{1,\eta}$ on cylinder D however will almost always lie on a curve (marked in blue) with a non constant YUV value. **Right:** A perturbation where both projections of $d_{1,n}$ have the same YUV value. One can always choose active points from a neighbouring trajectory d' (marked in white) to increase $C_{\mathbf{x}^*}$, due to the forshortening effect of cylinder D observed from camera 2.

One can trace a trajectory $d_{1,\eta}$ in the neighbourhood of p (see Figure 4.4, Left) where the projections of the active points on $d_{1,\eta}$ in camera view 1 have the same YUV value, η. The projection of $d_{1,\eta}$ in camera view 1 is a line on the cylindrical surface where the surface normal is constant. However the projection of $d_{1,\eta}$ in camera view 2 will not lie on the equivalent line on the other cylinder. Hence the

set of active points on $d_{1,\eta}$ will increase $C_{\mathbf{x}^*}$ for this particular perturbation. This is not enough to satisfy Proposition 4.5.1, which requires the above statement to be true for *all* perturbations where $\mathbf{x} \neq \mathbf{x}^*$. Unfortunately, there will always be one perturbation where this is false (see Figure 4.4, Right). Under this perturbation, $d_{1,\eta}$ on the perturbed cylinder is aligned to the intersection of the two planes whose line projections (in the appropriate camera views) have the same YUV value η. Here, the active points on $d_{1,\eta}$ will not increase $C_{\mathbf{x}^*}$. However, suppose we choose another trajectory d' that is parallel to $d_{1,\eta}$ on the projected cylinder. The projection of d' will have a constant YUV value in camera 1 but not in camera 2, due to the foreshortening effect present in camera 2 but not camera 1. Hence $C_{\mathbf{x}^*}$ will increase due to active points belonging to d'.

This result leads us to an exception set of $\mathbf{x}^*$, denoted as ε, where the set of active points belonging to this case will not increase $C_{\mathbf{x}^*}$. Firstly, let us define Υ, the set of $\mathbf{x}^*$ with the following property : the pose contains (at least) two digit segments C and D exhibiting the same foreshortening effect in camera 1 and 2 respectively, such that there exists a perturbation δ where the trajectory $d_{1,\eta}$ and its neighbourhood of parallel trajectories d' all project to 'equivalent' line segment patterns in both camera views. Figure 4.5 is an example of one such $\mathbf{x}^*$ that belongs to Υ.

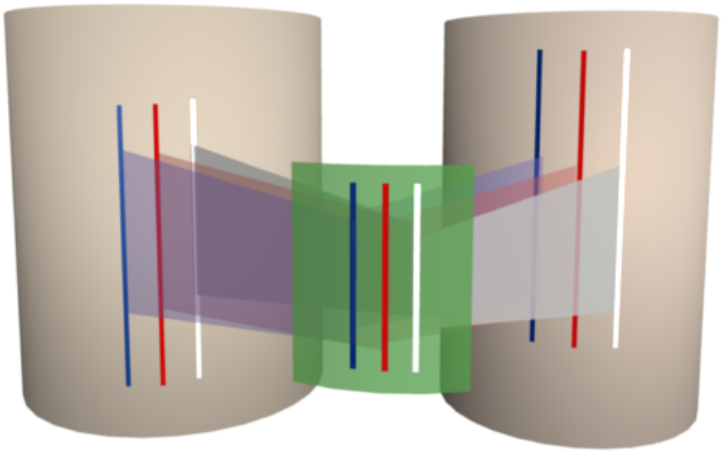

Figure 4.5: $\mathbf{x}^*$ belongs to Υ if $d_{1,\eta}$ and its neighbourhood of parallel trajectories d' all project to 'equivalent' line segment patterns in both camera views.

Recall that s_c and s_d are the projections of the active point p (belonging to $d_{1,\eta}$) on the surface of cylinders C and D respectively. The exception set ε is a subset of Υ, where the surface normals at s_c and s_d are the same under the

perturbation δ.

If the surface normals at s_c and s_d are different[‡], then it is always possible to increase $C_{\mathbf{x}^*}$ by choosing another active point in the neighbourhood of p. To illustrate this, we take a simple pose belonging to Υ where the main axes of cylinders C and D are parallel to the y-axes of both cameras (see Figure 4.6).

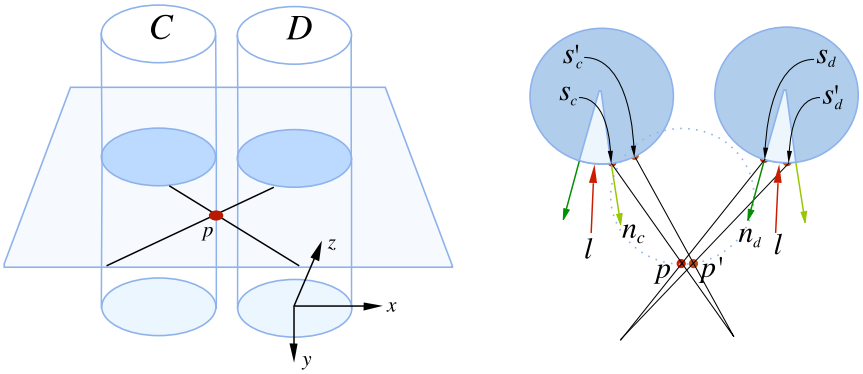

Figure 4.6: Left: The setup of a simple situation where $\mathbf{x}^*$ belongs to Υ. **Right:** The cross-section of the setup. l indicates the projected light direction. The dotted circle indicates the perturbed cylinder. The lighter regions are parts where the YUV value is greater than u.

p_1 and p_2 (and thus s_c and s_d) have the same YUV value η since the light direction is equidistant to n_c and n_d. This produces the lighting pattern as seen in Figure 4.6. Points to the right (anticlockwise) of s_c on the cylinder are darker than u while points to the left are lighter than u. This pattern is reversed on d, where points to the right of s_d are lighter than u. One can choose another point p' to the right of p on the projected cylinder such that p'_1 lands on the lighter region while p'_2 lands on the darker region, thereby increasing $C_{\mathbf{x}^*}$.

Conversely, if the surface normals at s_c and s_d are the same in this situation, then the reverse lighting pattern argument does not hold. Since the lighting patterns match and the projections of the neighbourhood of p match, active points in this neighbourhood will not cause $C_{\mathbf{x}^*}$ to increase.

Note that the occurrence of $\mathbf{x}^*$ belonging to the exception set is rare. Thus far we have only treated the hand segments as discrete rigid bodies. It is likely that the set ε will shrink further after accounting for restrictions due to the structural configuration of the hand. Finally it is still possible for $C_{\mathbf{x}^*}$ to increase even if the

[‡]surface points with different surface normals can have the same YUV value as long as the angles between the surface normals and the light source are the same

pose $\mathbf{x}^*$ being considered belongs to ε, as long as the active points belonging to the other cases cause $C_{\mathbf{x}^*}$ to increase. Given the articulated nature of the hand, it is extremely likely that the perturbation δ (for a given $\mathbf{x}^* \in \varepsilon$) will cause other cases in the possibility tree to be violated, thereby increasing $C_{\mathbf{x}^*}$.

This completes the proof of Proposition 4.5.1. The next subsection will address the positivity of the Hessian at the unique global mininum.

4.5.2 The Positivity of the Hessian of $C_{\mathbf{x}^*}$ at $\mathbf{x}^*$

For cases where a unique global minimum exists, we now show analytically that the cost function has a positive definite Hessian at the unique global minimum. This is sufficient to satisfy condition (4.6) locally.

Proposition 4.5.2. *Under Assumptions 1 to 5 and 8 to 10, the Hessian of $C_{\mathbf{x}^*}$ at $\mathbf{x}^*$ is postive definite for cases where a unique global minimum exists.*

We first rewrite the overall cost function as

$$C_{\mathbf{x}^*}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} C \circ M_i(\mathbf{x}), \tag{4.9}$$

where C is the cost function with respect to a set of visual cues and M is the function that evaluates the set of visual cues for the ith sample point (see next Section 4.5.3 for details on the derivation). Because $C_{\mathbf{x}^*}(\mathbf{x}^*) = 0$ (see Proposition 4.5.1), the Hessian H of $C_{\mathbf{x}^*}$ at $\mathbf{x}^*$ can be rewritten [70] as

$$H = \frac{1}{N}\sum_{i=1}^{N} {J_{M_i}}^{\mathrm{T}} H_c J_{M_i}, \tag{4.10}$$

where H_c is the Hessian of C and J_{M_i} is the Jacobian of M for the ith sample point. Each of the summands in (4.10) is at least positive semi-definite. Thus achieving a positive definite H is equivalent to adding enough summands in (4.10) such that H becomes a full rank matrix.

H_c is the sum of the Hessian of the photo-consistency cost function H_{c_p}, the Hessian of the silhouette cost function H_{c_s}, and the Hessian of the filling cost function H_{c_f}. Note that

$$\mathrm{rank}(H) \geq \mathrm{rank}(\hat{H}), \text{ where } \hat{H} = \frac{1}{N}\sum_{i=1}^{N} {J_{M_i}}^{\mathrm{T}} H_{c_p} J_{M_i}. \tag{4.11}$$

Showing that $\hat{H}$ has full rank (and therefore is positive definite) at the minimum is sufficient to satisfy condition (4.6) locally. The analysis will explore a constellation of sample points on the hand model surface that makes $\hat{H}$ (and consequently H) a full rank matrix.

4.5.3 The Structure of $\hat{H}$

In this section we show that one can rearrange the expression for $\hat{H}$ as a sum of matrices generated by the outer product of a set of vectors ϕ_i, *i.e.*

$$\hat{H} = \frac{1}{N}\sum_{i=1}^{N} \phi_i {\phi_i}^{\mathrm{T}}. \tag{4.12}$$

Then proving that $\hat{H}$ has full rank is equivalent to finding a linear independent set of ϕ_i. To see why this is so, consider the following lemma.

Lemma 4.5.3. *Let $\phi_i \in \mathbb{R}^m, i = 1, ..., m$ be a set of linearly independent vectors. Then the matrix*

$$A := \sum_{i=1}^{m} \phi_i {\phi_i}^{T} \tag{4.13}$$

is a full rank matrix.

Proof. Let $x \in \mathbb{R}^m$. Expanding Ax gives

$$\begin{aligned} Ax &= \sum_{i=1}^{m} \phi_i {\phi_i}^{\mathrm{T}} x & (4.14)\\ &= \phi_1({\phi_1}^{\mathrm{T}}x) + ... + \phi_m({\phi_m}^{\mathrm{T}}x). & (4.15) \end{aligned}$$

Rearranging gives

$$Ax = M(M^{\mathrm{T}}x) \tag{4.16}$$

where

$$M := \begin{bmatrix} \phi_1 \dots \phi_m \end{bmatrix}. \tag{4.17}$$

M has full rank since the set of ϕ_i's are linearly independent. Therefore $A = MM^{\mathrm{T}}$ will also have full rank. □

We now examine how $\hat{H}$ can be expressed in terms of the vectors ϕ_i. The Hessian of the photo-consistency cost function, H_{c_p}, in $\hat{H}$ is

$$H_{c_p} = \begin{bmatrix} I_{3\times3} & -I_{3\times3} \\ -I_{3\times3} & I_{3\times3} \end{bmatrix}. \tag{4.18}$$

Assumption 8 simplifies H_{c_p} to

$$H_{c_p} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{4.19}$$

J_{M_i} is the Jacobian of $M_i(\mathbf{x})$. $M_i(\mathbf{x})$ is the function that takes the ith sample point, ω_i, on the surface of a hand segment as input and evaluates the intensity value (*i.e.* the Y channel) at the sample point projection in both camera views. We now examine the structure of J_{M_i}.

Let $\Gamma_i(\omega_i, \mathbf{x})$ be a chain of Euclidean transformations that transforms the sample point ω_i on a hand segment b to the world coordinates. For example, suppose a sample point ω_i is chosen from the palm. Let $T_h^g(\omega_i, \mathbf{x})$ denote the Euclidean transformation that transforms ω_i in the h coordinate system to the g coordinate system. Then

$$\Gamma_i(\omega_i, \mathbf{x}) = T^0_{\text{palm}}(\omega_i, \mathbf{x}). \tag{4.20}$$

The dependency on $\mathbf{x}$ comes from the fact that the rotation and translation parameters governing T^0_{palm} are elements of $\mathbf{x}$. If ω_i belongs to another hand segment, say the DIP joint of the index finger, then

$$\Gamma_i(\omega_i, \mathbf{x}) = T^0_{\text{palm}} \circ T^{\text{palm}}_{\text{PIP}} \circ T^{\text{PIP}}_{\text{DIP}}(\omega_i, \mathbf{x}). \tag{4.21}$$

Let T_0^j denote the Euclidean transformation that transforms a point in the world coordinate frame to the coordinate frame of the jth camera. T_0^j can be written out in full as

$$T_0^j(\omega_i) := R_0^j \omega_i + t_0^j, \tag{4.22}$$

where R_b^0 is a 3×3 rotation matrix and t_b^0 is a translation vector. Also, let K_j be the projection function that maps a point in the world coordinates to the image plane of camera j

$$K_j(p) := \begin{bmatrix} \alpha_{j,x} & 0 & o_{j,x} \\ 0 & \alpha_{j,y} & o_{j,y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \tag{4.23}$$

$\alpha_{j,x}$ and $\alpha_{j,y}$ are the focal length parameters and $o_{j,x}$ and $o_{j,y}$ are the coordinates of the camera's principal point. Via assumption 9, K_j is taken without loss of generality to be the identity function, *i.e.* $\alpha_{j,x} = \alpha_{j,y} = 1$ and $o_{j,x} = o_{j,y} = 0$. Finally, let D_j be the depth normalisation function,

$$D_j(k_j) := \begin{bmatrix} \frac{1}{k_{j,z}} & 0 & 0 \\ 0 & \frac{1}{k_{j,z}} & 0 \end{bmatrix} \begin{bmatrix} k_{j,x} \\ k_{j,y} \\ k_{j,z} \end{bmatrix}. \tag{4.24}$$

Then M_i can be expressed as

$$M_i(\mathbf{x}) = \begin{bmatrix} M_{i,1}(\mathbf{x}) \\ M_{i,2}(\mathbf{x}) \end{bmatrix}, \tag{4.25}$$

where $M_{i,j}(\mathbf{x})$ is

$$M_{i,j}(\mathbf{x}) := I_j \circ D_j \circ K_j \circ T_0^j \circ \Gamma_i(\omega_i, \mathbf{x}). \tag{4.26}$$

Note that I_j is the function that evaluates the intensity value at the pixel of the sample point projection in the jth camera view. I_j is used for evaluating the photo-consistency cost function (*cf.* Section 3.3.3). For convenience in the analysis, let $A_j := I_j \circ D_j \circ K_j \circ T_0^j$ and express $M_{i,j}(\mathbf{x})$ as

$$M_{i,j}(\mathbf{x}) = A_j \circ \Gamma_i(\omega_i, \mathbf{x}). \tag{4.27}$$

This split in (4.27) allows us to separate parts of the function $M_i(\mathbf{x})$ that are dependent on the geometry of the camera setup (*i.e.* A_j) from the model-parameter-dependent parts. Let $h_i := \Gamma_i(\omega_i, \mathbf{x})$, $p_j := T_0^j(h_i)$, $k_j := K_j(p)$, $d_j := D_j(k_j)$. Then the Jacobian of $M_i(\mathbf{x})$ can be written as

$$J_{M_i}(\mathbf{x}) = J_A(h_i) J_{\Gamma_i}(\mathbf{x}) =: \begin{bmatrix} Y_{i,1} \\ Y_{i,2} \end{bmatrix} J_{\Gamma_i}(\mathbf{x}), \tag{4.28}$$

where $Y_{i,k} \in \mathbb{R}^{1\times 3}$ is defined as

$$\begin{aligned} Y_{i,j} &:= \nabla(I_j \circ D_j \circ K_j \circ T_0^j(h_i)) && (4.29)\\ &= \nabla I_j(d_j) J_{D_j}(k_j) J_{K_j}(p_j) J_{T_0^j}(h_i) && (4.30)\\ &= \nabla I_j(d_j) {F_j}^{\mathrm{T}} \mathbf{I} R_0^j && (4.31)\\ &= \nabla I_j(d_j) {F_j}^{\mathrm{T}} R_0^j, && (4.32) \end{aligned}$$

with F_j being

$$F_j := \begin{bmatrix} \frac{1}{k_{j,z}} & 0 \\ 0 & \frac{1}{k_{j,z}} \\ -\frac{k_{j,x}}{k_{j,z}^2} & -\frac{k_{j,y}}{k_{j,z}^2} \end{bmatrix}. \tag{4.33}$$

Note that $J_{\Gamma_i}(\omega_i) \in \mathbb{R}^{3\times m}$, where m is the number of model parameters in $\mathbf{x}$. Finally, let $\triangle Y_i := Y_{i,1} - Y_{i,2}$, and define $\phi_i \in \mathbb{R}^m$ as

$$\phi_i := (\triangle Y_i J_{\Gamma_i}(\mathbf{x}))^{\mathrm{T}}, \tag{4.34}$$

then one can rewrite $\hat{H}$ as

$$\hat{H} = \frac{1}{N} \sum_{i=1}^{N} \phi_i {\phi_i}^{\mathrm{T}}. \tag{4.35}$$

To prove Proposition 4.5.2, we show in the following sections how one can achieve a linear independent set of ϕ_i's by choosing a suitable constellation of sample points on the hand model surface. First, we just examine the palm of the hand (ignoring the fingers) as a single rigid body exhibiting rotational and translational movements (6 DOFs), and show that the corresponding $\hat{H}$ for the 6 DOF palm can achieve full rank with a suitable choice of sample points. Afterwards we extend the proof in a straightforward manner to an articulated body, *i.e.* the rest of the hand with movable joints.

4.5.4 A Positive Definite $\hat{H}$ for a Rigid Body

Six degrees of freedom parameterise the rotation and translation movement of a rigid body and thus $\phi_i \in \mathbb{R}^6$ in this instance. As an overview, we show how one can find a linearly independent set of $\Psi := \{\phi_1, ..., \phi_6\}$ (required for the positivity of $\hat{H}$) by choosing an appropriate corresponding $\{\omega_1, ..., \omega_6\}$ set on a rigid body. Specifically Ψ_i has the following structure:

$$\Psi := \{\phi_1, ..., \phi_6\} = \{ \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \end{bmatrix} \}, \tag{4.36}$$

where $*$ denotes some unknown non-zero real value and $\delta << 1$.

The proof of Ψ being a linearly independent set is as follows; we initially show how one can pick a set of sample points, $\{\omega_1, \omega_2, \omega_3\}$, on the rigid body such that the corresponding set of ϕ_i's (*i.e* the ϕ_1, ϕ_2, ϕ_3 entries in Ψ) are linearly independent. Choosing these sample points will increase the rank of $\hat{H}$ to 3. Then we choose another set of sample points, $\{\omega_4, \omega_5\}$, whose corresponding ϕ_i's (*i.e* the ϕ_4, ϕ_5 entries in Ψ) can be shown to be linearly independent of each other based solely on the 4th and 6th entries of the ϕ_i's. Adding $\{\omega_4, \omega_5\}$ to the previous set increases the rank of $\hat{H}$ to 5. Finally, we show how one can choose a sample point ω_6 such that its corresponding ϕ has the structural form of ϕ_6 in Ψ. Adding ω_6 completes the Ψ set, promoting the rank of $\hat{H}$ to 6.

Decomposing ϕ_i

To appreciate how one can obtain the ϕ_i's in Ψ, the structure of ϕ_i as a function of the corresponding sample point ω_i needs to be examined. Recall from Equation (4.34) that

$$\phi_i := (\triangle Y_i J_{\Gamma_i}(\mathbf{x}))^{\mathrm{T}}. \tag{4.37}$$

There are 6 degrees of freedom for a rigid body - 3 DOF for the translation vector $t \in \mathbb{R}^3$ and 3 Euler angles, $\theta_x, \theta_y, \theta_z$, that parameterise the rotation matrix. Note that the model parameter vector $\mathbf{x}^*$ for a rigid body is defined as

$$\mathbf{x} := [t_x, t_y, t_z, \theta_x, \theta_y, \theta_z]. \tag{4.38}$$

Therefore, $J_{\Gamma_i}(\mathbf{x})$ is a $\mathbb{R}^{3\times 6}$ matrix. Each column of J_{Γ_i} represents the partial derivative with respect to a particular model parameter, *i.e.*

$$J_{\Gamma_i}(\mathbf{x}) = [\frac{\partial \Gamma_i}{\partial t_x}, \frac{\partial \Gamma_i}{\partial t_y}, \frac{\partial \Gamma_i}{\partial t_z}, \frac{\partial \Gamma_i}{\partial \theta_x}, \frac{\partial \Gamma_i}{\partial \theta_y}, \frac{\partial \Gamma_i}{\partial \theta_z}]. \tag{4.39}$$

It can be shown that

$$J_{\Gamma_i}(\mathbf{x}) = [\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \frac{\partial \Gamma_i}{\partial \theta_x}, \frac{\partial \Gamma_i}{\partial \theta_y}, \frac{\partial \Gamma_i}{\partial \theta_z}]. \tag{4.40}$$

The structure of $\frac{\partial \Gamma_i}{\partial \theta_x}, \frac{\partial \Gamma_i}{\partial \theta_y}, \frac{\partial \Gamma_i}{\partial \theta_z}$ depends on the coordinate system used to parameterise the rotation matrix. Without loss of generality, we choose to parameterise the coordinate system for the rotation matrix to be the coordinate system of camera 1 that has been shifted by a translation vector t_0^1 (see Figure 4.7). Under this coordinate system (denoted as $\tilde{C}$), $J_{\Gamma_i}(\mathbf{x})$ becomes

$$J_{\Gamma_i}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & 0 & \omega_{i,z} & -\omega_{i,y} \\ 0 & 1 & 0 & -\omega_{i,z} & 0 & \omega_{i,x} \\ 0 & 0 & 1 & \omega_{i,y} & -\omega_{i,x} & 0 \end{bmatrix}. \tag{4.41}$$

Choosing this coordinate system does not result in a loss of generality and is possible for all poses $\mathbf{x}^*$ by pre-multiplying sample points in the 3D-2D pipeline by an offset rotation matrix. The purpose of the coordinate change is to make the analysis easier. See Appendix B for further details on this issue and how Equation (4.41) is derived. Note that from here onwards, ω_i is taken to be expressed in the $\tilde{C}$ coordinate system.

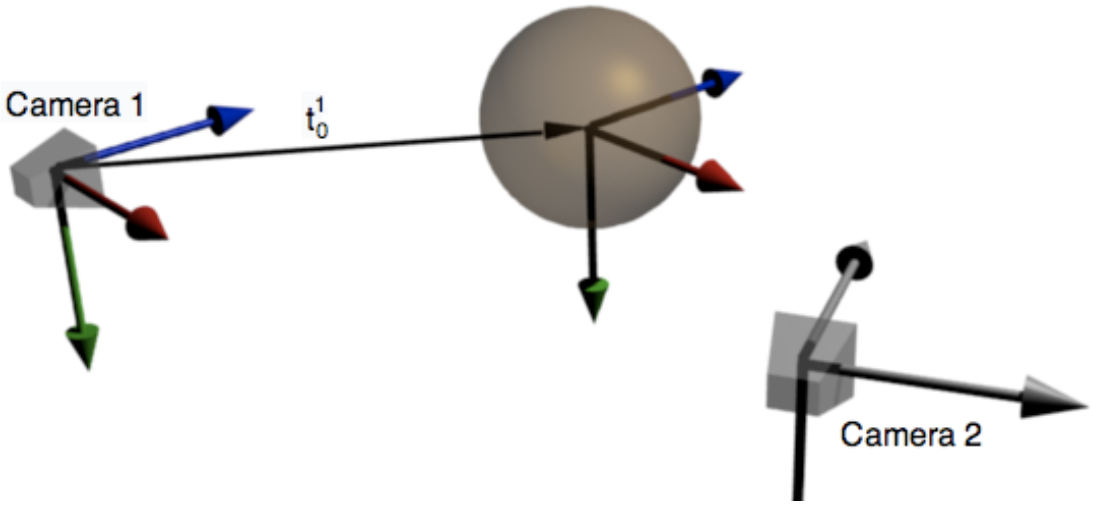

Figure 4.7: The coordinate system, $\tilde{C}$, of the rigid body is aligned with the coordinate system of camera 1.

We now examine the structure of $\triangle Y_i$. Using Equation (4.32), $\triangle Y_i$ can be expanded out to

$$\begin{aligned} \triangle Y_i &= Y_{i,1} - Y_{i,2} && (4.42) \\ &= \nabla I_1(d_1) {F_1}^{\mathrm{T}} R_0^1 - \nabla I_2(d_2) {F_2}^{\mathrm{T}} R_0^2. && (4.43) \end{aligned}$$

Note that d_j is dependent on the ith sample point. We drop this dependence from our notation for the sake of simplicity. $\nabla I_j(d_j)$ is the intensity gradient observed at the pixel coordinates of d_j of the projection of ω_i on the image plane of camera j. It is the projection of the "3D colour intensity gradient" observed on the surface of the hand. To be precise, the 3D colour intensity gradient is a surface tangent at ω_i whose direction indicates the greatest change in colour intensity. How $\nabla I_j(d_j)$ is derived from the 3D colour intensity gradient is explained below.

Derivation of $\nabla I_j(d_j)$

Let $\tilde{Q} : \zeta \subset \mathbb{R}^3 \rightarrow \mathbb{R}$ be the mapping from the surface of the hand, ζ, to an intensity value and $I_j : \mathbb{R}^2 \rightarrow \mathbb{R}$ be the mapping from the image coordinates $d_j = (u_j, v_j)$ in camera j to an intensity value. Then I_j is related to $\tilde{Q}$ via

$$I_j(d_j) = I_j(u_j, v_j) = \tilde{Q} \circ (T_0^j)^{-1} \circ K^{-1} \circ D_{\zeta,j}^{-1}(u_j, v_j). \quad (4.44)$$

Note that the inverse projection mapping $K^{-1} \circ D_{\zeta,j}^{-1}$ is unique since the camera can only view the side of the hand that is facing towards it. Once again note that $I_j(d_j)$ is dependent on the ith sample point, since the image coordinates d_j depend on ω_i. Let $\tilde{d}_j := D_{\zeta,j}^{-1}(u, v), \tilde{k}_j := K^{-1}(\tilde{d}_j)$, $\tilde{p} := (T_0^j)^{-1}(\tilde{k}_j)$. Therefore,

$$\nabla I_j(d_j) = \nabla \tilde{Q}(\tilde{p}) J_{(T_0^j)^{-1}}(\tilde{k}_j) J_{K^{-1}}(\tilde{d}_j) J_{D_{\zeta,j}^{-1}}(u_j, v_j). \tag{4.45}$$

$$= \nabla \tilde{Q}(\tilde{p}) (R_0^j)^{\mathrm{T}} K^{-1} \begin{bmatrix} \frac{\partial D_{\zeta,j}^{-1}}{\partial u_j} & 0 \\ 0 & \frac{\partial D_{\zeta,j}^{-1}}{\partial v_j} \\ 0 & 0 \end{bmatrix}. \tag{4.46}$$

Note that $\tilde{p}$ is in fact ω_i and that $\nabla \tilde{Q}(\tilde{p}) \in \mathbb{R}^3$ is the gradient of the colour intensity at the point $\tilde{p}$, *i.e* the 3D colour intensity gradient at ω_i. For convenience let $g_i := \nabla \tilde{Q}(\tilde{p})$. The subscript i in g_i is to remind readers that the 3D colour intensity gradient is dependent on ω_i. Thus equation (4.46) becomes

$$\nabla I_j(d_j) = g_i{}^{\mathrm{T}} (R_0^j)^{\mathrm{T}} K^{-1} \begin{bmatrix} \frac{\partial D_{\zeta,j}^{-1}}{\partial u_j} & 0 \\ 0 & \frac{\partial D_{\zeta,j}^{-1}}{\partial v_j} \\ 0 & 0 \end{bmatrix}. \tag{4.47}$$

Let $\sigma_j := \frac{\partial D_{\zeta,j}^{-1}}{\partial u_j}$ and $\varphi_j := \frac{\partial D_{\zeta,j}^{-1}}{\partial v_j}$ for convenience. Since we assume K to be the identity (via Assumption 9), Equation (4.47) becomes

$$\nabla I_j(d_j) = g_i{}^{\mathrm{T}} (R_0^j)^{\mathrm{T}} \begin{bmatrix} \sigma_j & 0 \\ 0 & \varphi_j \\ 0 & 0 \end{bmatrix}. \tag{4.48}$$

Finding σ_j and φ_j

σ_j and φ_j denote the changes in depth with respect to a change in the image coordinates u_j and v_j, respectively. They are dependent on the surface of the rigid body. To evaluate these variables, an assumption on the local surface curvature of the rigid body is required. As per Assumption 10, we assume that the hand surface can be well approximated locally by sections of quadric C^2 surfaces stitched together. In the case of the palm, one can assume locally spherical surfaces for the knuckles on the back of the palm and ellipsoidal surfaces for sides of the palm (see Figure 4.10 on page 56). Under this assumption, one can derive equations that describe σ_j and φ_j.

Suppose we choose a sample point ω_i on a quadric. Let s_j be ω_i in the camera coordinates of the jth camera, *i.e.*

$$s_j := R_0^j \omega_i + t_j, \tag{4.49}$$

where t_j is the translation vector from the camera to the world coordinates. Rearranging (4.49) gives

$$\begin{aligned} \omega_i &= (R_0^j)^{-1}(s_j - t_j) && (4.50)\\ &= R_j^0(s_j - t_j). && (4.51) \end{aligned}$$

The neighbourhood of ω_i belongs to a quadric surface and thus satisfies the following implicit equation,

$$\omega_i{}^{\mathrm{T}} Q \omega_i = r. \tag{4.52}$$

The symmetric matrix $Q \in \mathbb{R}^{3\times 3}$ and constant r determine the type of quadric surface. Let $Q_j := R_j^0{}^{\mathrm{T}} Q R_j^0$. Since Q is symmetric, it follows that $Q_j = Q_j{}^{\mathrm{T}}$, since

$$\begin{aligned} Q_j &= R_j^0{}^{\mathrm{T}} Q R_j^0 && (4.53)\\ &= (R_j^0{}^{\mathrm{T}} Q{}^{\mathrm{T}} R_j^0)^{\mathrm{T}} && (4.54)\\ &= (R_j^0{}^{\mathrm{T}} Q R_j^0)^{\mathrm{T}} && (4.55)\\ &= Q_j{}^{\mathrm{T}}. && (4.56) \end{aligned}$$

Thus, using (4.51), Equation (4.52) can be expressed in the jth camera coordinate frame as

$$\begin{aligned} &(R_j^0(s_j - t_j))^{\mathrm{T}} Q (R_j^0(s_j - t_j)) = r \quad \text{or} && (4.57)\\ &s_j{}^{\mathrm{T}} Q_j s_j - 2 t_j{}^{\mathrm{T}} Q_j s_j + t_j{}^{\mathrm{T}} Q_j t_j = r. && (4.58) \end{aligned}$$

One can view s_j as a ray of a finite length that originates from the jth camera origin to ω_i. To be precise, s_j can be expressed as

$$s_j = k_{j,z} \begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} = k_{j,z} d_j. \tag{4.59}$$

At this point one should make note of the fact that φ_j is $\frac{\partial k_{j,z}}{\partial v_j}$. This is straightforward to see; as denoted in Equation (4.48), φ_j is the change in depth in the jth camera coordinate frame (*i.e.* $D_{\zeta,j}^{-1}(d_j)$ or $k_{j,z}$) with respect to a change in

the v coordinate of the jth camera image plane (*i.e.* $\frac{\partial k_{j,z}}{\partial v_j}$). Substituting (4.59) into (4.58) gives

$$k_{j,z}^2 d_j{}^{\mathrm{T}} Q_j d_j - 2k_{j,z} t_j{}^{\mathrm{T}} Q_j d_j = r^2 - t_j{}^{\mathrm{T}} Q_j t_j. \tag{4.60}$$

Differentiating (4.60) implicitly for $\frac{\partial k_{j,z}}{\partial v_j}$ gives

$$(2k_{j,z} d_j{}^{\mathrm{T}} Q_j d_j \frac{\partial k_{j,z}}{\partial v_j} + 2k_{j,z}^2 d_j{}^{\mathrm{T}} Q_j e_2) - \tag{4.61}$$

$$(2t_j{}^{\mathrm{T}} Q_j d_j \frac{\partial k_{j,z}}{\partial v_j} + 2k_{j,z} t_j{}^{\mathrm{T}} Q_j e_2) = 0. \tag{4.62}$$

Rearranging this gives

$$\frac{\partial k_{j,z}}{\partial v_j} = \frac{k_{j,z} t_j{}^{\mathrm{T}} Q_j e_2 - k_{j,z}^2 d_j{}^{\mathrm{T}} Q_j e_2}{k_{j,z} d_j{}^{\mathrm{T}} Q_j d_j - t_j{}^{\mathrm{T}} Q_j d_j}, \tag{4.63}$$

which simplifies to

$$\frac{\partial k_{j,z}}{\partial v_j} = k_{j,z} \frac{t_j{}^{\mathrm{T}} Q_j e_2 - k_{j,z} d_j{}^{\mathrm{T}} Q_j e_2}{k_{j,z} d_j{}^{\mathrm{T}} Q_j d_j - t_j{}^{\mathrm{T}} Q_j d_j}. \tag{4.64}$$

Taking into account that $s_j = k_{j,z} d_j$, φ_j is thus

$$\varphi_j = \frac{\partial k_{j,z}}{\partial v_j} = -k_{j,z} \frac{(s_j - t_j)\,{}^{\mathrm{T}} Q_j e_2}{(s_j - t_j)\,{}^{\mathrm{T}} Q_j d_j}. \tag{4.65}$$

Similarly σ_j is thus

$$\sigma_j = \frac{\partial k_{j,z}}{\partial u_j} = -k_{j,z} \frac{(s_j - t_j)\,{}^{\mathrm{T}} Q_j e_1}{(s_j - t_j)\,{}^{\mathrm{T}} Q_j d_j}. \tag{4.66}$$

Finding ϕ_i's with a zero 2nd entry

Recall from Equation (4.36) the type of ϕ_i's in the set Ψ:

$$\Psi := \{\phi_1, ..., \phi_6\} = \{ \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \end{bmatrix} \}. \tag{4.67}$$

We now examine where one can choose sample points on a rigid body to generate these ϕ_i's with a guaranteed zero in the second entry (denoted as $\phi_{i,2}$). For

simplicity, we first assume the rigid body to be an ellipsoid and explore how one can choose the corresponding Ψ set. This is further generalised to the palm by arguing that the complex shape of the palm can be approximated by patches of quadric surfaces stitched together and that a Ψ set can be constructed by taking sample points from these patches (see Figure 4.10 on page 56).

As we will see later, there exist trajectories on the rigid body that host such sample points. One can then show that $\phi_1, ..., \phi_5$ in Ψ can be chosen from these paths. Being able to choose $\phi_1, ..., \phi_5$ that have a zero second entry is convenient as it makes choosing the last sample point ϕ_6 in Ψ fairly trivial. By Equation (4.37) and (4.41), $\phi_{i,2}$ is given as

$$\phi_{i,2} = \triangle Y_i \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \triangle Y_{i,2}. \tag{4.68}$$

Substituting the expression in (4.43) for $\triangle Y_i$ gives

$$\phi_{i,2} = (\nabla I_1(d_1) {F_1}^{\mathrm{T}} R_0^1 - \nabla I_2(d_2) {F_2}^{\mathrm{T}} R_0^2) e_2. \tag{4.69}$$

Substituting the expression in (4.48) for $\nabla I_1(d_1)$ and $\nabla I_2(d_2)$ gives

$$\phi_{i,2} = {g_i}^{\mathrm{T}} \left({R_0^1}^{\mathrm{T}} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \varphi_1 \\ 0 & 0 \end{bmatrix} {F_1}^{\mathrm{T}} R_0^1 - {R_0^2}^{\mathrm{T}} \begin{bmatrix} \sigma_2 & 0 \\ 0 & \varphi_2 \\ 0 & 0 \end{bmatrix} {F_2}^{\mathrm{T}} R_0^2 \right) e_2 \tag{4.70}$$

$$= {g_i}^{\mathrm{T}} {R_0^1}^{\mathrm{T}} \left(\begin{bmatrix} \sigma_1 & 0 \\ 0 & \varphi_1 \\ 0 & 0 \end{bmatrix} {F_1}^{\mathrm{T}} - {R_1^2}^{\mathrm{T}} \begin{bmatrix} \sigma_2 & 0 \\ 0 & \varphi_2 \\ 0 & 0 \end{bmatrix} {F_2}^{\mathrm{T}} R_1^2 \right) R_0^1 e_2. \tag{4.71}$$

Since we have chosen the $\tilde{C}$ coordinate system, R_0^1 becomes the identity. This simplifies Equation (4.71) to

$$\phi_{i,2} = {g_i}^{\mathrm{T}} \left(\begin{bmatrix} \sigma_1 & 0 \\ 0 & \varphi_1 \\ 0 & 0 \end{bmatrix} {F_1}^{\mathrm{T}} - {R_1^2}^{\mathrm{T}} \begin{bmatrix} \sigma_2 & 0 \\ 0 & \varphi_2 \\ 0 & 0 \end{bmatrix} {F_2}^{\mathrm{T}} R_1^2 \right) e_2. \tag{4.72}$$

The assumption made for the camera view configuration (see Assumption 1) means that R_1^2 has the following form

$$R_1^2 = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}. \tag{4.73}$$

It is straightforward to show that R_1^2 can be omitted, *i.e.* Equation (4.72) is equivalent to the following

$$\phi_{i,2} = g_i \left(\begin{bmatrix} \sigma_1 & 0 \\ 0 & \varphi_1 \\ 0 & 0 \end{bmatrix} {F_1}^{\mathrm{T}} - \begin{bmatrix} \sigma_2 & 0 \\ 0 & \varphi_2 \\ 0 & 0 \end{bmatrix} {F_2}^{\mathrm{T}} \right) e_2. \tag{4.74}$$

After substituting the expressions for F_1 and F_2 (see Equation (4.33)) and using some algebraic manipulation

$$\phi_{i,2} = (\frac{\varphi_1}{k_{1,z}} - \frac{\varphi_2}{k_{2,z}}) g_{i,y}, \tag{4.75}$$

where $g_{i,y} = g_i e_2$.

The easiest options for setting $\phi_{i,2} = 0$ are to find regions on the rigid body where either $g_{i,y} = 0$ or $\varphi_1 = \varphi_2 = 0$.

Under the assumptions of a single light source (Assumption 2) and a Lambertian surface (Assumption 3), there is a closed path on the rigid body where the corresponding $g_{i,y}$'s of the sample points are zero. The shape of the path is dependent on the direction of the light source. From now on, this path on the rigid body will be denoted as $\mathcal{L}_{\text{light}}$. Figure 4.8 shows the dependency of this closed path on the light direction for a spherical surface.

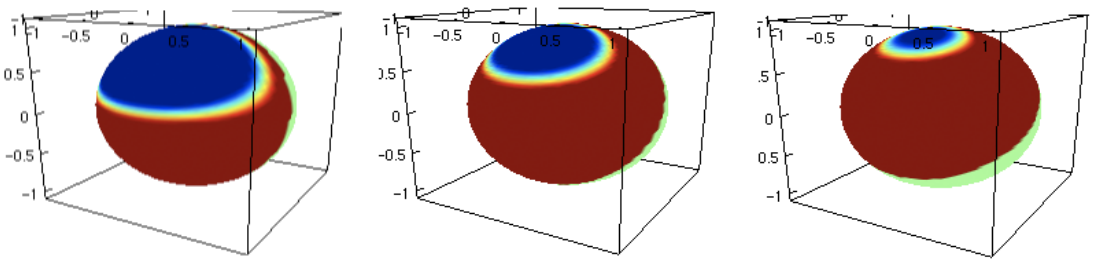

Figure 4.8: The ring on the spherical surface indicates the trajectory $\mathcal{L}_{\text{light}}$ where $g_{i,y} = 0$. The shape and size of $\mathcal{L}_{\text{light}}$ varies depending on the elevation angle between the light source and the horizontal plane. $\mathcal{L}_{\text{light}}$ for elevation angles of 10 deg (left), 30 deg (middle) and 50 deg (right) are shown.

For the case $\varphi_1 = \varphi_2 = 0$, we need to examine the structure of φ_1 and φ_2. Substituting the expression for φ_j (Equation (4.65)) into Equation (4.75) gives

$$\phi_{i,2} = -\left[\frac{(s_1 - t_1)^{\mathrm{T}} Q_1 e_2}{(s_1 - t_1)^{\mathrm{T}} Q_1 d_1} - \frac{(s_2 - t_2)^{\mathrm{T}} Q_2 e_2}{(s_2 - t_2)^{\mathrm{T}} Q_2 d_2} \right] g_{i,y}. \tag{4.76}$$

One can show that $(s_1 - t_1)^{\mathrm{T}} Q_1 e_2 = (s_2 - t_2)^{\mathrm{T}} Q_2 e_2$, given the fact that $R_2^1 e_2 = e_2$, since

$$\begin{aligned}
(s_2 - t_2)^{\mathrm{T}} Q_2 e_2 &= (s_2 - t_2)^{\mathrm{T}} R_2^{0\,\mathrm{T}} Q R_2^0 e_2 && (4.77)\\
&= \omega^{\mathrm{T}} Q R_1^0 R_2^1 e_2 && (4.78)\\
&= \omega^{\mathrm{T}} Q R_1^0 e_2 && (4.79)\\
&= (s_1 - t_1)^{\mathrm{T}} R_1^{0\,\mathrm{T}} Q R_1^0 e_2 && (4.80)\\
&= (s_1 - t_1)^{\mathrm{T}} Q_1 e_2. && (4.81)
\end{aligned}$$

Let

$$\varsigma_{i,j} := (s_j - t_j)^{\mathrm{T}} Q_j d_j. \quad (4.82)$$

Then

$$\phi_{i,2} = -(s_1 - t_1)^{\mathrm{T}} Q_1 e_2 \left[\frac{1}{\varsigma_{i,1}} - \frac{1}{\varsigma_{i,2}} \right] g_{i,y}. \quad (4.83)$$

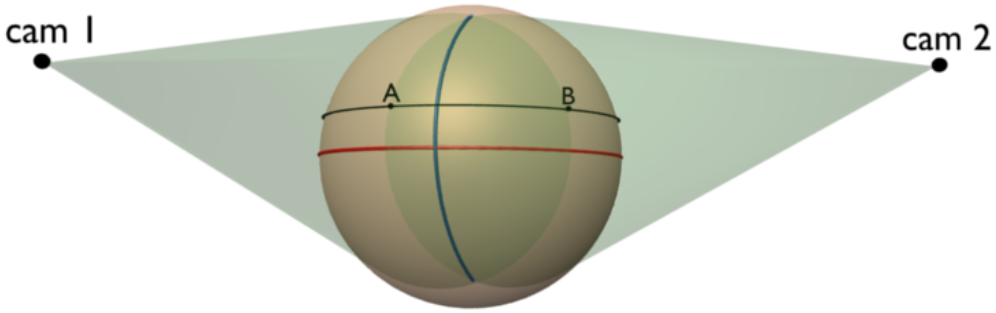

Figure 4.9: Trajectories on a spherical surface where $\phi_{i,2} = 0$ are indicated in red and blue. The red trajectory, $\mathcal{L}_{\text{horiz}}$, is the equator of the sphere. The dark green region on the sphere represents the portion of the sphere visible in both camera views. One can show that the blue trajectory, $\mathcal{L}_{\text{long}}$, always exists within the mutually visible region of the sphere. Note that the open angle of the mutually visible region is much larger in practice than that depicted above. The fact that the spherical surface patches on the hand are small in comparison with the distances between the hand and the cameras means that the opening angle reaches close to 150° for our camera arrangement (see Figure 3.1).

In addition to $\mathcal{L}_{\text{light}}$, there are two trajectories on the mutually visible portion of the ellipsoid (*i.e.* the portion visible in both camera views) where the sample

points' corresponding $\phi_{i,2}$'s are equal to 0. One runs longitudinally (see Figure 4.9) and will be denoted as $\mathcal{L}_{\text{long}}$. Its exact location on the ellipsoid varies depending on the calibration matrices of the cameras, the stereo configuration, the curvature and the relative position of the ellipsoidal surface. Nevertheless, one can show that this trajectory always exists within the mutually visible portion of the ellipsoid (see Appendix C for further details).

The other trajectory runs roughly horizontally[§] across the ellipsoid where $(s_1 - t_1)^{\mathrm{T}} Q_j e_2 = 0$. This trajectory will be denoted as $\mathcal{L}_{\text{horiz}}$. For a spherical rigid body, $\mathcal{L}_{\text{horiz}}$ runs exactly horizontally. For other ellipsoids, the trajectory has a slight slant - the slanting angle is dependent on the ratios of the ellipsoid's principal axes and the ellipsoid's orientation (see Appendix C for further details).

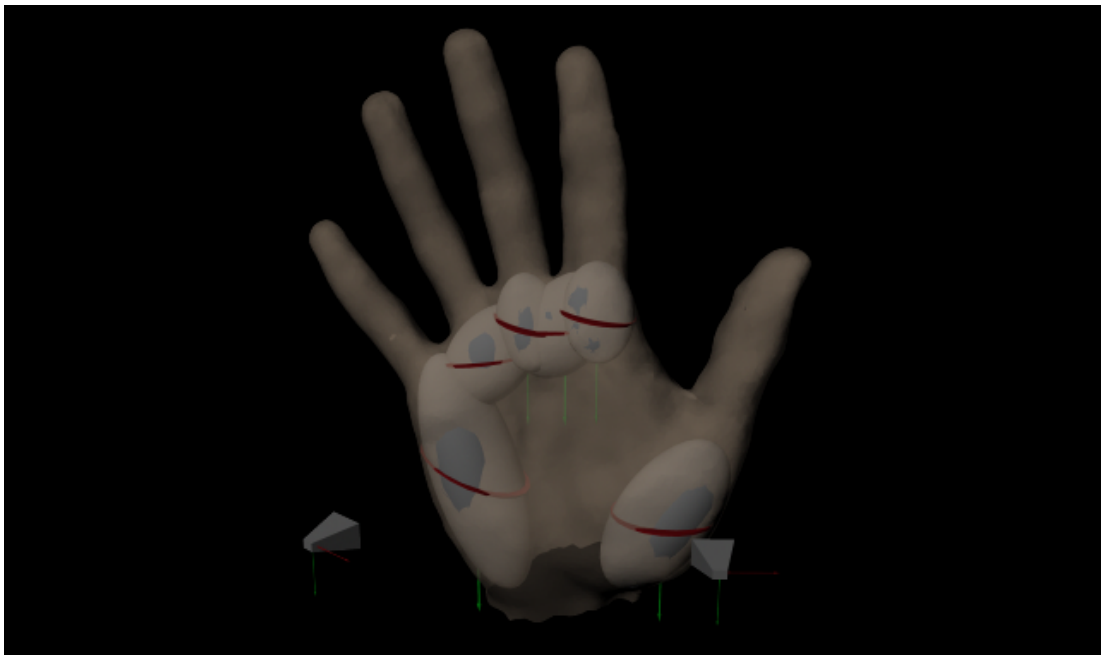

Figure 4.10: Various sections of the palm can be well approximated by ellipsoids. The red lines indicate the typical positions of $\mathcal{L}_{\text{horiz}}$ for each ellipsoid.

The Palm as a Patchwork of Quadrics

To extend the trajectory results to a real hand, we assume that parts of the hand can be constructed by stitching patches of quadric surfaces together, as per Assumption 10. Several parts of the palm can be well approximated by ellipsoids,

[§]with respect to the camera coordinate frames.

in particular the sides of the palm and the parts just below the fingers (see Figure 4.10). On each ellipsoid will be a $\mathcal{L}_{\text{horiz}}$. In what follows, we will show how one can choose a linear independent set $\{\phi_1, \phi_2, \phi_3\}$ from one of these ellipsoids. We then show how one can choose $\{\phi_4, \phi_5\}$ from another ellipsoid on the palm. Then obtaining the last point ϕ_6 to complete the Ψ set becomes a trivial matter of choosing yet another point on the remaining parts of the palm.

Choosing $\{\omega_1, \omega_2, \omega_3\}$

Recall that we wish to construct a linearly independent set Ψ to show the positivity of $\hat{H}$. The first step is to find a set $\{\omega_1, \omega_2, \omega_3\}$ such that their corresponding ϕ_i's are linearly independent and exhibit the following structure:

$$\{\phi_1, \phi_2, \phi_3\} = \{ \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \end{bmatrix} \}, \tag{4.84}$$

where $|\delta| << 1$.

We will ignore the 4th and 6th entries of $\{\phi_1 ... \phi_3\}$ for now and show that the linear independence of $\{\phi_1 ... \phi_3\}$ can be determined solely by examining the 1st, 3rd and 5th entries of ϕ_i. In the following analysis, we denote $\hat{\phi}_i$ as ϕ_i without the 2nd, 4th and 6th entries. The set of sample points that result in a linear independent set $\{\hat{\phi}_1 ... \hat{\phi}_3\}$ will be chosen along $\mathcal{L}_{\text{horiz}}$ of one of the ellipsoids on the palm. Recall that

$$\phi_i = (\triangle Y_i J_{\Gamma_i}(\mathbf{x}))^{\mathrm{T}} \tag{4.85}$$

$$= (\triangle Y_i \begin{bmatrix} 1 & 0 & 0 & 0 & \omega_{i,z} & -\omega_{i,y} \\ 0 & 1 & 0 & -\omega_{i,z} & 0 & \omega_{i,x} \\ 0 & 0 & 1 & \omega_{i,y} & -\omega_{i,x} & 0 \end{bmatrix})^{\mathrm{T}}. \tag{4.86}$$

Hence $\hat{\phi}_i$ is simply

$$\hat{\phi}_i = (\triangle Y_i \begin{bmatrix} 1 & 0 & \omega_{i,z} \\ 0 & 0 & 0 \\ 0 & 1 & -\omega_{i,x} \end{bmatrix})^{\mathrm{T}}. \tag{4.87}$$

Recall from Equation (4.43) on page 49 that

$$\begin{aligned}
\triangle Y_i &= Y_{i,1} - Y_{i,2} && (4.88)\\
&= \nabla I_1(d_1) {F_1}^{\mathrm{T}} R_0^1 - \nabla I_2(d_2) {F_2}^{\mathrm{T}} R_0^2 && (4.89)\\
&= {g_i}^{\mathrm{T}} {R_0^1}^{\mathrm{T}} \left(\begin{bmatrix} \sigma_1 & 0 \\ 0 & \varphi_1 \\ 0 & 0 \end{bmatrix} {F_1}^{\mathrm{T}} - {R_1^2}^{\mathrm{T}} \begin{bmatrix} \sigma_2 & 0 \\ 0 & \varphi_2 \\ 0 & 0 \end{bmatrix} {F_2}^{\mathrm{T}} R_1^2 \right) R_0^1. && (4.90)
\end{aligned}$$

Under the $\tilde{C}$ coordinate system, R_0^1 becomes the identity. Expanding F_1, F_2 (see Equation (4.33) on page 46) gives

$$\triangle Y_i = {g_i}^{\mathrm{T}} \left(\begin{bmatrix} \sigma_1 & 0 \\ 0 & \varphi_1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{k_{1,z}} & 0 & -\frac{k_{1,x}}{k_{1,z}^2} \\ 0 & \frac{1}{k_{1,z}} & -\frac{k_{1,y}}{k_{1,z}^2} \end{bmatrix} - {R_1^2}^{\mathrm{T}} \begin{bmatrix} \sigma_2 & 0 \\ 0 & \varphi_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{k_{2,z}} & 0 & -\frac{k_{2,x}}{k_{2,z}^2} \\ 0 & \frac{1}{k_{2,z}} & -\frac{k_{2,y}}{k_{2,z}^2} \end{bmatrix} R_1^2 \right) \tag{4.91}$$

$$= {g_i}^{\mathrm{T}} \left(\begin{bmatrix} \sigma_1 \frac{1}{k_{1,z}} & 0 & -\sigma_1 \frac{k_{1,x}}{k_{1,z}^2} \\ 0 & \varphi_1 \frac{1}{k_{1,z}} & -\varphi_1 \frac{k_{1,y}}{k_{1,z}^2} \\ 0 & 0 & 0 \end{bmatrix} - {R_1^2}^{\mathrm{T}} \begin{bmatrix} \sigma_2 \frac{1}{k_{2,z}} & 0 & -\sigma_2 \frac{k_{2,x}}{k_{2,z}^2} \\ 0 & \varphi_2 \frac{1}{k_{2,z}} & -\varphi_2 \frac{k_{2,y}}{k_{2,z}^2} \\ 0 & 0 & 0 \end{bmatrix} R_1^2 \right). \tag{4.92}$$

Once again, note that the quantities $\phi_j, \sigma_j, k_{j,x}, k_{j,y}, k_{j,z}$ are all dependent on the ith sample point.

Since the sample points $\omega_1 ... \omega_3$ are chosen on $\mathcal{L}_{\text{horiz}}$, $\varphi_1 = \varphi_2 = 0$. For these sample points, equation (4.92) simplifies to

$$\triangle Y_i = {g_i}^{\mathrm{T}} \left(\begin{bmatrix} \sigma_1 \frac{1}{k_{1,z}} & 0 & -\sigma_1 \frac{k_{1,x}}{k_{1,z}^2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - {R_1^2}^{\mathrm{T}} \begin{bmatrix} \sigma_2 \frac{1}{k_{2,z}} & 0 & -\sigma_2 \frac{k_{2,x}}{k_{2,z}^2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} R_1^2 \right). \tag{4.93}$$

For convenience let

$$\hat{\sigma}_j := \sigma_j \frac{1}{k_{j,z}}, \tag{4.94}$$

and thus

$$\triangle Y_i = {g_i}^{\mathrm{T}} \left(\hat{\sigma}_1 \begin{bmatrix} 1 & 0 & -\frac{k_{1,x}}{k_{1,z}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \hat{\sigma}_2 {R_1^2}^{\mathrm{T}} \begin{bmatrix} 1 & 0 & -\frac{k_{2,x}}{k_{2,z}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} R_1^2 \right). \tag{4.95}$$

$\hat{\sigma}_1$ and $\hat{\sigma}_2$ vary non-linearly along $\mathcal{L}_{\text{horiz}}$. Let ω_1 be a point on $\mathcal{L}_{\text{horiz}}$ where its corresponding $\hat{\sigma}_1 = 0$ and $\hat{\sigma}_2 \neq 0$. In addition, let ω_3 be a point on $\mathcal{L}_{\text{horiz}}$ where its

corresponding $\hat{\sigma}_1 << \hat{\sigma}_2$. Finally, let ω_2 be a sample point located somewhere else along $\mathcal{L}_{\text{horiz}}$. We will show that a linear independent set of $\hat{\phi}_i$'s can be achieved by choosing these ω_1, ω_2, ω_3. First we show that ϕ_1 and ϕ_3 are linearly independent. For ω_1, the corresponding $\triangle Y_1$ is

$$\triangle Y_1 = -{g_1}^{\mathrm{T}} {R_1^2}^{\mathrm{T}} \left(\hat{\sigma}_2 \begin{bmatrix} 1 & 0 & -\frac{k_{2,x}}{k_{2,z}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} R_1^2 \right) \tag{4.96}$$

$$= \tilde{g}_{1,x} \hat{\sigma}_2 \begin{bmatrix} 1 & 0 & -\frac{k_{2,x}}{k_{2,z}} \end{bmatrix} R_1^2, \tag{4.97}$$

where $\tilde{g}_{1,x}$ is the x-component of $-{g_1}^{\mathrm{T}} {R_1^2}^{\mathrm{T}}$. Therefore the corresponding $\hat{\phi}_1$ is

$$\hat{\phi}_1 = \tilde{g}_{1,x} \hat{\sigma}_2 (\begin{bmatrix} 1 & 0 & -\frac{k_{2,x}}{k_{2,z}} \end{bmatrix} R_1^2 \begin{bmatrix} 1 & 0 & \omega_{1,z} \\ 0 & 0 & 0 \\ 0 & 1 & -\omega_{1,x} \end{bmatrix}) ^{\mathrm{T}}. \tag{4.98}$$

The magnitude of $\hat{\phi}_1$ does not affect linear independence. Let ω_i^j be ω_i in the jth camera coordinates. ω_i^j is related to k_j via the projection matrix K_j. Since K_j is the identity via Assumption 9, $\omega_i^j = k_j$[¶]. Then let $\bar{\phi}_1$ be a rescaled version of $\hat{\phi}_1$ *i.e.*

$$\bar{\phi}_1 := \frac{k_{2,z}}{\tilde{g}_{1,x} \hat{\sigma}_2} \bar{\phi}_1 \tag{4.99}$$

$$= (\begin{bmatrix} k_{2,z} & 0 & -k_{2,x} \end{bmatrix} R_1^2 \begin{bmatrix} 1 & 0 & \omega_{1,z} \\ 0 & 0 & 0 \\ 0 & 1 & -\omega_{1,x} \end{bmatrix}) ^{\mathrm{T}} \tag{4.100}$$

$$= (\begin{bmatrix} \omega_{1,z}^2 & 0 & -\omega_{1,x}^2 \end{bmatrix} R_1^2 \begin{bmatrix} 1 & 0 & \omega_{1,z} \\ 0 & 0 & 0 \\ 0 & 1 & -\omega_{1,x} \end{bmatrix}) ^{\mathrm{T}}. \tag{4.101}$$

In the same manner one can find the expression for $\bar{\phi}_3$ of ω_3, *i.e.*

$$\bar{\phi}_3 = (\begin{bmatrix} \omega_{3,z}^2 & 0 & -\omega_{3,x}^2 \end{bmatrix} R_1^2 \begin{bmatrix} 1 & 0 & \omega_{3,z} \\ 0 & 0 & 0 \\ 0 & 1 & -\omega_{3,x} \end{bmatrix}) ^{\mathrm{T}} + \epsilon, \tag{4.102}$$

[¶]Note that this generalises to a projection matrix with a non-unity focal length as the focal length merely scales ω_i^j.

where $\epsilon \in \mathbb{R}^3$ is an error term due to the fact that $\hat{\sigma}_1 \neq 0, \hat{\sigma}_1 << \hat{\sigma}_2$ at ω_3. Note that ϵ is a continuous function of ω that can be made arbitrarily small since $\epsilon \to 0$ as $\omega_3 \to \omega_B$, where ω_B lies exactly on the boundary of the mutually visible region.

Let $a := \begin{bmatrix} \omega_{1,z}^2 & 0 & -\omega_{1,x}^2 \end{bmatrix}$ and $b := \begin{bmatrix} \omega_{3,z}^2 & 0 & -\omega_{3,x}^2 \end{bmatrix}$. It is clear that the vector connecting the sample point ω_1 to camera 2's origin (*i.e.* ω_1^2) and the vector connecting ω_3 to camera 2's origin (*i.e.* ω_3^2) together span a plane, and therefore are linearly independent. a and b are rotated projections of these respective vectors on the horizontal plane, reflected on the YZ plane. They too are also linearly independent[‖]. Consequently $\bar{\phi}_1$ and $\bar{\phi}_3$ can be shown to be linearly independent based on their 1st and 2nd entries. To see why this is so, $\bar{\phi}_1$ and $\bar{\phi}_3$ can be rewritten as

$$\bar{\phi}_1 = \begin{bmatrix} (aR_1^2)_x \\ (aR_1^2)_z \\ * \end{bmatrix}, \tag{4.103}$$

and

$$\bar{\phi}_3 = \begin{bmatrix} (bR_1^2)_x \\ (bR_1^2)_z \\ * \end{bmatrix} + \epsilon. \tag{4.104}$$

Notice that the rotation matrix R_1^2 does not change the linear independence between a and b. In addition R_1^2 is a rotation on the horizontal plane and so, like a and b, aR_1^2 and bR_1^2 will remain on the horizontal plane. This implies that the linear independence of $\bar{\phi}_1$ and $\bar{\phi}_3$ (and consequently $\hat{\phi}_1$ and $\hat{\phi}_3$) will be preserved for a sufficiently small ϵ. An ω_3 with a sufficiently small ϵ can always be chosen since $\epsilon \to 0$ in a continuous manner as $\omega_3 \to \omega_B$.

Once an ω_3 with a sufficiently small ϵ is chosen, we fix ω_1 and ω_3. We then choose a ω_2 somewhere along $\mathcal{L}_{\text{horiz}}$. Let ω be a candidate point for ω_2 and $\hat{\phi}$ be the associated vector. To show linear independence, let

$$\Upsilon := \hat{\phi} \cdot (\hat{\phi}_1 \times \hat{\phi}_3), \tag{4.105}$$

where $\times$ denotes the cross-product operator, and observe that $\Upsilon \neq 0$ implies the linear independence of $\{\hat{\phi}_1, \hat{\phi}, \hat{\phi}_3\}$. Note that $\hat{\phi}_1, \hat{\phi}_3$ and $(\hat{\phi}_1 \times \hat{\phi}_3)$ are constant vectors since ω_1 and ω_3 have been fixed. For readability let $Z := \hat{\phi}_1 \times \hat{\phi}_3$. Then Υ can be re-expressed as

[‖]This is always true because it is not possible for $\mathcal{L}_{\text{horiz}}$ on an ellipsoid to vertically align with the y-axis of the camera views. $\mathcal{L}_{\text{horiz}}$ aligning with the y-axis would lead to linear dependence of a and b.

$$\Upsilon = \hat{\phi} \cdot Z = Z_1\hat{\phi}_x + Z_2\hat{\phi}_y + Z_3\hat{\phi}_z, \tag{4.106}$$

where Z_1, Z_2, Z_3 are elements of Z and $\hat{\phi}_x, \hat{\phi}_y, \hat{\phi}_y$ are elements of $\bar{\phi}$. The main idea is to show that the non-linear terms $\hat{\phi}_x, \hat{\phi}_y, \hat{\phi}_z$ do not cancel each other out for the given constants Z_1, Z_2, Z_3. Recall from Equation (4.87) that $\hat{\phi}$ on $\mathcal{L}_{\text{horiz}}$ is

$$\hat{\phi} = \begin{bmatrix} \triangle Y_x \\ \triangle Y_z \\ \triangle Y_x\omega_z - \triangle Y_z\omega_x \end{bmatrix}, \tag{4.107}$$

where $\hat{k}$ is a scaling factor. Then Υ becomes

$$\begin{aligned} \Upsilon &= Z_1\triangle Y_x + Z_2\triangle Y_z && (4.108)\\ &+ Z_3(\triangle Y_x\omega_z - \triangle Y_z\omega_x). && (4.109) \end{aligned}$$

One can show that (see Appendix D.2)

$$\triangle Y_x = (1+E)(\omega_z + t^0_{1,z}) + Et^1_{2,z} \tag{4.110}$$

and that

$$\triangle Y_z = -\left((1+E)(\omega^1_x + t^0_{1,x}) + Et^1_{2,x}\right), \tag{4.111}$$

where E is a rational trigonmetric polynomial (see Appendix D.2). Substituting this into Υ gives

$$\begin{aligned} \Upsilon &= Z_1\left((1+E)(\omega_z + t^0_{1,z}) + Et^1_{2,z}\right) && (4.112)\\ &\quad -Z_2\left((1+E)(\omega_x + t^0_{1,x}) + Et^1_{2,x}\right) && (4.113)\\ &\quad +Z_3\left(\omega_z((1+E)(\omega_z + t^0_{1,z}) + Et^1_{2,z}\right) && (4.114)\\ &\quad +Z_3\left(\omega_x((1+E)(\omega_x + t^0_{1,x}) + Et^1_{2,x}\right). && (4.115) \end{aligned}$$

Denote

$$\begin{aligned} A(\omega_x, \omega_z) &:= \{Z_1(\omega_z + t^0_{1,z}) - Z_2(\omega_x + t^0_{1,x}) && (4.116)\\ &\quad +Z_3(\omega_x(\omega_x + t^0_{1,x}) + \omega_z(\omega_z + t^0_{1,z})\}, && (4.117) \end{aligned}$$

and the constant B as

$$B := Z_1t^1_{2,z} - Z_2t^1_{2,x} + Z_3(t^1_{2,x} + t^1_{2,z}). \tag{4.118}$$

Then Υ in Equation (4.115) can be expressed as

$$\Upsilon = A(\omega_x, \omega_z) + E(A(\omega_x, \omega_z) + B). \tag{4.119}$$

We further exploit the fact that ω_x and ω_z are dependent on each other in the sense that ω lies on $\mathcal{L}_{\text{horiz}}$, which is an elliptic curve since it is a planar cross-section of an ellipsoid. Therefore sample points ω (and thus their corresponding ϕ's) on $\mathcal{L}_{\text{horiz}}$ can be re-parameterised as the polar coordinates of an elliptic curve *i.e.*

$$\omega = R_e^0 \begin{bmatrix} q \sin\vartheta \\ 0 \\ r \cos\vartheta \end{bmatrix} + t_e^0, \tag{4.120}$$

where $\vartheta \in [0, 2\pi]$. R_e^0 and $t_e^0 \in \mathbb{R}^3$ are the rotation matrix and translation vector respectively, that transform sample points from the local elliptic curve coordinates to the $\tilde{C}$ coordinate system. Hence Υ can be expressed entirely in terms of ϑ as

$$\Upsilon = \Upsilon(\vartheta) = A(\vartheta) + E(\vartheta)(A(\vartheta) + B). \tag{4.121}$$

We wish to show that $\Upsilon(\vartheta)$ is a non-linear continuous function which implies that there exists an ω_2 (and via the continuity argument, ω's in the neighbourhood of ω_2) where $\Upsilon \neq 0$. To do so requires one to examine the properties of $A(\vartheta)$ and $E(\vartheta)$. First we use a high level argument to show that $A(\vartheta)$ will always be a non-linear trignometric polynomial. $A(\vartheta)$ is a linear combination of trigonmetric polynomials $p_k(\vartheta), k = \{1, 2, 3\}$ *i.e.*

$$A(\vartheta) = Z_1 p_1(\vartheta) - Z_2 p_2(\vartheta) + Z_3 p_3(\vartheta). \tag{4.122}$$

p_1 and p_2 are trigonometric polynomials of the 1st order whereas p_3 is a trigonmetric polynomial of the 2nd order. Therefore any linear combination of p_1 and p_2 will not be able to cancel out the p_3 term. The second order terms in p_3 can be shown to exist always as Z_3 is guaranteed to be non-zero (see Appendix D.1).

Next, $E(\vartheta)$ can be shown to be a rational trigonometric function (see Appendix D.2). Suppose $\Upsilon(\vartheta) = 0, \forall\vartheta$. This would imply that

$$E(\vartheta) = \frac{A(\vartheta)}{A(\vartheta) + B} \tag{4.123}$$

for segments on $\mathcal{L}_{\text{horiz}}$ where $(A(\vartheta) + B) \neq 0$. Figure 4.11 shows the typical shape of the functions $E(\vartheta)$ and $\frac{A(\vartheta)}{A(\vartheta)+B}$ for a particular pose of an ellipsoid. One can appreciate that these two functions are not equal.

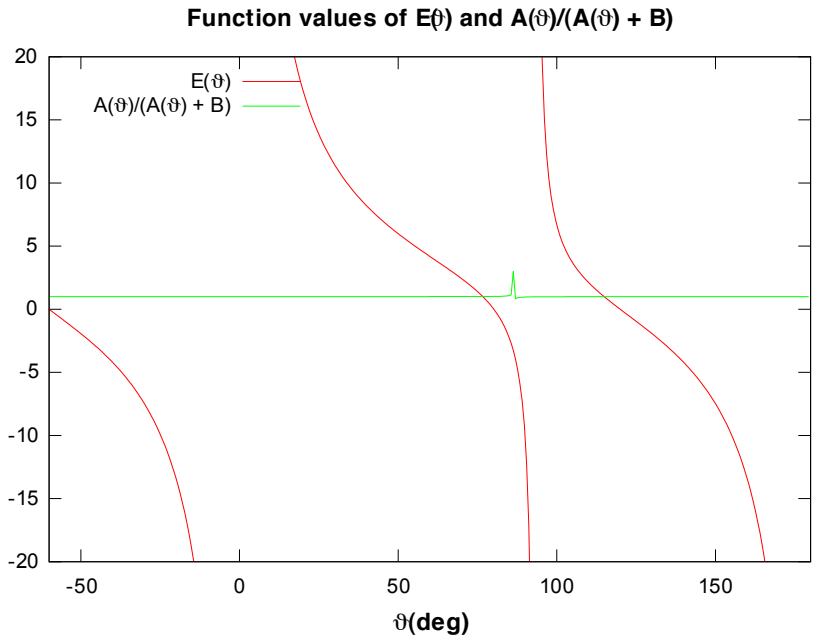

Figure 4.11: Typical function values for $E(\vartheta)$ and $\frac{A(\vartheta)}{A(\vartheta)+B}$. The spike in the green curve is attributed to the region where the denominator $A(\vartheta) + B \to 0$.

This means that $\exists\vartheta, \text{s.t.}\Upsilon(\vartheta) \neq 0$. Let the corresponding sample point generated by this ϑ be ω_2. Hence the corresponding $\Psi = \{\phi_1, \phi_2, \phi_3\}$ forms a linearly independent set. Recall from (4.36) on page 47 that the 4th and 6th entries of ϕ_1, ϕ_2 and ϕ_3 are of the order $\delta*$. To see how sample points with such entries can be chosen please refer to the discussion after Lemma 4.5.4 in the next subsection.

Choosing $\{\omega_4, \omega_5\}$

The next step is to find $\{\omega_4, \omega_5\}$ such that their corresponding ϕ_i's are linearly independent based on the 4th and 6th entries alone and exhibit the following structure:

$$\{\phi_4, \phi_5\} = \{ \begin{bmatrix} * \\ 0 \\ * \\ * \\ * \\ * \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ * \\ * \\ * \end{bmatrix} \}. \tag{4.124}$$

It is straightforward to show that such points can be picked on $\mathcal{L}_{\text{horiz}}$. Recall that

$$\phi_i = (\triangle Y_i J_{\Gamma_i}(\mathbf{x}))^{\mathrm{T}} \tag{4.125}$$

$$= (\triangle Y_i \begin{bmatrix} 1 & 0 & 0 & 0 & \omega_{i,z} & -\omega_{i,y} \\ 0 & 1 & 0 & -\omega_{i,z} & 0 & \omega_{i,x} \\ 0 & 0 & 1 & \omega_{i,y} & -\omega_{i,x} & 0 \end{bmatrix})^{\mathrm{T}}. \tag{4.126}$$

Since $\triangle Y_i = \begin{bmatrix} * & 0 & * \end{bmatrix}^{\mathrm{T}}$ for any ω_i on $\mathcal{L}_{\text{horiz}}$, ϕ_i can alternatively be expressed as

$$\phi_i = (\triangle Y_i \begin{bmatrix} 1 & 0 & 0 & 0 & \omega_{i,z} & -\omega_{i,y} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \omega_{i,y} & -\omega_{i,x} & 0 \end{bmatrix})^{\mathrm{T}}. \tag{4.127}$$

Suppose we choose 2 sample points ω_4, ω_5 on $\mathcal{L}_{\text{horiz}}$ where the corresponding ϕ_1, ϕ_2 are linearly independent based on the 1st and 3rd entry (the procedure for doing this has been examined in the previous subsection). Then ω_4 and ω_5 can be shown to be linearly independent based on the 4th and 6th entry, provided $\omega_{4,y}$ and $\omega_{5,y}$ are non zero values. To see why this is so, consider the following lemma

Lemma 4.5.4. *Let* $\tilde{\phi}_1 = \begin{bmatrix} a \\ b \end{bmatrix}$ *and* $\tilde{\phi}_2 = \begin{bmatrix} c \\ d \end{bmatrix}$ *be linearly independent vectors. Let* e *and* f *be non-zero real numbers. Then* $\hat{\phi}_1 = e \begin{bmatrix} a \\ -b \end{bmatrix}$ *and* $\hat{\phi}_2 = f \begin{bmatrix} c \\ -d \end{bmatrix}$ *are also linearly independent.*

Proof. The linear independence of $\tilde{\phi}_1$ and $\tilde{\phi}_2$ is equivalent to $(\tilde{\phi}_1 \times \tilde{\phi}_2) \neq 0$. One can re-express $(\hat{\phi}_1 \times \hat{\phi}_2)$ as

$$(\hat{\phi}_1 \times \hat{\phi}_2) = -ef(\tilde{\phi}_1 \times \tilde{\phi}_2). \tag{4.128}$$

It is follows that $(\hat{\phi}_1 \times \hat{\phi}_2) \neq 0$ and therefore $\hat{\phi}_1$ and $\hat{\phi}_2$ are also linearly independent. □

The 4th entry of ϕ_i is in fact the 3rd entry of ϕ_i scaled by $\omega_{i,y}$. Similarly the 6th entry of ϕ_i is the 1st entry of ϕ_i scaled by $-\omega_{i,y}$. Hence by Lemma (4.5.4) ϕ_4 and ϕ_5 are linearly independent based on their 4th and 6th entries.,

To find a Ψ set $\{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5\}$ such that they are linearly independent, we exploit the fact that the palm consists of several ellipsoidal surfaces (see Figure

4.10 on page 56). First choose $\omega_1, \omega_2, \omega_3$ from an ellipsoid where $\omega_{i,y}$ of the sample points on $\mathcal{L}_{\text{horiz}}$ are close to 0. Then choose ω_4, ω_5 from another ellipsoid where $\omega_{i,y} >> 0$. For example, in Figure 4.10, one can choose $\omega_1, \omega_2, \omega_3$ from one of the bottom two ellipsoids, and ω_4, ω_5 from one of the ellipsoids on the top row. To show linear independence of this Ψ set, consider the following lemma [54].

Lemma 4.5.5. *Suppose a square matrix M can be broken into sub-blocks*

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \tag{4.129}$$

where A is square and full rank. Then the determinant of M can be expressed as

$$\det(M) = \det(A)\det(D - CA^{-1}B), \tag{4.130}$$

where a non-zero $\det(M)$ *implies linear independence of the columns of M.*

Recall that our Ψ at this stage exhibits the following form

$$\Psi = \{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5\} \tag{4.131}$$

$$\sim \begin{bmatrix} \phi_{1,1} & \phi_{2,1} & \phi_{3,1} & \phi_{4,1} & \phi_{5,1} \\ 0 & 0 & 0 & 0 & 0 \\ \phi_{1,3} & \phi_{2,3} & \phi_{3,3} & \phi_{4,3} & \phi_{5,3} \\ -\delta_1\phi_{1,3} & -\delta_2\phi_{2,3} & -\delta_3\phi_{3,3} & -N\phi_{4,3} & -(N+\epsilon)\phi_{5,3} \\ * & * & * & * & * \\ \delta_1\phi_{1,1} & \delta_2\phi_{2,1} & \delta_3\phi_{3,1} & N\phi_{4,1} & (N+\epsilon)\phi_{5,1} \end{bmatrix}, \tag{4.132}$$

where $|\delta_i| << 1$ and $N + \epsilon > N >> 0$.

For convenience, we will swap row 4 with 5 and row 3 with 1. We then normalise the matrix representation of Ψ based on the new row 1. We will also omit the 2nd row for now since it does not play a role in showing the linear independence of the current Ψ set. Note that linear independence is not lost by any of these operations. The resulting matrix $\bar{\Psi}$ ** takes on the following form

$$\bar{\Psi} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ k_1 & k_2 & k_3 & k_4 & k_5 \\ * & * & * & * & * \\ -\delta_1 & -\delta_2 & -\delta_3 & -N & -(N+\epsilon) \\ \delta_1 k_1 & \delta_2 k_2 & \delta_3 k_3 & Nk_4 & (N+\epsilon)k_5 \end{bmatrix}, \tag{4.133}$$

**See Appendix E for properties of this matrix.

where $|\delta_i| << 1$ and $0 << N < N+\epsilon$. It should be clear that linear independence of the columns of $\bar{\Psi}$ implies linear independence of Ψ. Let A be the sub-block

$$\begin{bmatrix} 1 & 1 & 1 \\ k_1 & k_2 & k_3 \\ * & * & * \end{bmatrix} \tag{4.134}$$

of $\bar{\Psi}$. This uniquely defines the B, C and D sub-blocks of $\bar{\Psi}$. Then the determinant of $\bar{\Psi}$ via Lemma 4.5.5 is

$$\det(\bar{\Psi}) = \det(A)\det(D - CA^{-1}B). \tag{4.135}$$

In the previous section we have already shown that the columns of the sub-block A form a linear independent set. Therefore $\det(A)$ is non-zero. Suppose we first consider the limit case where $\delta_1 = \delta_2 = \delta_3 = 0$. At this limit $CA^{-1}B = 0$. Hence the limit form of $\bar{\Psi}$ is

$$\det(\bar{\Psi}_{\text{lim}}) = \det(A)\det(D). \tag{4.136}$$

We also know that the columns of the sub-block D are linearly independent (via the linear independence argument based on the 4th and 6th entries of ω_4 and ω_5) and so $\det(D) \neq 0$. Therefore $\det(\bar{\Psi}_{\text{lim}}) \neq 0$, implying that the columns of $\bar{\Psi}_{\text{lim}}$ are linearly independent. Since the set of full-rank matrices is open and dense in the set of all square matrices, one can say that for sufficiently small $\delta_1, ..., \delta_3$ the columns of $\bar{\Psi}$ will be linearly independent, based on the continuity argument.

The allowable size of $\delta_1, ..., \delta_3$ for $\bar{\Psi}$ to be full rank is dependent on several factors, in particular the difference in magnitude between the δ's and N. By using a bounding argument on the element values in $\bar{\Psi}$ one can show that a lower bound on $|\det(D - CA^{-1}B)|$ is

$$N^2(k_4 - k_5) - (7\delta N + 49\delta^2)\triangle k, \tag{4.137}$$

where $\triangle k$ is the upper bound on the difference in k values, *i.e.*

$$\triangle k = \max(|k_i - k_j|), \tag{4.138}$$

where $1 \leq i, j \leq 5, i \neq j$.[††]

From the above equation (4.137) one can see that for a N that is sufficiently larger than δ, the $N^2(k_4 - k_5)$ term dominates and so $|\det(D - CA^{-1}B)| > 0$.

[††]See Appendix E for assumptions and extended details of the bounding argument.

Typically N is at least an order of magnitude larger than δ on the surface of the palm which is enough for $|\det(D - CA^{-1}B)| > 0$. Details of this result can be found in Appendix E.

Choosing $\{\omega_6\}$

Choosing the last sample point ω_6 to make Ψ a linearly independent set is easy. With the exception of sample points on $\mathcal{L}_{\text{light}}$, $\mathcal{L}_{\text{long}}$ and $\mathcal{L}_{\text{horiz}}$, almost all other sample points on the rigid body will have a non-zero $\phi_{i,2}$ entry. Adding the corresponding ϕ_i of any of these points to the set Ψ will immediately increase the rank of $\hat{H}$ from 5 to 6, thereby achieving full rank.

4.5.5 A Positive Definite $\hat{H}$ for the Articulated Hand

Previously it has been shown that one can choose a linearly independent set Ψ on the palm. Then extending the positivity argument to a 26-DOF articulated body is straightforward. Sample points chosen on the palm do not contribute to the partial derivatives of the segments further down the kinematic tree of the hand, *i.e.* the digits. Hence for the hand's 26-DOF case, Ψ will have the following form

$$\Psi := \{\phi_1, ..., \phi_6\} = \{ \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ \delta* \\ * \\ \delta* \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ * \\ * \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} * \\ 0 \\ * \\ * \\ * \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \\ 0 \\ \vdots \\ 0 \end{bmatrix} \}, \tag{4.139}$$

where $\phi_{i,7}, ..., \phi_{i,26} = 0$. Suppose we choose a couple of sample points ω_7, ω_8 on the MCP segment of the index finger such that their corresponding ϕ_i's can be shown to be linearly independent of each other based on the $7th$ and $8th$ entries only. Adding their corresponding ϕ_i's to Ψ gives

$$\Psi := \{\phi_1, ..., \phi_8\} = \{ \begin{bmatrix} \ast \\ 0 \\ \ast \\ \delta\ast \\ \ast \\ \delta\ast \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ 0 \\ \ast \\ \delta\ast \\ \ast \\ \delta\ast \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ 0 \\ \ast \\ \delta\ast \\ \ast \\ \delta\ast \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ 0 \\ \ast \\ \ast \\ \ast \\ \ast \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ 0 \\ \ast \\ \ast \\ \ast \\ \ast \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \vdots \\ 0 \end{bmatrix} \}, \tag{4.140}$$

and thus the rank of $\hat{H}$ increases to 8. Next we choose a sample point ω_9 on the PIP segment and another ω_{10} on the DIP segment of the index finger. Adding their corresponding ϕ_i's to Ψ gives

$$\Psi := \{\phi_1, ..., \phi_{10}\} = \{ \begin{bmatrix} \ast \\ 0 \\ \ast \\ \delta\ast \\ \ast \\ \delta\ast \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ 0 \\ \ast \\ \delta\ast \\ \ast \\ \delta\ast \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ 0 \\ \ast \\ \delta\ast \\ \ast \\ \delta\ast \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ 0 \\ \ast \\ \ast \\ \ast \\ \ast \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ 0 \\ \ast \\ \ast \\ \ast \\ \ast \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ \ast \\ 0 \\ \vdots \\ 0 \end{bmatrix} \}, \tag{4.141}$$

which consequently promotes the rank of $\hat{H}$ from 8 to 10. Note that the 11th-26th entries of $\phi_7, ..., \phi_{10}$ are 0. This is because the sample points on the index finger do not contribute to the partial derivatives of the other digits. In this manner, one needs to choose 6 points on the palm, 2 points at the MCP segment and 1 point each at the PIP and DIP segments for each digit to achieve a full rank $\hat{H}$ (and consequently a full rank H) for the 26-DOF articulated hand. This completes the proof of Proposition 4.5.2.

4.6 Summary

It has been shown that the tracking system can be viewed as a stochastic approximation problem. Due to the effects of noise from the cameras and sparse point sampling, the true gradient of the ideal cost function cannot be observed by the tracker. Only gradient estimates are available for use by the tracker's optimisation algorithm. Nevertheless the Robbins-Monro theorem states that under several conditions on the stepsize selection, the noise and the gradient function of the overall cost, it is possible for an optimiser (that uses gradient estimates) to converge in mean square and with probability 1 to the true optimum. This chapter has shown the extent to which these conditions are satisfied locally by the tracker. We have examined a couple of optimisation routines that satisfy the step-size condition, namely SGD and oBFGS. We have also argued that the noise in the system is unbiased and that its variance is uniformly bounded.

In addition we have shown that the tracker's overall cost function is relatively well-behaved. With the exception of a thin set of degenerate hand poses, a unique global minimum exists for the cost function. The proof for a unique global minimum is based on the set of active points that has shifted under a given perturbation and showing that this set increases $C_{\mathbf{x}*}$. Two exception scenarios have been highlighted where certain subsets of active points do not increase $C_{\mathbf{x}*}$. One exception pertains to active points belonging to the case 2 subset in the instance where the digit segment's main axis lies on the epipolar plane spanned by one end of the said segment. The other exception pertains to active points belonging to the case 6 subset in the instance where $\mathbf{x} \in \epsilon$ such that there exists a perturbation δ where the lighting patterns at the projections of the perturbed neighbourhood in the two camera view exactly match. Given the sizeable parameter space of $\mathbf{x}$, these exception cases are rare. Again, it needs to be reiterated that these exceptions pertain only to a *subset* of active points that shift under a perturbation. Even in poses for which these exceptions occur, it is still possible (and highly likely) that $C_{\mathbf{x}*}$ increases provided that active points belonging to other cases cause $C_{\mathbf{x}*}$ to increase.

Finally, it is also proven that the Hessian, H, of the ideal cost function at the optimal pose $\mathbf{x}^*$ is positive definite. The proof for the Hessian's positivity is based on the fact that H is at the very least positive semi-definite and that the rank of H can only increase with each sample point picked in the ideal cost evaluation. We have demonstrated that choosing a finite set of sample points of a certain constellation on the hand is enough to prove that H is positive definite at $\mathbf{x}^*$.

This involves choosing five sample points at different $\mathcal{L}_{\text{horiz}}$ trajectories belonging to different ellipsoidal regions of the palm, one sample point elsewhere on the palm and for each digit of the hand, two sample points at the MCP segment, one sample point at the PIP segment and one sample point at the DIP segment.

These properties are sufficient to satisfy locally (in the neighbourhood of the unique global minimum) the cost function conditions of the Robbins-Monro theorem. Hence, under appropriate noise and step size conditions, and under the assumption that the optimisation routine does not cause the tracker to leave the basin of attraction, the property of local convergence follows.

This theoretical local convergence property reinforces the small angle-change assumption often made in tracking. It suggests that the tracker will converge if the changes in joint angles from frame t to frame $t+1$ are small, such that the initial value of the tracker for frame $t+1$, *i.e.* the final estimate for frame t, lies in the region close enough to the minimum where the Hessian is positive definite.

On a macroscopic level, this chapter has illustrated one of possibly many ways of proving local convergence for a hand tracking system. It is conceivable that other hand/body tracking systems with similar characteristics (such as a cost function that is a sum of squared matching loss functions, a stereo or multi-view setup) can be shown to exhibit similar local convergence properties by applying the same methodology used in this chapter.

Chapter 5

Initial Tracking Results

This chapter presents the implementation details of the hand tracking system and the tracking results from various initial experiments. Section 5.1 describes the practical implementation issues regarding the hand model, such as hand model initialisation, handling self-occlusion and preventing finger collisions. In Section 5.2 issues related to the camera setup and the pre-processing of camera images are discussed. The last section of the chapter presents the initial tracking results. Experiments evaluating tracking accuracy on real image sequences containing typical hand gestures have been performed. Also investigated is the effect of the point sampling scheme on tracking performance.

5.1 Tracker Implementation

5.1.1 Hand Model Initialisation

The hand model used in the experiments is obtained from a 3D range scan of the author's hand [38]. Hence, the hand model is already a good fit to the tracked hand in the testing video sequences. Nevertheless one would need to re-adapt this hand model online in the initial frames should a different user with a different hand enter the scene.

One could adopt an Expectation-Maximisation-like strategy to estimate the shape of the tracked hand. Firstly one requires the user to initially keep the hand at a rough predefined dress pose *e.g.* an open palm. Parameters that are responsible for the rigid transformations of each hand segments are first optimised, possibly regularised at the predefined dress pose to ensure robustness. This is analogous to the E-step. Next these rigid transformation parameters are fixed,

and the tracker's cost function is then optimised over the parameters that deform the shape of the hand model (which are typically fixed during tracking). This is analogous to the M-step. These steps are repeated over several iterations. Alternatively, one may simultaneously optimise over the rigid transformation and deformation parameters as described in [49]. Yet another option would be to use active appearance models in the fitting process [23].

5.1.2 Visibility and Occlusion Handling

Each sample point varies in its degree of visibility. Figure 5.1 shows how the degree of visibility determines the types of cost evaluation that can be performed on the sample point.

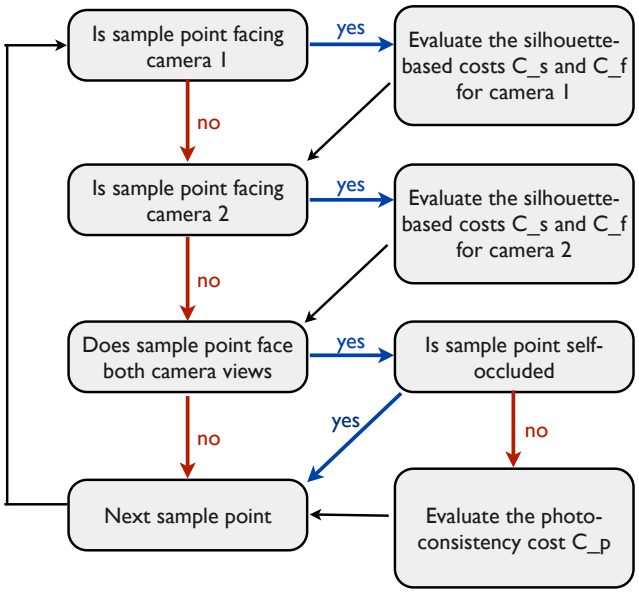

Figure 5.1: Cost evaluation for a sample point based on its visibility in the two camera views.

Every sample point is initially tested to see if it is facing each of the camera views. Let n_j and s_j be the surface normal and 3D coordinates of a given sample point expressed in the jth camera coordinate frame respectively. Then the sample point is classed as being visible in the jth camera view if $n_j \cdot s_j < 0$. Sample points that are facing the camera views contribute to the silhouette-based costs.

A sample point that is visible in both camera views is further tested for self-occlusion. A sample point that is not occluded by other parts of the hand in both camera views can be used to evaluate the photo-consistency cost. For

computational efficiency, a simple bounding volume model (see Figure 5.2) is used for the self-occlusion test. The rectangular faces of the bounding volume model are projected to the image plane and sorted into a list ordered by depth. Each sample point belongs to a particular rectangular face. To determine a sample point's visibility, the rectangular face that the sample point belongs to is checked to see if it is the nearest rectangular face to the camera view at the location of the sample point projection. If so, the sample point is considered visible. Otherwise, the sample point is self-occluded.

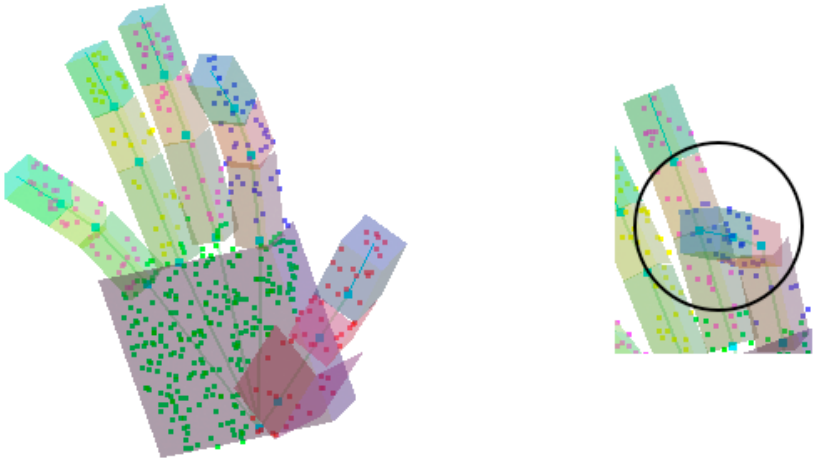

Figure 5.2: Left: Bounding volume model for the self-occlusion test. Right: Sample points on the middle finger (in red) occluded by the index finger (in blue) are discarded.

5.1.3 Finger Collisions

Provision against finger collisions is handled by introducing a penalty cost when adjacent fingers occupy the same volume in space. The Euclidean distances between each finger joint and its neighbouring joints are compared to check for finger collisions. Let $d_{i,j}$ be the Euclidean distance in 3D space between joint i and its adjacent joint j. Also, let $m_{i,j}$ be the allowable threshold distance between the joints i and j. Then the penalty cost due to the collision of joints i and j is given as

$$\mathbf{p}_{=} \begin{cases} A(e^{-b(\frac{d_{i,j}}{m_{i,j}})^2} + e^{-b}), & \text{if } d_{i,j} < m_{i,j} \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

A is a weighting factor and b is a scalar that controls the rate of decay. The use of exponentials in the penalty cost allows for better control in the gradient response that corrects for finger collisions. Figure 5.3 shows the typical shape of the finger collision penalty function.

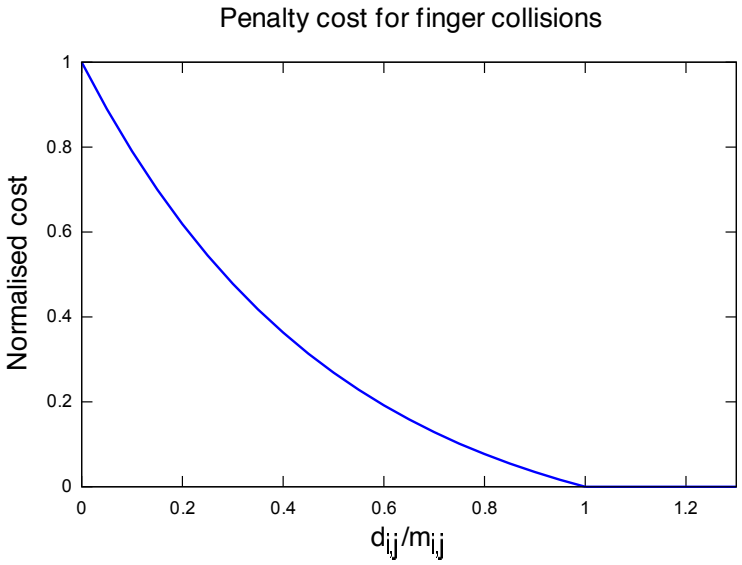

Figure 5.3: The cost penalty rises as the distance between joints become shorter. In this instance $b = 2$ and $A = \frac{1}{1-e^{-b}}$.

Note that the addition of the finger collision cost does not affect the unique global minimum of the overall cost function. This is because the digits of the hand do not share the same volume in space for all optimal hand poses. Hence the cost penalty due to finger collisions at the optimal pose will be zero. In addition, the Hessian of the the finger collision cost function at the unique global mnimum is at least positive semi-definite and so it does not affect the positivity of the Hessian of the overall cost function.

5.2 Camera Setup

A pair of firewire cameras are mounted on a stereo rig, pointed in a convergent manner towards the moving hand (see Figure 3.1 in Chapter 3). A directional light source above one of the cameras provides additional illumination to the scene. The background in the scene resembles a typical office environment.

Custom software is written using the standard libdc1394 library to control the video capture. It is important that the images captured by the two cameras

are synchronised. Image pairs that are not in sync are most problematic for fast hand movements. Any shift in hand pose during an out-of-sync image capture results in observing a deformed hand at best, or at worst, a physically impossible object. The libdc1394 library allows one to perform camera synchronisation via a software or an external hardware trigger.

Our firewire cameras are synchronised using an external hardware trigger to avoid potential software delays (*e.g.* CPU delays during high computation load and network delays) when the software trigger mode is used. An external Atmega128 microprocessor is used to generate a 30 Hz trigger pulse that synchronises the camera pair (see Figure 5.4).

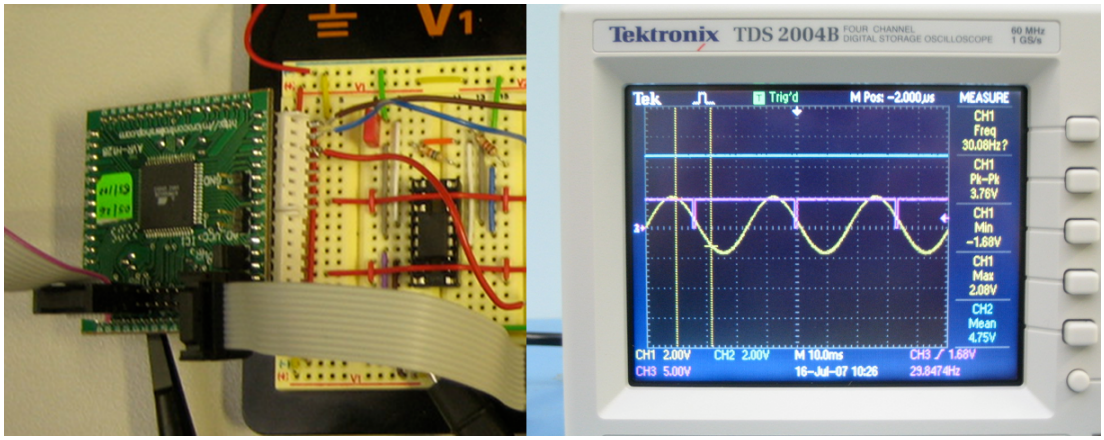

Figure 5.4: Left: The Atmega128 microprocessor that controls the synchronisation of cameras. Right: The trigger pulse (in pink) used for synchronisation.

Both cameras are calibrated via checkerboard images prior to tracking. The intrinsic and relative extrinsic parameters of the cameras are retrieved using the Caltech Camera Calibration Matlab Toolbox [10].

5.2.1 Segmentation of the Hand

The hand is segmented from the background to generate the hand silhouette. This is done via the simple but effective skin detection technique by Jones and Rehg [36]. A histogram model of the skin color is built from a couple of training skin images. To ensure robustness a histogram model of the background is also generated. A pixel is classifiied as a skin pixel if

$$\frac{P(\text{pixel}|\text{skin})}{P(\text{pixel}|\text{background})} \geq \epsilon, \tag{5.2}$$

where a threshold of $\epsilon = 0.5$ is empirically found to give a reasonably good trade-off between correct pixel classification and false positives/negatives in the segmented image.

The segmented image is typically noisy (see Figure 5.5). To produce a cleaner segmentation, the segmented image is first eroded to eliminate islands of noise. A dilation operator is then applied over the image to fill any holes in the hand silhouette. The erosion procedure is then repeated to produce the final image of the hand's silhouette.

The distance map required by the silhouette cost function is produced by running the Chamfer distance estimation algorithm [9] over the silhouette image. The OpenCV [1] library's implementation of the Chamfer distance estimation algorithm is used.

Figure 5.5: Left: Original image. Middle: Skin detection via a trained histogram model. Right: Noise removal and gap filling via dilation and erosion methods.

5.2.2 Colour Calibration of Images

Colour calibration of the images is required as differences in the cameras' RGB responses will manifest as noise bias in the photo-consistency cost function. The cameras' RGB profiles are calibrated using a Macbeth colour palette (see Figure 5.6).

Firstly, a pair of images of the same colour palette is taken by the two cameras in their video capturing positions. A random set of pixels are chosen at each colour square and the pixels' RGB values in this set are averaged to give the expected RGB value for the colour square. This procedure is repeated for the same colour square in the corresponding image. A typical relationship between the pairs of RGB values over all the colour squares is shown in the scatterplot in Figure 5.7. Note that to get a reasonably well-distributed set of colour points at varying intensities in the scatterplot, multiple images of the colour palette in different orientations have been taken. A non-linear model is used to characterise

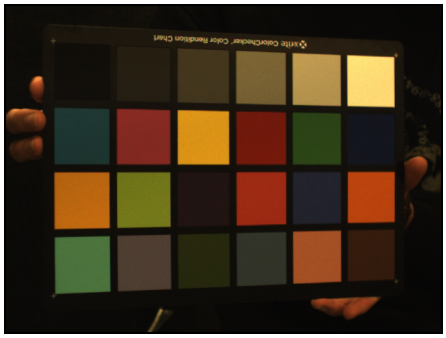

Figure 5.6: A Macbeth colour palette is used for the colour calibration of the cameras.

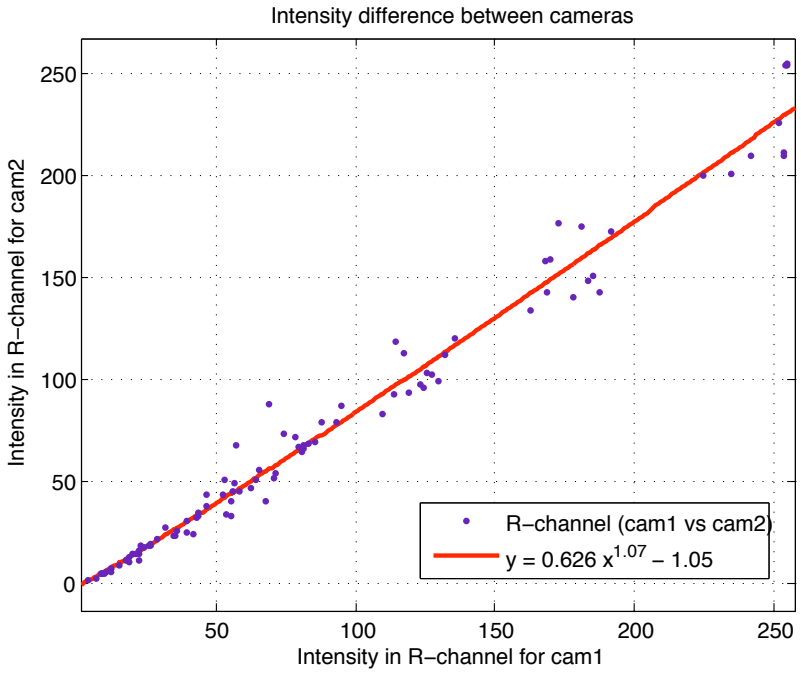

Figure 5.7: A scatterplot showing differences in the intensity responses (in the R-channel) of the cameras.

the relationship between the cameras' response for each of the R,G,B channels. Although the response observed in Figure 5.7 is fairly linear, this is not always guaranteed. Hence a broader non-linear model class (see Equation 5.3) is used to approximate the camera response relationship. Let r_j be the intensity of the R channel in the jth camera and let A, b, c be fitting parameters. Then the model equation for the R-channel is

$$r_1 = A(r_2)^b + c. \tag{5.3}$$

Note that each of the RGB channels are fitted separately. The set of fitted models is used to rectify the RGB response of one of the cameras so that it matches the RGB response of the other camera.

5.2.3 Image Gradient Approximation

Image gradients from the colour calibrated images and the silhouette-based distance maps are empirically evaluated by passing a derivative filter over these data images. The image gradients provide the partial derivatives needed to propagate the error gradients from the images to the parameter space. The filter used is a separable 5×5 filter, formed by the convolution of a 1D Sobel and a 1D Gaussian filter (see Figure 5.8). The filter version shown in Figure 5.8 calculates the derivative in the vertical direction of the image. The transpose of this filter is used to calculate the derivative in the horizontal direction of the image. Experiments indicate little difference between using a 5×5 and up to a 9×9 filter. Filters with a kernel size larger than 9×9 however give less accurate tracking results.

0.3086
0.6914
0
-0.6914
-0.3086

0.0545	0.2442	0.4026	0.2442	0.0545

0.0168	0.0754	0.1242	0.0754	0.0168
0.0377	0.1689	0.2784	0.1689	0.0377
0	0	0	0	0
-0.0377	-0.1689	-0.2784	-0.1689	-0.0377
-0.0168	-0.0754	-0.1242	-0.0754	-0.0168

Figure 5.8: Derivative filters for finding the image gradients.

5.3 Experiments

The tracking performance is evaluated on real image sequences of the moving hand. The hand movements examined in the testing sequence are based on typical hand gestures in a HCI context. Section 5.3.1 explores the types of gestures chosen.

5.3.1 Gestures

The hand gestures examined in the video sequences are geared towards user-interface control in a desktop environment, for example a free-form modelling software. There have been several studies in the Human Computer Interaction (HCI) community that look at the gesture types used in CPU control [89, 31, 60, 58, 61, 39]. Adopting Quek's classification scheme [60, 61], one can broadly classify hand gestures in the HCI context into two classes, *Manipulative* or *Semaphoric*. Gestures from each of these categories are explored in the video sequences.

Manipulative (or acting) gestures [58, 39, 61] are those for manipulating (virtual) objects and would be an important class of gestures for freeform modelling. Examples of manipulative gestures include rotating/moving/deforming a virtual object. Of the two gesture classes, manipulative gestures require greater tracking accuracy as the tracked movement directly translates to the amount of manipulative force acting on the virtual object. Experimental sequences containing manipulative elements include the pinch, pick-and-drop, and dial-turning sequences (see Section 5.3.2).

Semaphoric (or iconic) gestures [58, 39, 61] can be regarded as gestures that translate to computer interface commands. They are arbitrary in that the choice of a particular static pose or gesture to symbolise a command is up to the discretion of the designer. For example, an open palm with aligned digits may represent a 'stop' command. Sign language is largely based on semaphoric gestures. Sequences containing semaphoric gestures include the pick-and-drop and the rotating palm sequences (see Section 5.3.2).

Often the intentions of the user are executed through a combination of manipulative and semaphoric gestures. As an example, take the pick-and-drop sequence (see Section 5.3.2) where a user wishes to move a selected object to another position. The picking motion where the fingers converge together is a symbolic gesture to command the computer to select a particular object. This is followed by a manipulative gesture to move the selected object to a new position in space. The extent of the object's movement is dependent on the length of the moving hand's trajectory in space. To signify task completion, a semaphoric 'release' motion is used to fix the object in the new location.

Choosing an appropriate gesture vocabulary is a compromise of various factors including ease of recognition by the tracking system and ergonomics of a gesture-based user interface [56, 78, 79]. In general, the elements of a gesture vocabulary should be distinct to avoid ambiguity [58], straightforward to perform by most

[79] but remain reasonably easy to track. An interesting biomechanical aspect in gesture vocabulary design that is often overlooked is user fatigue [31]. One common oversight is the use of large arm movements in the gesture vocabulary. Studies have shown that the user's arms quickly tire [56, 15] and so gestures involving such movements should be avoided. The gestures in the video sequences have been chosen in light of these factors.

5.3.2 Real Image Sequence Testing

The performance of the tracking system is evaluated over a selection of video sequences showing a real hand. These sequences contain aspects of both semaphoric and manipulative gestures, both of which can conceivably be applicable to a gesture interface for freeform modelling. Via these video sequences we investigate the tracker performance whilst varying the sampling schemes, the number of samples used, and the optimisiation algorithm of choice.

A rough starting pose is used to initialise the tracker at the start of each video sequence. The weighting factors used in the cost function are as follows.

Table 5.1: Weighting values for the cost components of $C_{\mathbf{x}^*}$

Cost Component	Weight
Silhouette cost	1.5
Filling cost	1
Photoconsistency cost	100

The optimisation algorithms tested in the hand tracking system are SG, SMD, oBFGS (see Section 3.5 for these three), the Gauss-Newton method (GN) and the Levenberg Marquardt method (LM). Optimisation via LM and GN has been included as a point of reference. Each optimisation algorithm has the same termination conditions; either when the cost reaches below a certain threshold or when a maximum number of iterations are reached, whichever comes first. A cost threshold of 10 has been empirically found to be a good stopping condition. The maximum number of iterations allowed is set to 20. This fairly low number is chosen to account for the fact that a real-time system would be limited in time for pose tracking.

In terms of computational time, an average of 0.7s are required for the image pre-processing routines per frame (*i.e.* the generation of silhouette distance maps and the convolution operations required for the image gradients). Additionally Table 5.2 and Figure 5.9 show the average time required for the optimisation routine to complete each frame under the stopping conditions mentioned previously. All timing values are based on computations running on an Intel Core 2 Duo 2.4 GHz processor. The computational time is slightly slower on an Intel P4 3.4 GHz processor by $\sim 10\%$.

Table 5.2: Average time required (s) per frame for the optimisation routine.

	No. of Sample Points			
Optimisation Algorithm	128	256	512	1024
SG	0.16	0.26	0.40	0.68
OBFGS	0.30	0.47	0.75	1.3
SMD	0.16	0.26	0.42	0.73
LM	0.61	1.2	2.3	4.1
GN	0.61	1.2	2.3	4.0

The main bottleneck for each of the optimisation schemes is the evaluation of the cost function (and subsequently the corresponding error gradient and Hessian vector product in the backward-propagation mode). Processing time grows in an approximately linear fashion as more sample points are used (see Table 5.2 and Figure 5.9). The Levenberg-Marquardt and the Gauss-Newton schemes are a lot slower than the other schemes due to the explicit evaluation of the approximate inverse Hessian. To apply LM (or GN), the overall cost function needs to be expressed as

$$C_{\mathbf{x}^*}(\mathbf{x}) = \frac{1}{2} f(\mathbf{x})^{\mathrm{T}} f(\mathbf{x}). \tag{5.4}$$

The Hessian approximation used in LM and GN is $(J_f^{\mathrm{T}} J_f + \mu \mathbf{I})$ and $J_f^{\mathrm{T}} J_f$, respectively, where $J_f(\mathbf{x})$ is the Jacobian of $f(\mathbf{x})$. $f(\mathbf{x})$ takes values in $\mathbb{R}^9$ due to the 3 channels in the photo-consistency cost function, 2 channels in the silhouette cost function and 4 channels in the filling cost function. Hence the explicit evaluation of the Jacobian requires an extra 9 backward-propagation passes per iteration thereby substantially increasing computational load.

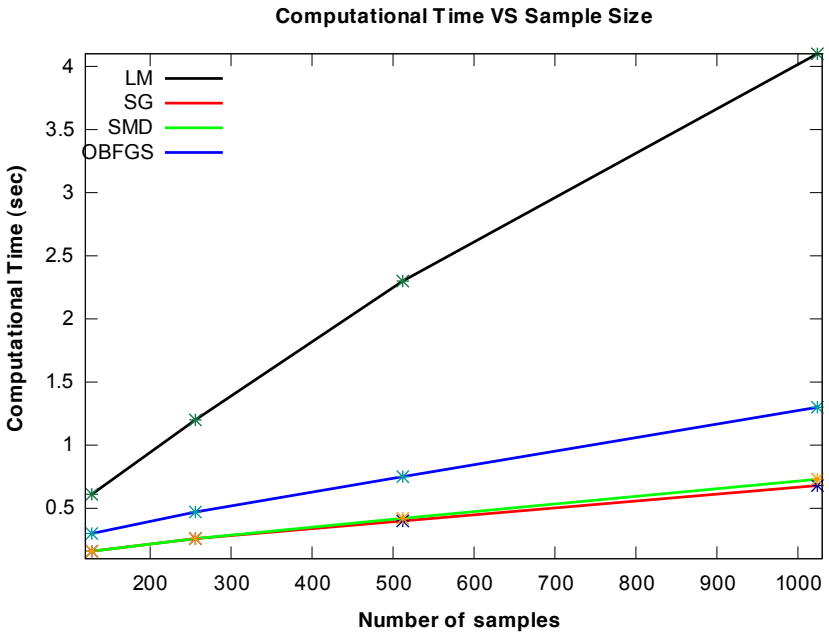

Figure 5.9: Average time (s) required per frame for the optimisation routines under different sample sizes.

As for oBFGS, it is about twice as slow as SMD or SG due to the fact that the update of the inverse Hessian requires 2 sets of forward and backward-propagation passes. Surprisingly, computational time for SMD and SG is very similar. This again highlights the fact that the cost evaluation is much more computationally expensive than the update routines of the optimisation algorithms.

Note that the current hand tracking system has not been optimised for speed. It is highly conceivable that the processing time can be drastically reduced by exploiting the parallelisable nature of the tracking pipeline. The evaluation of the cost function is parallelisable since the cost contributed by each sample point does not depend on neighbouring sample points on the hand surface. Therefore a sample point set can be subdivided and processed independently on separate CPU cores. The tracking system currently does not take advantage of this parallelisabilty. In fact only one of the two processor cores are used. By implementating a straightforward threading scheme, it is plausible that the processing times in Table 5.2 can be reduced by up to 50%.

In addition, image pre-processing procedures can be readily off-loaded onto the graphics card to be computed in parallel to the optimisation routine. One likely scenario would be to commence image pre-processing tasks for frame t on

the graphics card in parallel while the optimisation routine optimises over the previous frame $t-1$.

The following subsections explore the tracking performance on real video sequences in detail. Note that videos of tracking results for each of these sequences can be found on the accompanying CD-ROM.

Sequence 1 - Finger Flexion

This sequence involves the flexion of the individual digits followed by the simultaneous flexion of all fingers. Figure 5.10 shows the cost per frame for SG, SMD, and oBFGS. 1024 samples are used in the actual tracking.

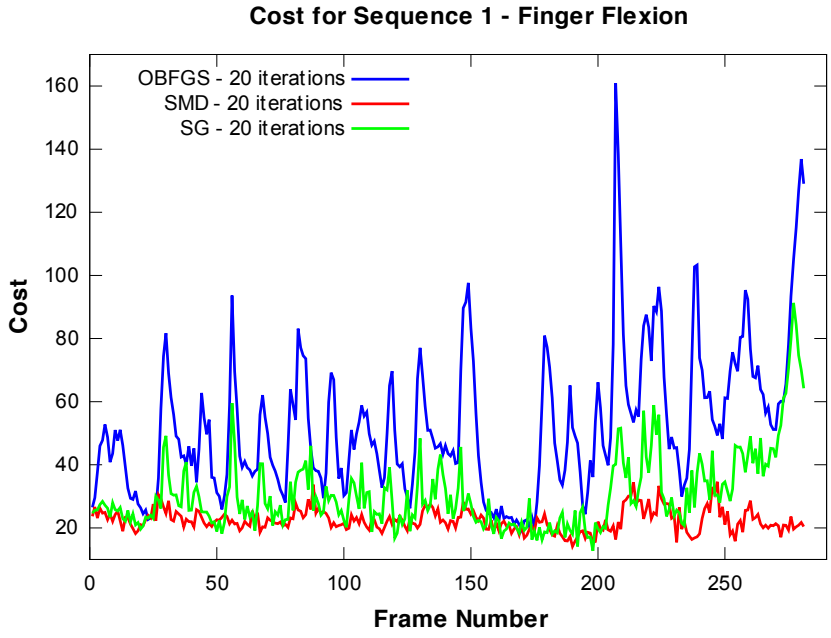

Figure 5.10: Cost per frame for finger flexion. A maximum of 20 iterations were allowed.

Both SG and oBFGS exhibit drifting errors as evidenced by the increasing trend in cost towards the end of the sequence. This error drift is the result of the allowable number of iterations per frame being too low. A 20 iteration threshold is not adequate for SG and oBFGS to converge close enough to the optimal position. One can compensate by setting a larger initial step size or a lower rate of step size decay for both SG and oBFGS. However additional testings indicate that doing so makes the tracking more 'shaky' and unstable as the noise in the system is not being dampened quickly enough.

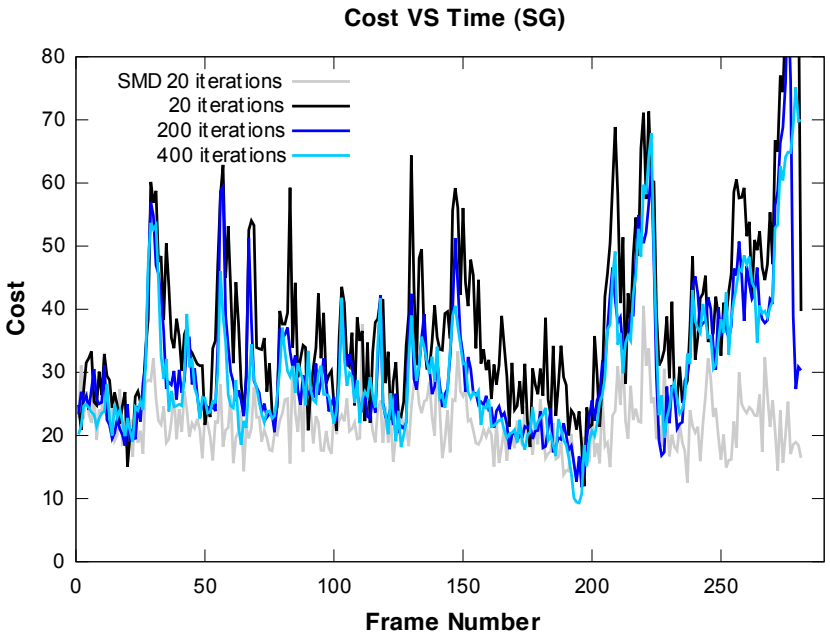

Figure 5.11: Cost per frame for SG for different maximal iteration counts. The tracking result obtained using SMD is shown in grey as a reference.

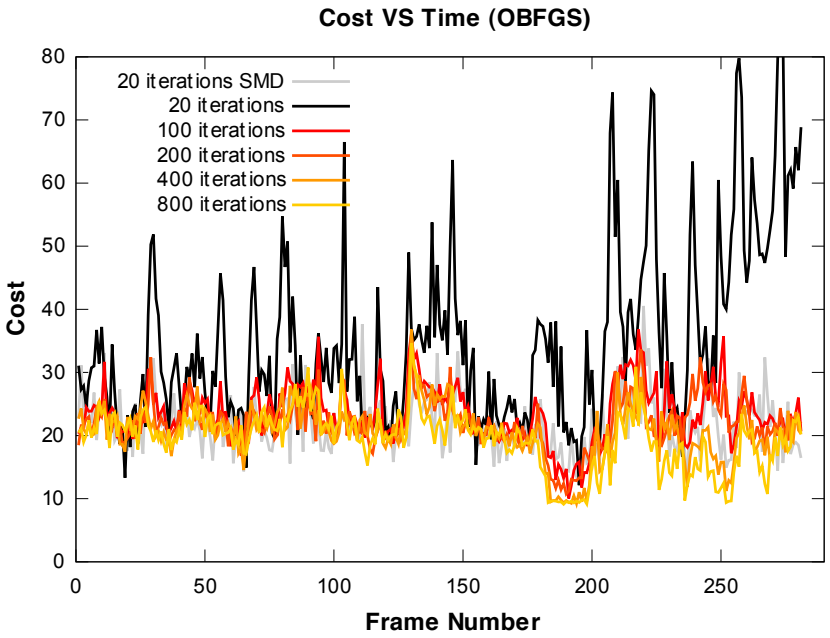

Figure 5.12: Cost per frame for oBFGS for different maximal iteration counts. The tracking result obtained using SMD is shown in grey as a reference.

To show that this increasing trend in cost is indeed due to insufficient iterations, the experiments have been repeated for SG and oBFGS with a larger number of allowable iterations per frame (see Figures 5.11 and 5.12). Here one can see that the increasing trend in cost disappears upon an increased number of allowable iterations. For the remaining video sequences only the results of SG and oBFGS under a maximum of 400 iterations will be shown. The tracking results of SG and oBFGS under a maximum of 20 iterations are simply not comparable to that of SMD.

The frames that exhibit a poor pose estimate include frames 86 and 139 for oBFGS (see Figure 5.13), frame 86 for SMD (see Figure 5.14) and frames 86, 139, 236 and 242 for SG (see Figure 5.15). Tracking using SMD appears to perform the best.

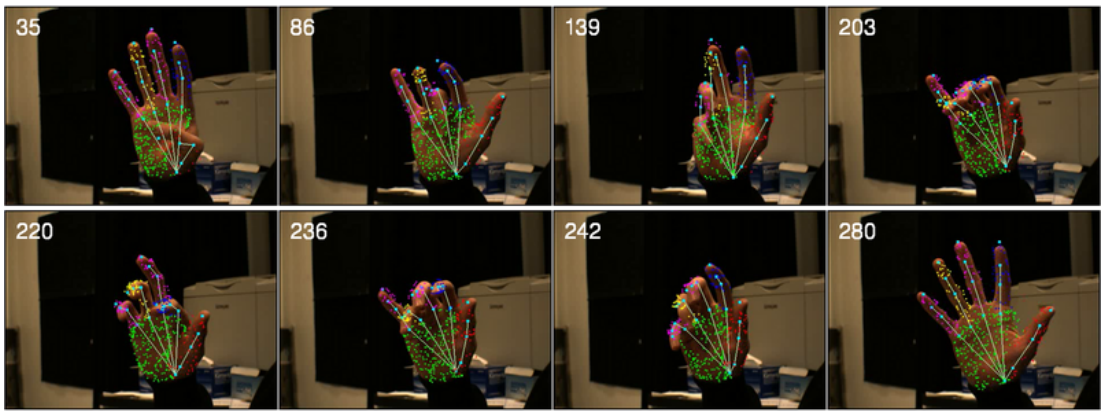

Figure 5.13: A selection of frames from Sequence 1 - Finger Flexion using oBFGS.

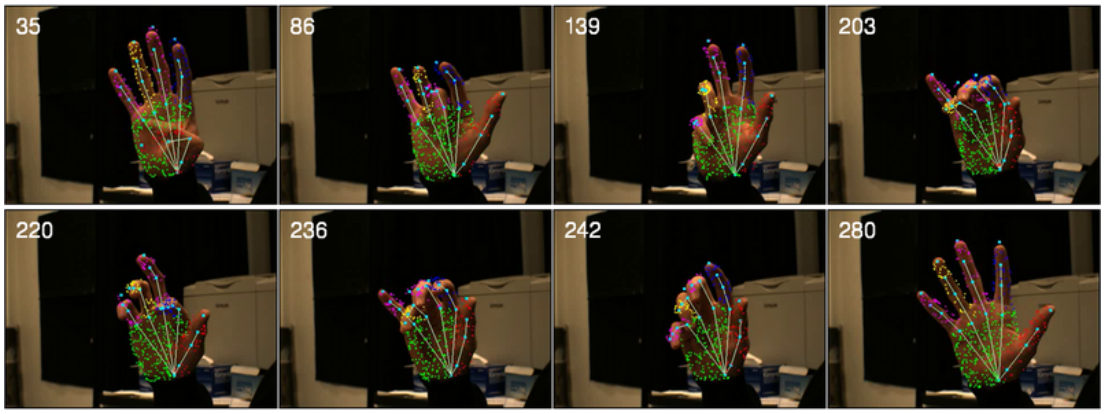

Figure 5.14: A selection of frames from Sequence 1 - Finger Flexion using SMD.

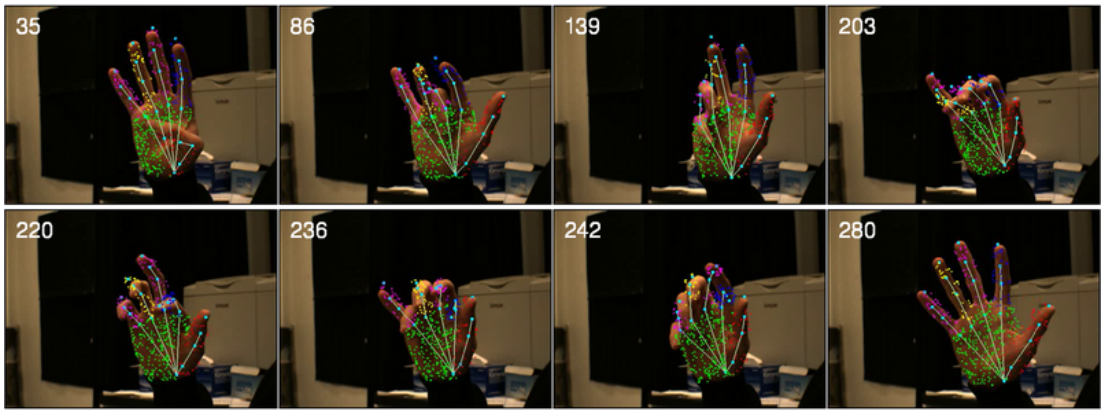

Figure 5.15: A selection of frames from Sequence 1 - Finger Flexion using SG.

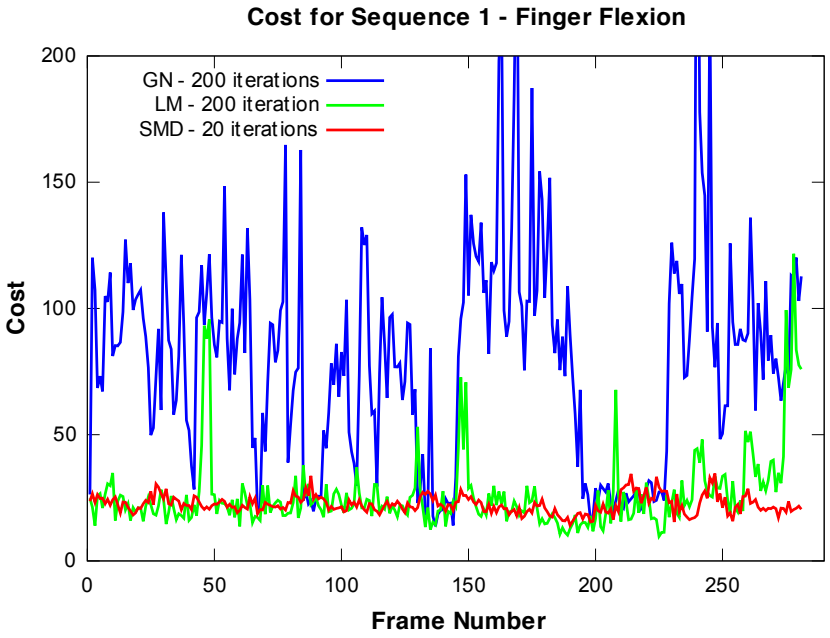

Figure 5.16: Cost per frame for finger flexion for the Levenberg-Marquardt and Gauss-Newton methods. A maximum of 200 iterations were allowed.

The Gauss-Newton method (GN) has given the worst result out of all the optimisation algorithms tested (see Figures 5.16 and 5.17). Despite spending 200 iterations at each frame, the tracking accuracy fluctuates wildly from frame to frame. This is expected because the Gauss-Newton method is designed for fast convergence for a deterministic rather than a stochastic cost function. There is no provision for handling noise in GN; the Hessian approximation ${J_f}^{\mathrm{T}} J_f$ and

the gradient estimates are used by the Gauss-Newton method as if they are true deterministic values. This leads to pose estimates that oscillate around but never converge to the optimal pose.

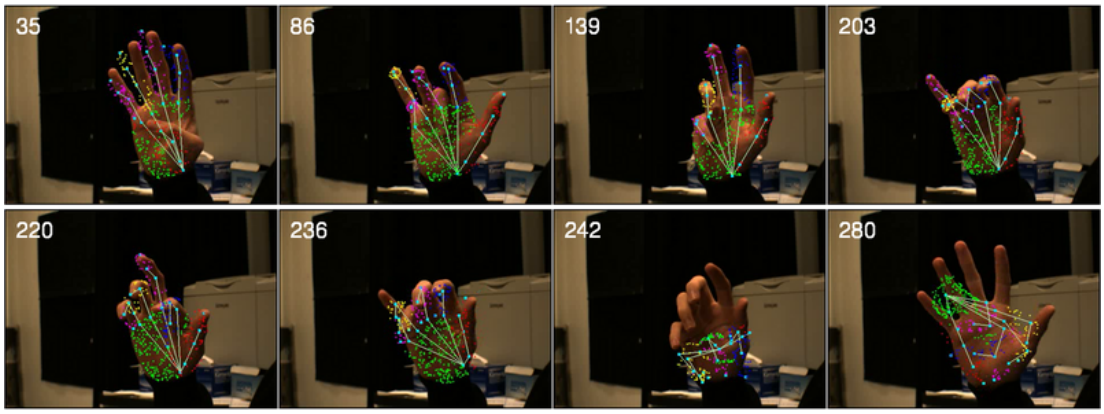

Figure 5.17: A selection of frames from Sequence 1 - Finger Flexion using GN.

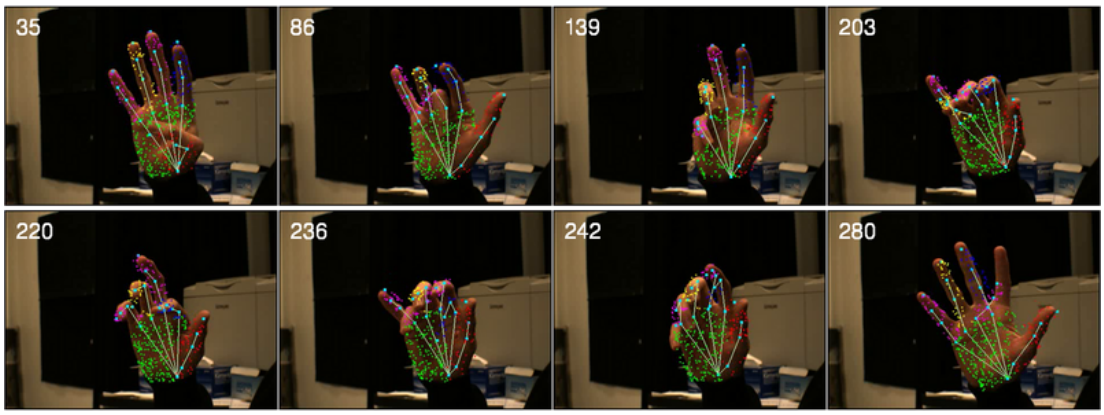

Figure 5.18: A selection of frames from Sequence 1 - Finger Flexion using LM.

The results obtained using the Levenberg-Marquardt method (LM) are much better than the GN method (see Figures 5.16 and 5.18), although not as good as the stochastic optimisation methods *i.e* SMD, oBFGS and arguably SG. The generally low error highlights the effect of the trust region at work in LM. The stochasticity of the cost function has the effect of making the $J_f^{\mathrm{T}} J_f$ estimate appear inaccurate from one iteration to the next, thereby causing the trust factor μ in LM to dampen the step size. Nevertheless LM is prone to bursts of poor tracking at intermittent frames. This is because fundamentally the trust region implemented in LM is not designed to actively dampen noise but rather to take

the largest possible step direction in the current step given its level of trust for the accuracy of the Hessian approximation $J_f{}^{\mathrm{T}} J_f$. Should the noise abate temporarily at certain iterations, LM can be misled to believe that its Hessian approximation is accurate thereby increasing the trust region. A larger trust region leads to LM taking a larger, often reckless, update step. Having said that, there exists a provision for overshooting in LM where an update step is only taken when the resulting cost becomes smaller. For a deterministic setup this would prevent LM from taking an update step that is too large. However, for a stochastic cost function, what may appear to be a valid step leading to a lower cost for one particular realisation of the stochastic cost function may not be a valid step for another realisation of the stochastic cost function. This compromises the effectiveness of the guard for overshooting, thereby leading to the bursts of tracking inaccuracy in the sequence.

Sequence 2 - Dial Turning

In this sequence the hand forms a grip over an imaginary dial and completes a turning motion. The error peak at frame 100 (see Figure 5.19) corresponds to a poor fit of the palm in the sequence (see Figures 5.21, 5.22, 5.23).

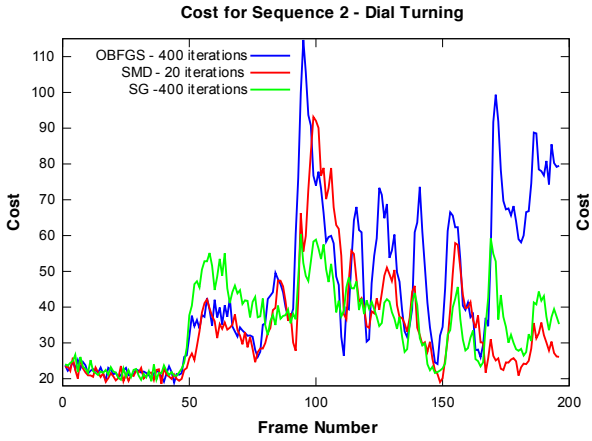

Figure 5.19: Cost per frame for dial turning motion (oBFGS, SMD, SG).

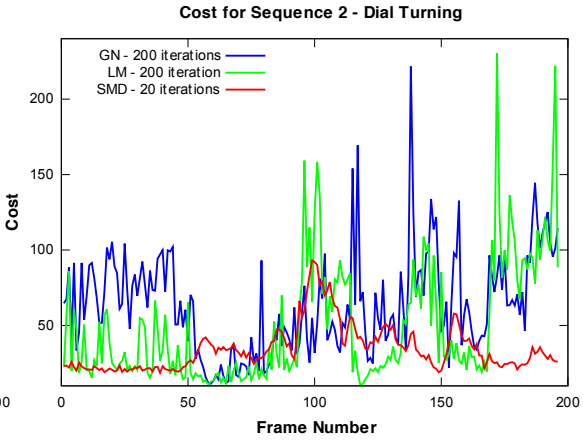

Figure 5.20: Cost per frame for dial turning motion (LM, GN).

In frame 112 the index and the middle finger become crossed over for both SMD and SG. This part of the sequence is particularly tricky since the fingers are close together and overlapping. However both SMD and SG do recover and the final open palm pose is reached. The high cost for oBFGS at the end of the sequence is due to the crossing over of the little and ring fingers.

Similar to what is observed in the flexion sequence, GN exhibits poor tracking results. The cost values in the first 50 frames (see Figure 5.20) under GN and, to a lesser extent, LM are high and tend to fluctuate wildly from frame to frame. The poor tracking accuracy even for this relatively stationary portion of the video sequence illustrates the lack of proper noise damping in both algorithms (see Figures 5.24 and 5.25).

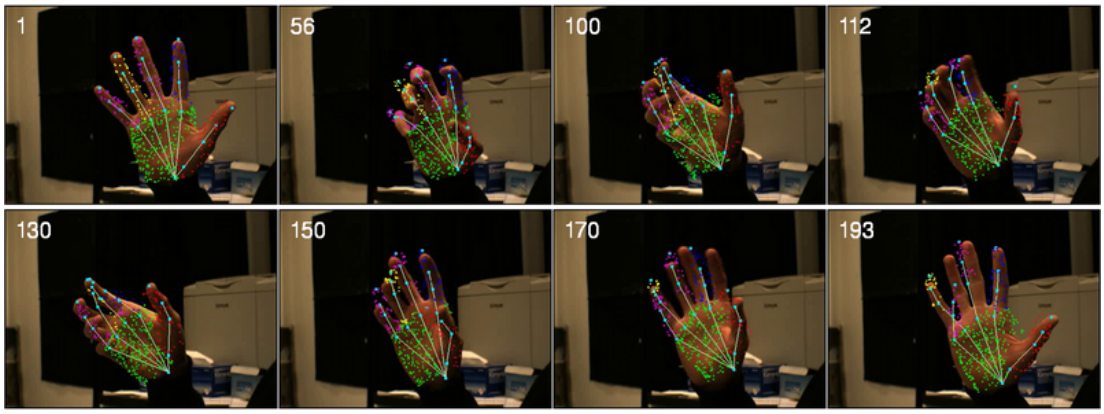

Figure 5.21: A selection of frames from Sequence 2 - Dial Turning using oBFGS.

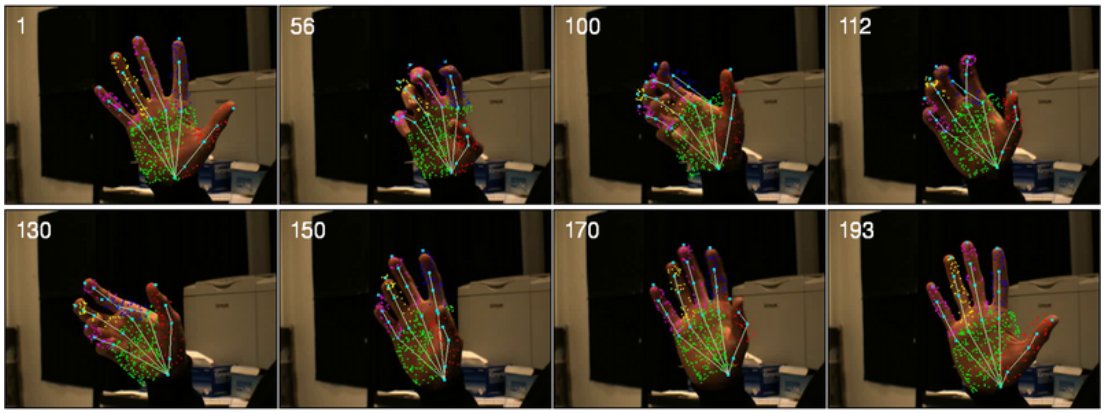

Figure 5.22: A selection of frames from Sequence 2 - Dial Turning using SMD.

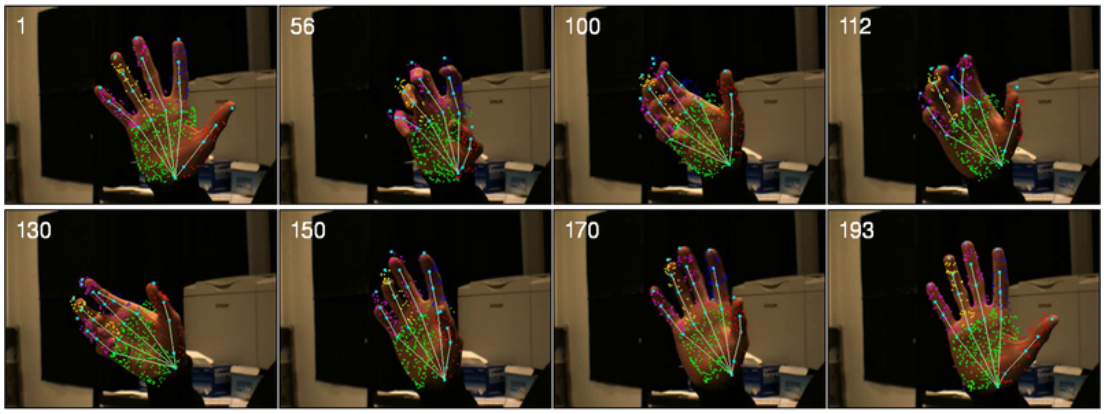

Figure 5.23: A selection of frames from Sequence 2 - Dial Turning using SG.

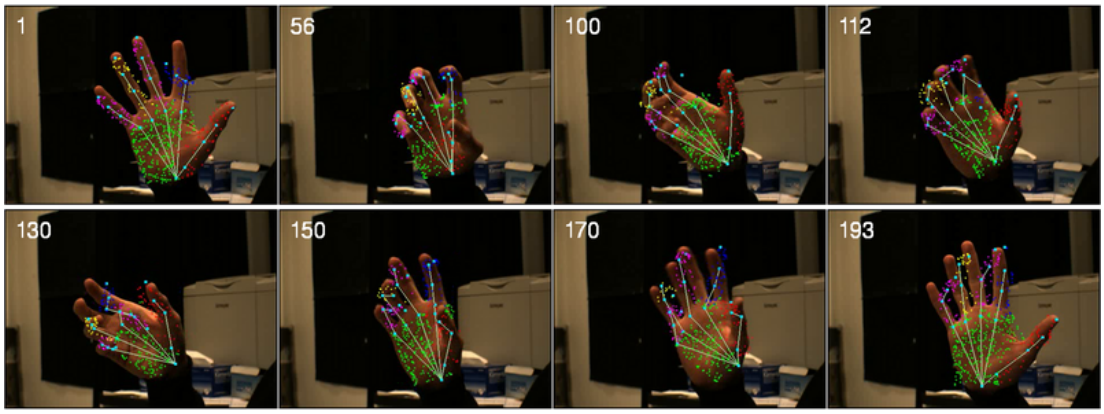

Figure 5.24: A selection of frames from Sequence 2 - Dial Turning using GN.

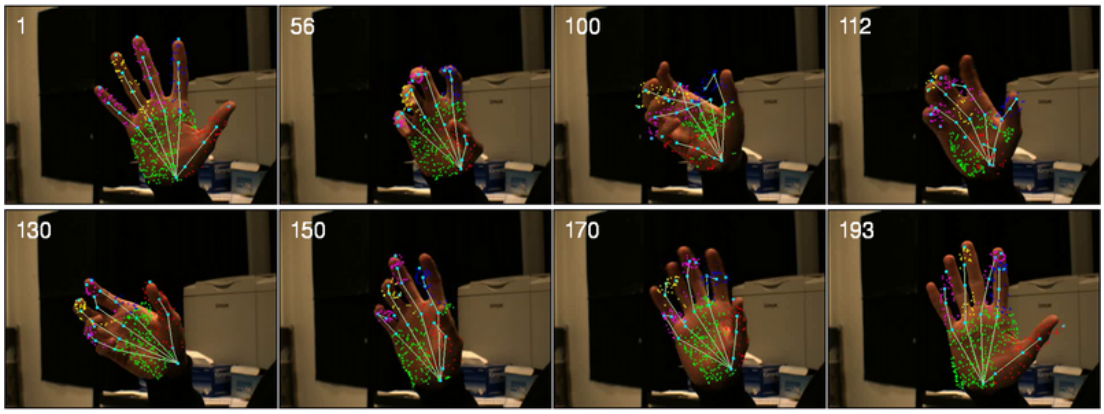

Figure 5.25: A selection of frames from Sequence 2 - Dial Turning using LM.

Sequence 3 - Pinching

In this sequence the hand forms a pinching gesture. The rough pinching motion is captured but there is a misalignment of the fingers. The misalignment starts at frame 31 (see Figures 5.28, 5.29, 5.30) where the middle finger occludes the index finger causing difficulty for the tracker to distinguish between the two fingers. Instead of the pinching motion occurring between the thumb and the index finger, the pinching motion in the subsequent frames occurs between the middle finger and the thumb. One would think that adding a motion predictor to the tracker may resolve this issue. This will be discussed in Chapter 6. Similar to the dial turning sequence, both SG and SMD do recover to the open palm pose towards the end. The high cost observed at the end for oBFGS (see Figure 5.26) is due to the crossing over of the index and middle fingers.

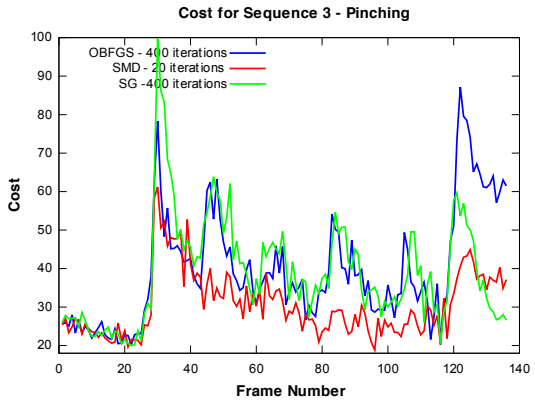

Figure 5.26: Cost per frame for the pinching motion. (oBFGS, SMD, SG).

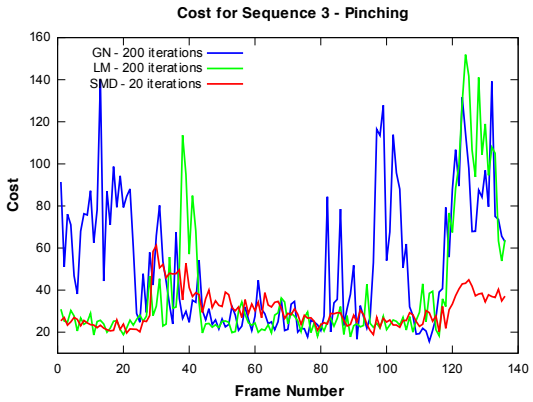

Figure 5.27: Cost per frame for the pinching motion (LM, GN).

Interestingly, the tracking results for the pinching motion using LM is correct (see Figure 5.27) in that the middle and index fingers are not mixed up at all. Similar to SMD, SG and oBFGS the curling of the other fingers remains inaccurate. Unfortunately under LM, the tracker does not recover well back to the open palm pose at the end of the sequence.

The tracking results for GN are again rather poor (see Figure 5.32).

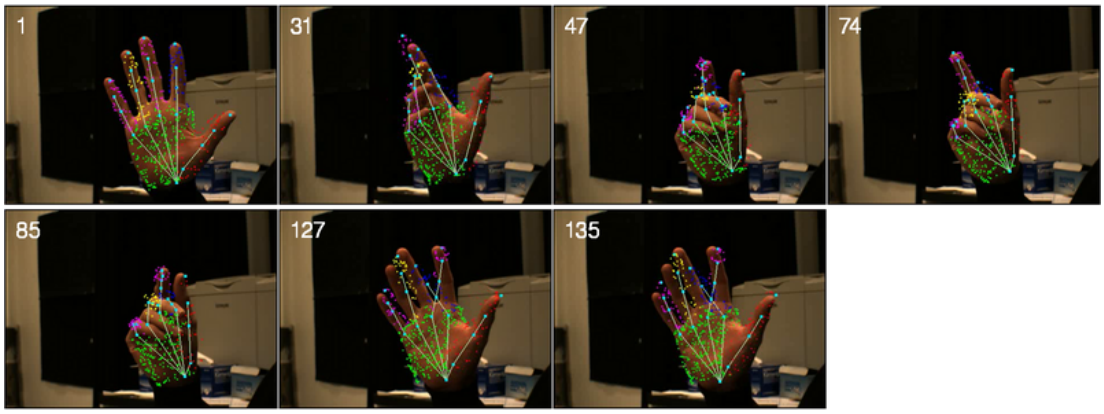

Figure 5.28: A selection of frames from Sequence 3 - Pinch using oBFGS.

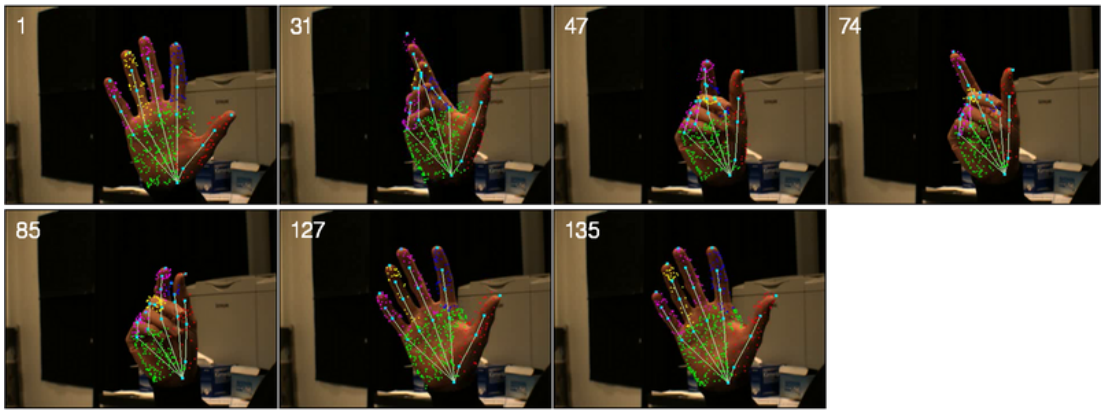

Figure 5.29: A selection of frames from Sequence 3 - Pinch using SMD.

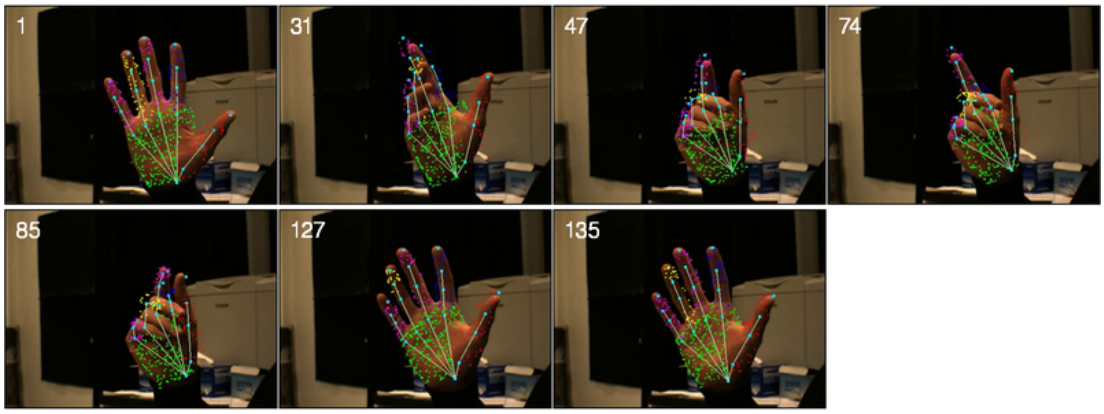

Figure 5.30: A selection of frames from Sequence 3 - Pinch using SG.

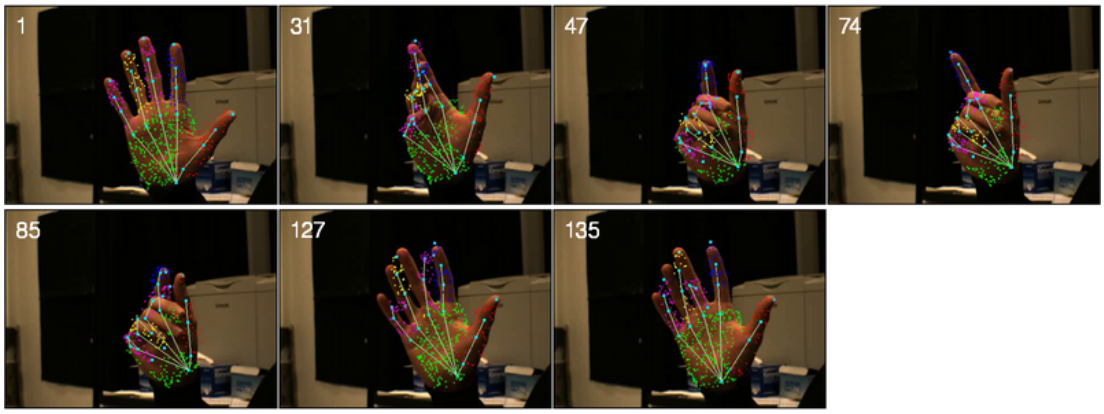

Figure 5.31: A selection of frames from Sequence 3 - Pinch using LM.

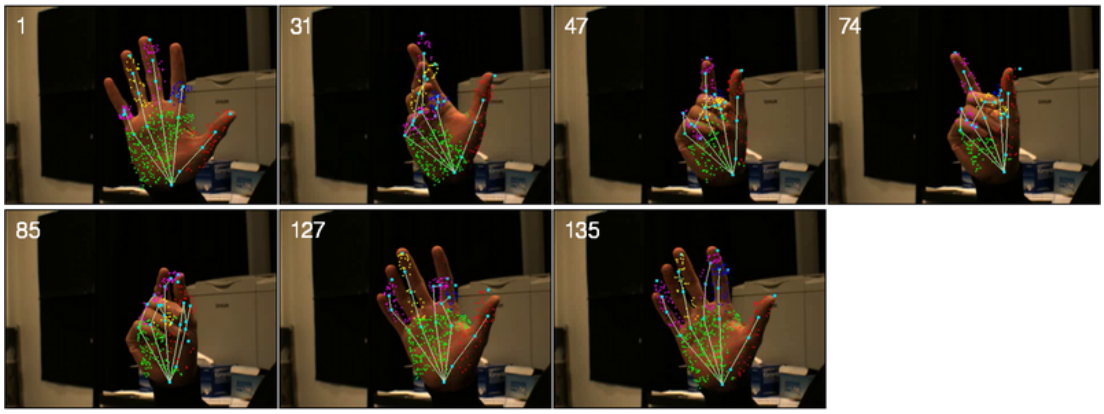

Figure 5.32: A selection of frames from Sequence 3 - Pinch using GN.

Sequence 4 - Pick and Drop

In this sequence the hand grabs an imaginary object and drops it at another location. The main problem in this sequence appears to be that the joints of the fingers are not bending quickly enough, causing the estimated pose to lose form. Frames 35 and 48 (see Figures 5.35, 5.36, 5.37, 5.38) exhibit this phenomenon; despite the pose lying within and filling the hand silhouette completely, the alignment of the fingers is incorrect. The high error peaks for oBFGS, SG and LM at frame 66 (see Figure 5.33) are due to the misalignment of digits. Nevertheless, the tracker under oBFGS, SMD, LM and SG recovers back to the open palm pose towards the end. This is not the case for GN.

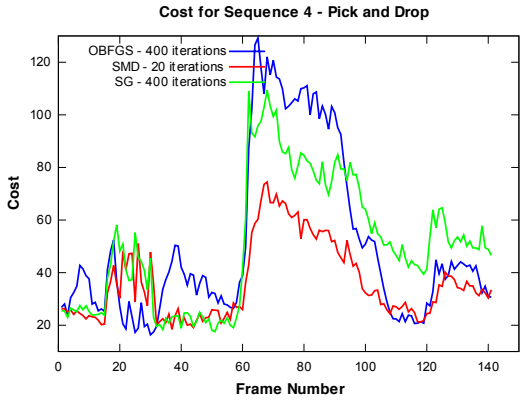

Figure 5.33: Cost per frame for the pick and drop motion.

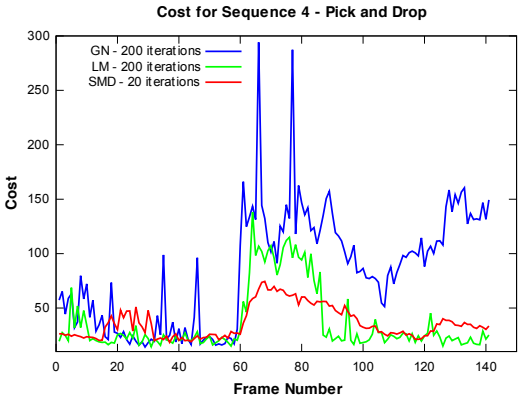

Figure 5.34: Cost per frame for the pick and drop motion.

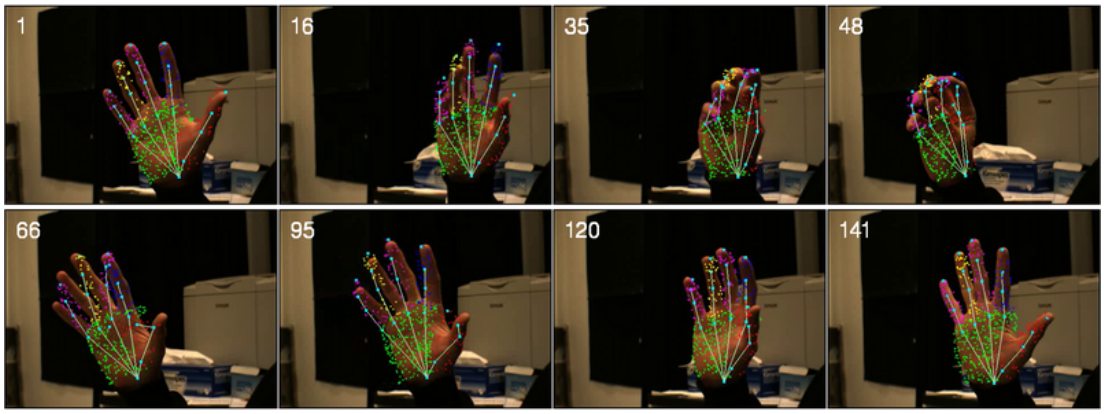

Figure 5.35: A selection of frames from Sequence 4 - Pick and Drop using oBFGS.

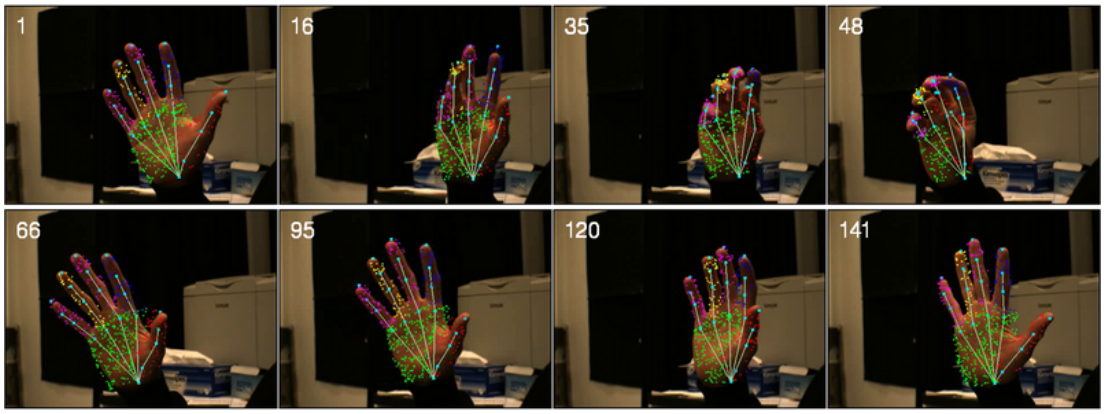

Figure 5.36: A selection of frames from Sequence 4 - Pick and Drop using SMD.

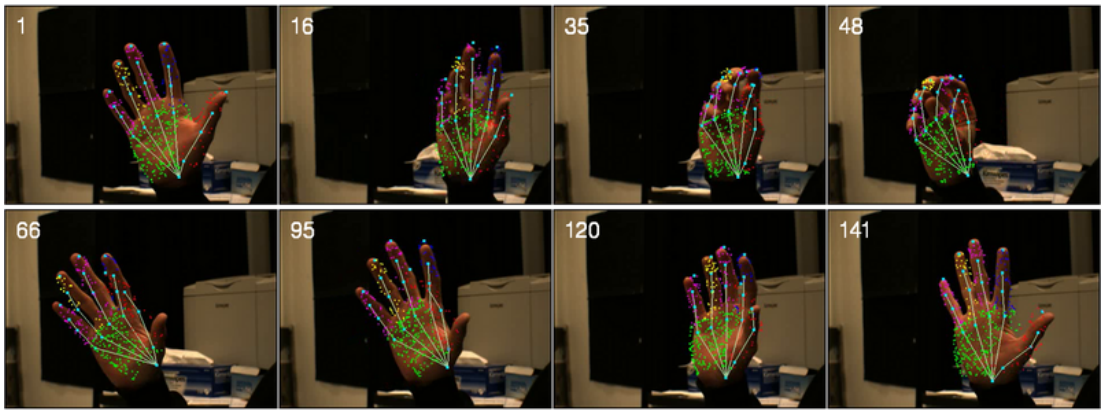

Figure 5.37: A selection of frames from Sequence 4 - Pick and Drop using SG.

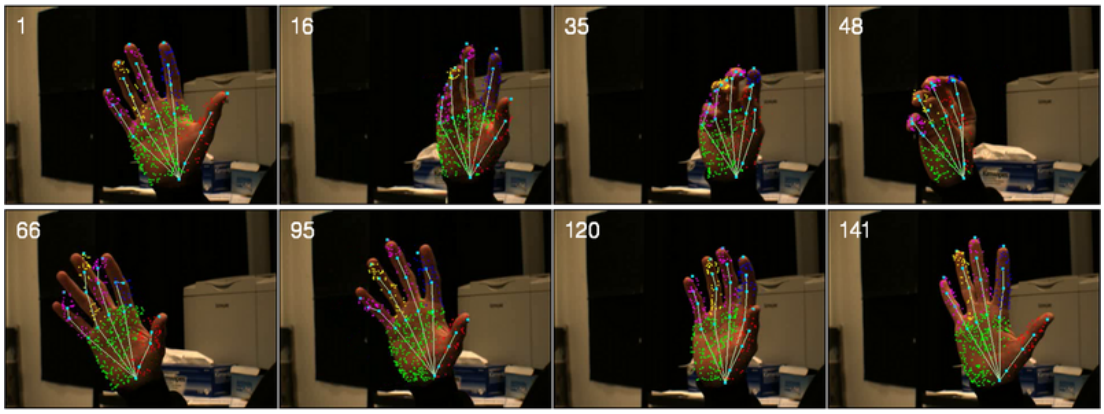

Figure 5.38: A selection of frames from Sequence 4 - Pick and Drop using LM.

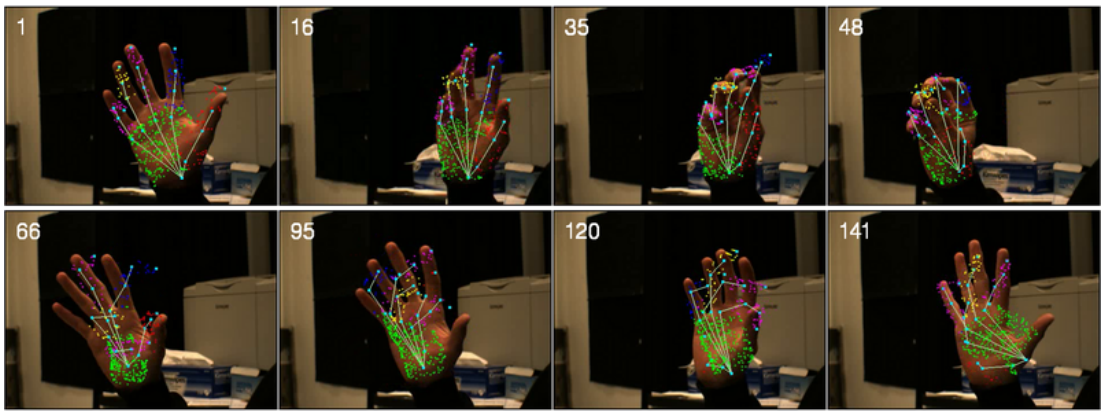

Figure 5.39: A selection of frames from Sequence 4 - Pick and Drop using GN.

Sequence 5 - Rotating Palm

In this sequence the palm flips in front of the camera, revealing the back of the hand before returning back to the original open palm pose. This is especially challenging as the hand motion undergoes severe self-occlusion, in particular at frames 27 and 65 where self-occlusion is the most severe due to the turning motion. The cost graph (see Figure 5.40) reflects this. An additional difficulty for these two frames is the motion ambiguity - without a prior on the past motion history, it is conceivable that the palm can flip either way. This is observed at frame 27 in Figures 5.42, 5.43, 5.44, 5.45, 5.46 in that oBFGS, SMD (20 iterations), SG, LM and GN all fail to follow through with the flipping motion. Temporal information in the form of motion prediction is required in such instances to help decide the switching direction. This will be discussed in Chapter 6.

However, it is interesting to note that SMD (at a maximum of 400 iterations) is able to transition through the first flipping motion at frame 27 and faithfully track the palm (see Figure 5.47). Unfortunately the algorithm is still unable to track the returning transition at frame 65. This further demonstrates the need for a motion predictor.

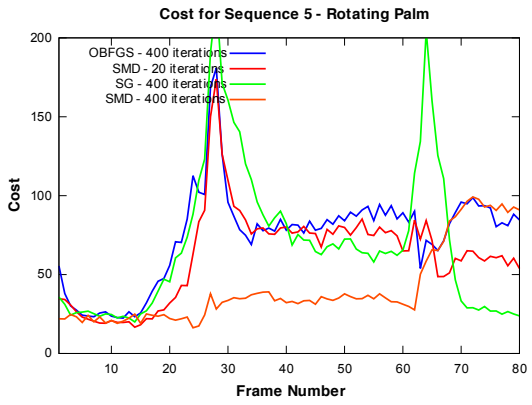

Figure 5.40: Cost per frame for the palm rotate motion (oBFGS, SMD, SG).

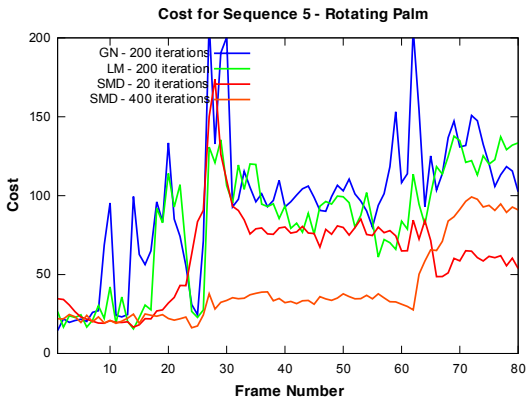

Figure 5.41: Cost per frame for the palm rotate motion (LM, GN).

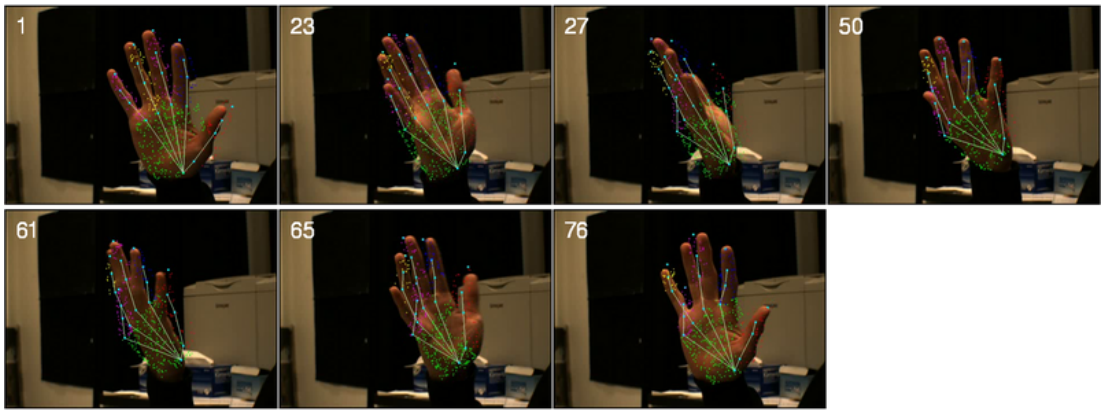

Figure 5.42: A selection of frames from Sequence 5 - Rotating Palm using oBFGS.

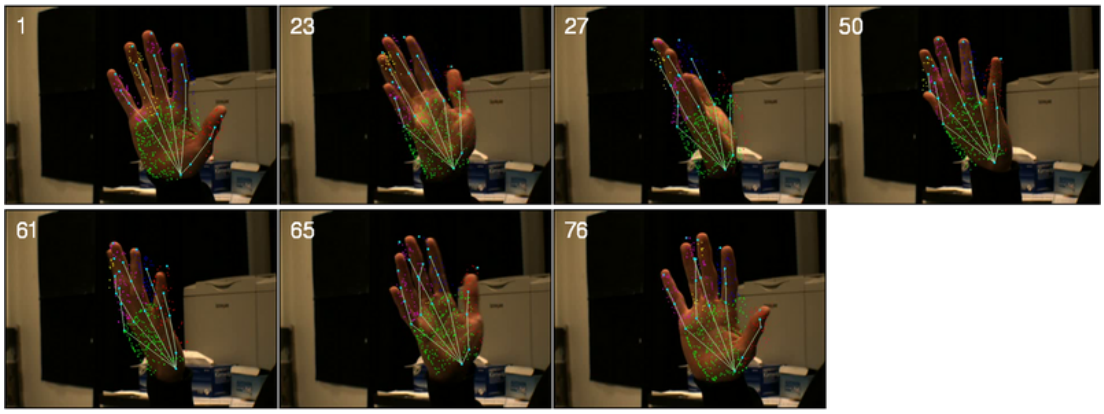

Figure 5.43: A selection of frames from Sequence 5 - Rotating Palm using SMD.

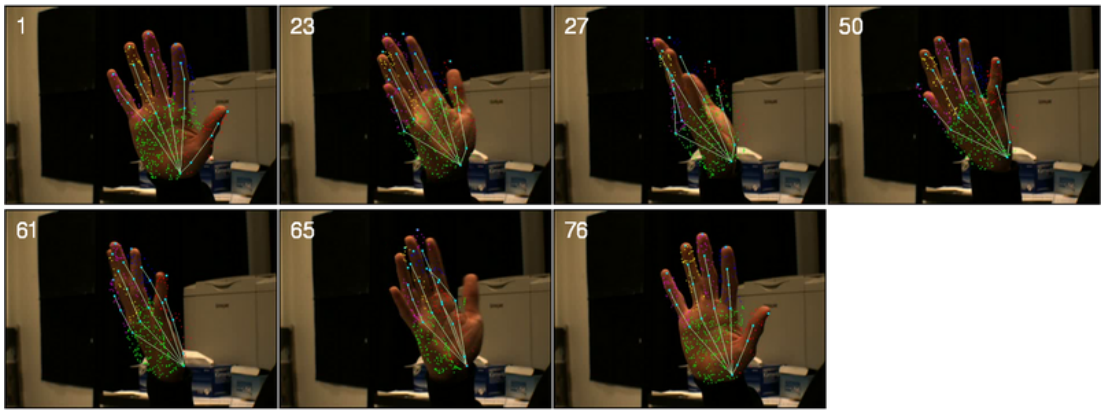

Figure 5.44: A selection of frames from Sequence 5 - Rotating Palm using SG.

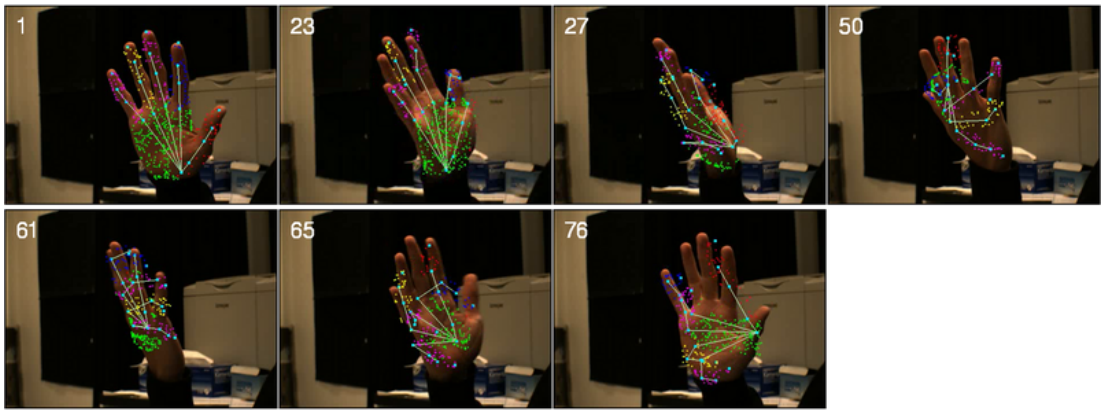

Figure 5.45: A selection of frames from Sequence 5 - Rotating Palm using LM.

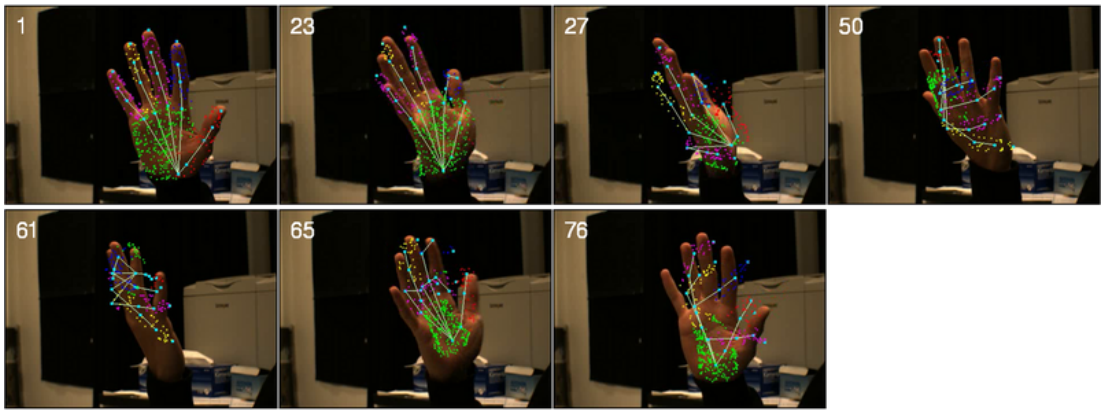

Figure 5.46: A selection of frames from Sequence 5 - Rotating Palm using GN.

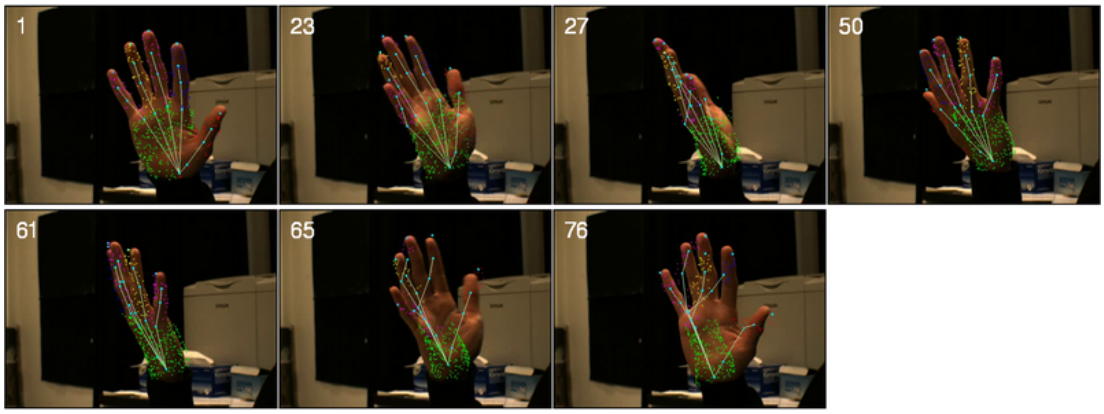

Figure 5.47: A selection of frames from Sequence 5 - Rotating Palm using SMD (max 400 iterations).

5.3.3 Sampling

As alluded to in Section 4.3, there remains the question of what makes a good sampling scheme for the purposes of tracking. It is possible to change the sampling scheme without affecting the position or the existence of the unique global minimum in the overall cost function. To do so requires the following restriction on the formulation of the sampling scheme: for every visible region of an arbitrary size on the hand surface, the probability of picking a sample point in that region is greater than 0. In other words, one is free to alter the sampling distribution on the visible region of the hand surface as long as there do not exist visible parts of the hand where the sampling scheme can never sample from. For example, a sampling scheme that does not sample from the PIP segment of the index finger would be invalid. Similarly a sampling scheme that only samples at the projected contours of the hand surface in the two camera views is also invalid.

Changing the sampling scheme can broaden or reduce the basin of attraction in which the tracker can converge locally. This interesting aspect of shaping the cost function is not addressed in the analysis in the previous chapter. There are endless ways of choosing a sampling scheme on the hand surface. The following are several sampling schemes that have been investigated.

1. Uniform Random: Randomly sample the hand surface, based on the distribution of vertices on the hand model surface.

2. Random-75-Palm: 75% of sample points are randomly chosen from the palm. The other 25% of sample points are chosen randomly from the remaining parts of the hand.

3. Random-50-Palm: 50% of sample points are randomly chosen from the palm. The other 50% of sample points are chosen randomly from the remaining parts of the hand.

4. Random-25-Palm: 25% of sample points are randomly chosen from the palm. The other 75% of sample points are chosen randomly from the remaining parts of the hand

5. Stratified-75-Palm: 75% of sample points are randomly chosen from the palm. Of the other 25%, an equal number of sample points are randomly chosen from each remaining hand segment.

6. Stratified-50-Palm: 50% of sample points are randomly chosen from the palm. Of the other 50%, an equal number of sample points are randomly chosen from each remaining hand segment.

7. Stratified-25-Palm: 25% of sample points are randomly chosen from the palm. Of the other 75%, an equal number of sample points are randomly chosen from each remaining hand segment.

8. Edge-weighted: Sample points near the projected edges/contours are more likely to be chosen than from the centre of the projected hand segment. This effect is achieved by first randomly sampling the hand surface, followed by a discarding pass where a sample point ω has a chance of being discarded based on the surface normal n at ω *i.e.*

$$\mathrm{P}(\omega \text{ is discarded}) = |n \cdot v|, \tag{5.5}$$

where v is the unit direction vector that points to ω from the camera origin.

These sampling schemes have been devised to test several hypotheses. One of the hypotheses is that it is important to get the orientation of the palm correct since it is the base of the hand's articulated structure. Errors in the palm orientation will propagate (and potentially magnify) to other segments in the kinematic chain. Hence the schemes with varying percentage of points sampled from the palm have been devised to test whether sampling more points on the palm makes for a more accurate palm orientation and subsequently a more accurate hand pose. The Random and Stratified schemes are also designed to test whether it is necessary to sample from each segment of the hand at each iteration. From a theoretical perspective it should not matter as long as each segment eventually gets sampled in at least once after any finite number of optimisation iterations, *i.e.* infinitely often in total. In practice however, it is not clear if this will affect performance since the number of iterations allowed is finite. Finally, the Edge-weighted sampling scheme is devised to test whether sample points taken close to the contours of the hand's 2D camera view projection are more important than sample points located at the centre of the hand. Intuitively one would believe that sample points close to the edge are more important since the silhouette and filling cost function predominantly affects these points.

The sampling schemes are tested on the finger-flexion sequence, with the tracker running SMD using 256 sample points. A lower number of sample points

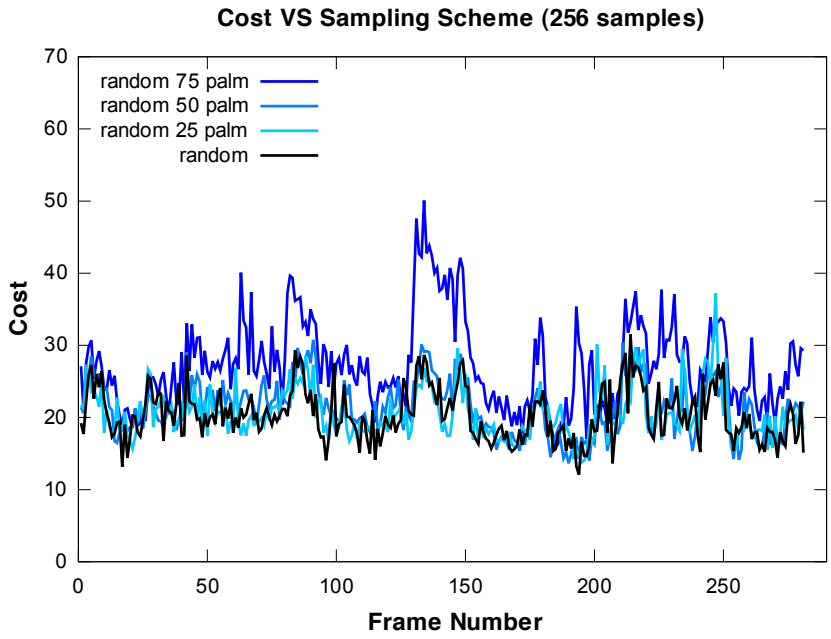

Figure 5.48: Cost variation under different random sampling schemes. Shown in black is the uniform random sampling scheme (based on the default point distribution on the hand model mesh).

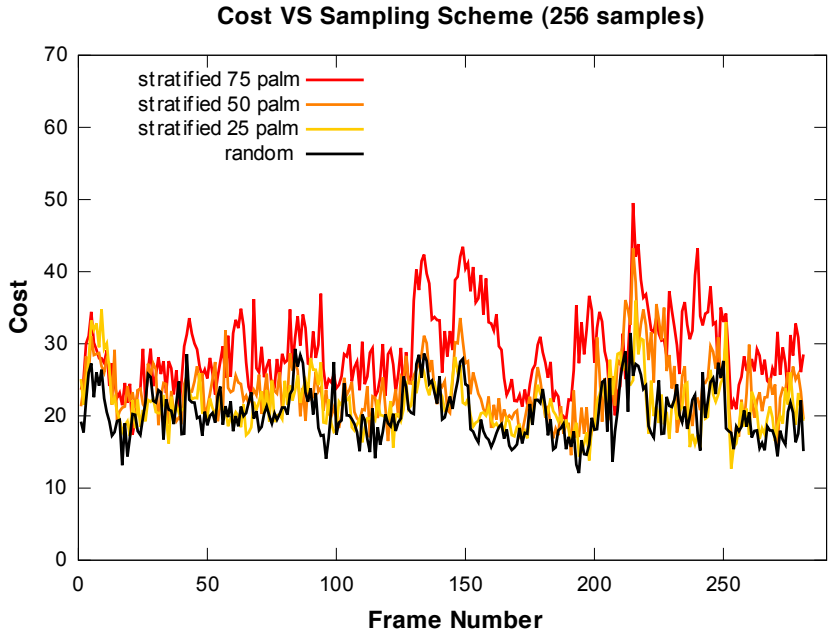

Figure 5.49: Cost variation under different stratified sampling schemes.

has been chosen since prior experiments have shown that the effects of the sampling scheme are not as apparent for a large sample size. As shown in Figures 5.48 and 5.49, the results indicate that sampling more points on the palm makes the tracking worse. The Uniform Random, Stratified-25-palm and Random-25-palm schemes perform the best. Incidentally the percentage of sample points chosen on the palm under the Uniform Random sampling scheme is on average 35%. There is little difference between the Random and the Stratified sampling scheme, suggesting that the system can tolerate one or two iterations where a segment has not been sampled.

Perhaps the most surprising result is that there is very little difference in tracking performance between uniform random sampling and edge-weighted sampling (see Figure 5.50). One likely explanation is that there are already enough edge points in uniform random sampling such that having more edge points does not substantially improve the pose estimate. Note that a uniform random sampling scheme on a 3D hand surface will naturally tend to have more points at the edges in the 2D image space due to the foreshortening effect in the 3D to 2D projection of sample points.

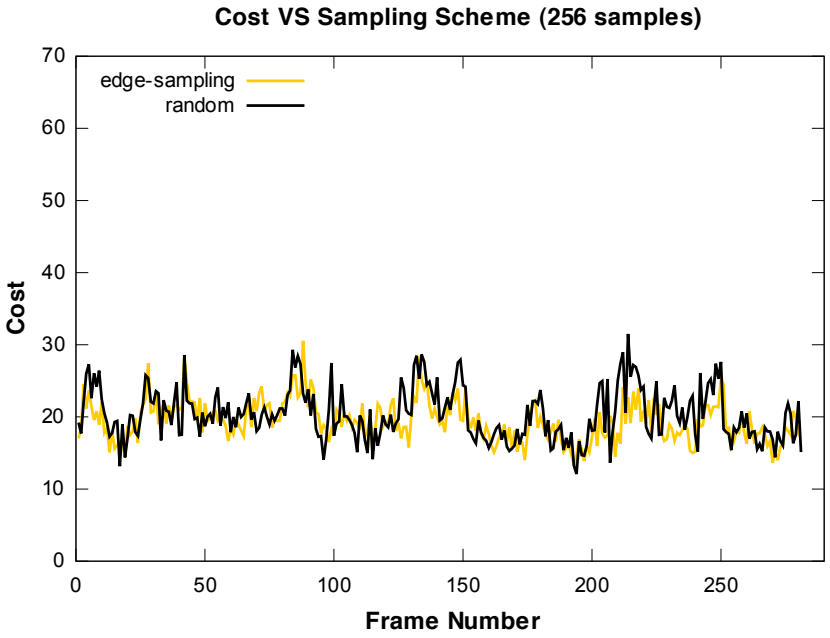

Figure 5.50: Cost variation for the edge sampling scheme.

Another interesting aspect in sampling is the number of samples required to get reasonable tracking performance. Sample sizes of 1024, 512, 256 and 128

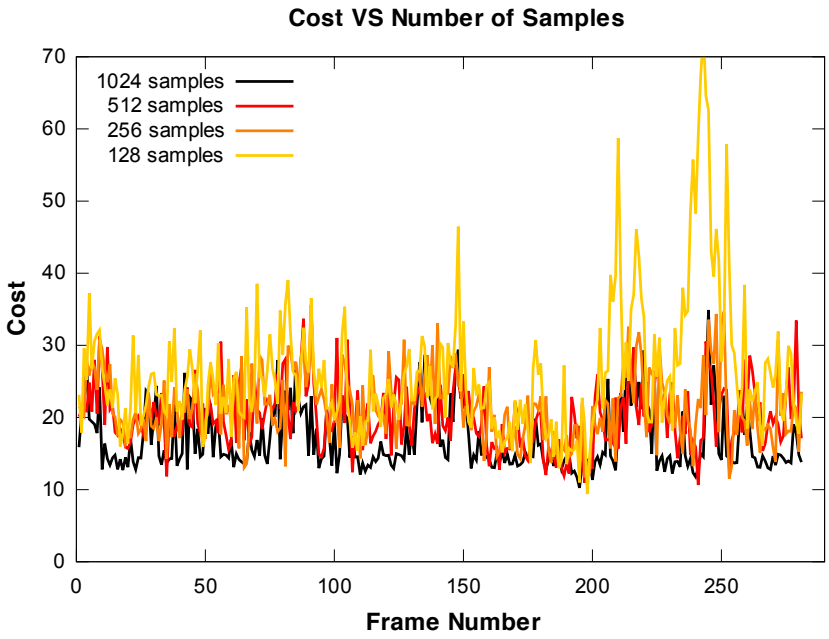

Figure 5.51: Cost variation under different sample sizes under Stratified 50 Palm sampling scheme.

(under the stratified-50-palm sampling scheme) are tested on the same finger-flexion sequence. The results can been seen in Figure 5.51. They indicate that under the same finite number of iterations, using more sample points gives a more accurate result. One would say that a sample size of 256 or 512 represents a good compromise between accuracy and computational load.

5.4 Summary

There is large variation in the achieved tracking performance. Tracking estimates are substantially more accurate in situations where the pose structure is strongly encapsulated by the silhouette. The open palm pose is an example of such a pose. Difficulty arises when the digits are close to one another, as well as when self-occlusion occurs between adjacent digits and also between the palm and the digit. Theoretically the photo-consistency cost function should resolve much of these difficult poses where the silhouette cue is less informative. This does not appear to be case in practice. The results empirically indicate that the photo-consistency cost function has a very small basin of attraction. Coupled with the presence of numerous local minima in the photo-consistency cost function, one would conclude that the photo-consistency cost function is only effective for localised fine-tuning of the pose. This is not enough to faithfully track a hand where fast and large finger movements are a common occurrence.

So far the temporal information of hand motion has not been exploited by the tracker. The articulated movement of the hand follows Newtonian mechanics. One can devise a motion predictor to initialise the tracker to a more favorable starting point based on the past motion history. Hopefully this new starting point puts the tracking system within the basin of attraction of the unique global minimum. This is explored in Chapter 6.

The importance of choosing the right sampling scheme has also been explored in this chapter. The results indicate that the effects of changing the sampling scheme for a sufficiently large sample size (more than 500 sample points) are minor. For smaller sample sizes, it appears that uniform random sampling is one of the better schemes.

This chapter has also explored the performance of the tracking system under various optimisation algorithms. It is clear that SMD is the algorithm of choice for hand tracking in this stochastic setting. Its computational time per iteration is low compared to GN, LM, oBFGS and comparable to the fastest optimisation algorithm, SG. More importantly however, SMD requires less iterations to converge to an acceptable solution than the other algorithms. GN and LM in particular appear ill-suited for this optimisation problem. This is understandable since both GN and LM are designed for optimisation over a deterministic cost function. Both algorithms do not have provisions for handling stochasticity in the cost function. Based on these results, we will only focus on the tracking performance using SMD in the subsequent chapters.

Chapter 6

Motion Prediction

Motion prediction is an important aspect in tracking and often improves tracking performance. For particle filter based trackers, motion prediction plays a significant role in a smart redistribution of particles for the next frame, given the past motion history [33]. Analogously, motion prediction can be used to find a good starting point for the optimisation routine in the next frame for gradient based trackers. As shown in Chapter 4 the tracker's convergence property only holds locally at the unique global minimum. Therefore it is desirable for the starting point at each frame to lie within the basin of attraction.

A variety of motion models have been proposed, ranging from anatomically correct dynamic models [85] to learned motion models derived from offline training on motion capture data [93, 73, 33, 3, 40, 77]. Learned motion models are popular and are generally described in a lower dimensional space [33, 93, 3, 73, 40, 77]. The rationale is that typical human motion lies in a subspace or submanifold of the high dimensional angle-parameter space. In [3], PCA is used to reduce dimensionality, and an auto-regressive motion model is trained on this reduced subspace. For applications where the set of motions being tracked is restricted, such a framework has been shown to be robust. However, a motion model learned this way does not generalise well to motions not observed in the training data. This is even more so if dimensionality reduction is used, since it is entirely possible that the new motion does not lie in the subspace the motion model is trained for. For a motion model learned offline to generalise well, one needs to ensure that the motion data used for training is not biased. This often means training over a large dataset that is rich enough, which can be impractical at times.

In this chapter we introduce an *online* adaptive vector-autoregressive (VAR) model for motion prediction. This is an attractive alternative in the sense that it

is efficient to evaluate and does not require offline training. The VAR prediction model is highly adaptive via trust factors and, as an online algorithm, potentially generalises better to different hand movements. The predicted pose estimate generated by the VAR prediction model is used as an improved starting point for the optimisation algorithm in our hand tracking system. Note that the proposed VAR prediction model is independent of the tracking system used and can for example be applied to a particle filter based system.

Ideally one ought to reformulate the prediction problem in an adaptive filtering framework where pose estimates made by the tracker's optimisation routine are taken as the filter input measurements. However implementing such a filter in a proper manner would require one to know about the non-linear dynamics of the hand which in itself is a difficult open problem [83, 85, 88]. Instead we opt for the easier alternative of a data-driven approach to motion prediction via VAR models.

The formulation of the VAR model is described in Section 6.1. Section 6.2 details the quantitative tracking results for a synthetic sequence when the VAR prediction model is applied. The results are compared against the no-motion and the decelerating motion prediction model. In Section 6.3, we introduce a robust adaptive version of the VAR motion prediction model. Section 6.4 explores the experimental results for the real video sequences. A comparison of the robust RVAR prediction model will also be made with a trained motion predictor in Section 6.5. Note that parts of this chapter have been published [22].

6.1 Motion Prediction Using a VAR Model

Both theoretical and experimental results from the previous chapters have suggested that tracking performance depends heavily on the initial value used at each frame for the optimisation routine. Ideally, the starting set of model parameters $\hat{\mathbf{x}}_k$ for the kth video frame should be initialised as close as possible to the optimal value $\mathbf{x}_k^*$, such that the tracker starts within the basin of attraction*.

In a data-driven approach, we treat the evolution of $\mathbf{x}_k^*$, $(\mathbf{x}_1^*, ..., \mathbf{x}_K^*)$, over K video frames as a 26-dimensional multiple time series. Given this multiple time series $(\mathbf{x}_1^*, ..., \mathbf{x}_K^*)$ at time K, a VAR (Vector Autoregressive) model can be used to predict a suitable initial value $\hat{\mathbf{x}}_{K+1}$ for the next frame $K+1$.

*The subscript k in $\mathbf{x}_k$ denoting $\mathbf{x}$ at the kth frame is used to avoid confusion with $\mathbf{x}_t$ which denotes $\mathbf{x}$ after the tth iteration in the optimisation routine.

6.1.1 VAR Model Formulation

Consider a multivariate weakly stationary process Y_k and its realisation, a 26-dimensional multiple time series $(y_1, ..., y_K)$, up to time K. The weakly stationary property of Y_k requires the expectation $E(Y_k)$ and autocovariance $E[(Y_k - \mu_Y)(Y_k - \mu_Y)]$ to be time invariant. Let U_k be a 26-dimensional vector of white noise. The VAR(p) model [50] of this process is given as

$$Y_k = A_1 Y_{k-1} + ... + A_p Y_{k-p} + U_k, \tag{6.1}$$

where p denotes the order of the VAR model and $A_1, ..., A_p$ are $\mathbb{R}^{26\times 26}$ parameter matrices.

Let X_k^* be the process that generates $(\mathbf{x}_1^*, ..., \mathbf{x}_K^*)$. X_k^* is not weakly stationary. If it were, it would imply that the hand pose in the video sequence is more or less stationary for all times k which is false. Hence one cannot directly apply a VAR model of X_k^* for motion prediction. Instead, differencing [14] is used to generate a weakly stationary process, Y_k^*, from X_k^*. Let Y_k^* be the process that generates model parameter 'accelerations' derived from X_k^*, *i.e.*,

$$Y_k^* = X_k^* - 2X_{k-1}^* + X_{k-2}^*. \tag{6.2}$$

We assume Y_k^* to be a weakly stationary process. This is a reasonable assumption since a statistical analysis of Y_k^* for the video sequences reveals that the model parameter accelerations consistently fluctuate around zero, with $E(Y_k^*)$ being time invariant. The autocovariance of Y_k^* is also at least locally constant.

Thus, we model Y_k^* as a VAR process and use the VAR model to make a one-step ahead prediction $\hat{y}_{K+1}$ for frame $K+1$ of the model parameter accelerations. This is done using the realisation-equivalent form of Equation (6.1), *i.e.*

$$\hat{y}_{K+1} = A_1 y_K^* + ... + A_p y_{K-p}^* + u_K. \tag{6.3}$$

In practice, u_K is taken to be the expectation $E(U_k) = 0$, since it is usually unknown. $\hat{y}_{K+1}$ is then mapped back to $\hat{\mathbf{x}}_{K+1}$ via

$$\hat{\mathbf{x}}_{K+1} = \hat{y}_{K+1} + 2\mathbf{x}_K^* - \mathbf{x}_{K-1}^*. \tag{6.4}$$

In general the tracking system does not know $(\mathbf{x}_1^*, ..., \mathbf{x}_K^*)$ as they are the ground truth values of the hand model parameters. The best that one can do is to use the corresponding tracker estimates $(\mathbf{x}_1, ..., \mathbf{x}_K)$ in place of $(\mathbf{x}_1^*, ..., \mathbf{x}_K^*)$ for motion

prediction. Hence a more accurate model would be

$$Y_k^* = A_1 Y_{k-1}^* + ... + A_p Y_{k-p}^* + U_k \tag{6.5}$$
$$Y_k = Y_k^* + CV_k, \tag{6.6}$$

which can be rewritten as

$$Y_k^* = A_1 Y_{k-1} + ... + A_p Y_{k-p} + \tag{6.7}$$
$$(U_k - (A_1 CV_{k-1} + ... + A_p CV_{k-p})), \tag{6.8}$$

where V_k is assumed to be a noise vector and C a noise-mixing matrix. However this ARMA-type model is difficult to solve as the statistics of V_k are unknown. We opt instead to use the VAR(p) model described in (6.1) for motion prediction. This is equivalent to treating the $U_k - (A_1 CV_{k-1} + ... + A_p CV_{k-p})$ term in (6.8) as white noise.

Doing this will obviously produce a less accurate prediction $\hat{y}_{K+1}$ and the corresponding prediction $\hat{\mathbf{x}}_{K+1}$ for the next frame $K+1$. But it is important to reiterate again that we are *not* trying to find $\mathbf{x}_{K+1}^*$ using a VAR motion prediction model. Our prediction $\hat{\mathbf{x}}_{K+1}$ merely provides a better initial value for the stochastic optimisation routine, which is set to find $\mathbf{x}_{K+1}^*$. It is sufficient for our prediction $\hat{\mathbf{x}}_{K+1}$ to be close enough to $\mathbf{x}_{K+1}^*$ such that starting the optimisation routine at $\hat{\mathbf{x}}_{K+1}$ for frame $K+1$ is better than starting at $\mathbf{x}_K$, *i.e.* without motion prediction. In fact, as we will see later in the experimental results, using this simplified model is already enough to improve tracking performance.

6.1.2 Estimation of VAR Parameters

Given the past observations $(y_1, ..., y_K)$ up to time K, the VAR parameters $A_1, ..., A_p$ are estimated via least squares estimation [50]. Note that the least squares estimator has the additional interpretation of being a maximum likelihood estimator if one assumes U_k to be gaussian. Let

$$\mathbf{Y} := (y_1, ..., y_K) \tag{6.9}$$
$$\mathbf{B} := (A_1, ..., A_p) \tag{6.10}$$
$$W_k := \begin{pmatrix} y_k \\ \vdots \\ y_{k-p+1} \end{pmatrix} \tag{6.11}$$
$$\mathbf{W} := (W_0, ..., W_{K-1}) \tag{6.12}$$

Then the least squares estimate, $\hat{\mathbf{B}}$, of $\mathbf{B}$ is given as

$$\hat{\mathbf{B}} = \mathbf{YW}'(\mathbf{WW}')^{-\mathbf{1}}. \tag{6.13}$$

In our case, we solve for $\hat{\mathbf{B}}$ online. That is, $\hat{\mathbf{B}}$ is re-evaluated with each new y_k obtained as the tracking system completes another frame in the video sequence. Therefore our prediction $\hat{y}_{K+1}$ of the accelerations of the hand model parameters for the next frame $K+1$ is

$$\hat{y}_{K+1} = \hat{\mathbf{B}}_K W_K. \tag{6.14}$$

An issue that is problematic in practice is that the computational cost for $\hat{\mathbf{B}}_K$ increases as new y_k's are obtained. To address this, we adopt a limited memory version of the least squares estimator by setting $Y := (y_{K-N}, ..., y_K)$ for a reasonably sized N. In other words, we drop the oldest sample y_{K-N} upon receiving a new sample y_{K+1}. Testing on video sequences shows that having a limited memory actually improves prediction results. This is because a limited memory implementation caters better for instances of volatility clustering where the autocovariance is locally constant, but can be observed to vary over longer time periods. Using a limited memory least squares estimator essentially means we are only using a local portion of the time series, which is more likely to have a constant autocovariance, thereby more likely to satisfy the weak stationarity assumption for a VAR model.

6.1.3 Performance Evaluation

A measure $\tilde{R}^2$ is introduced to quantify the improvement in tracking peformance when a VAR motion prediction model has been added to the tracking system. $\tilde{R}^2$ is loosely based on the R^2 measure used in regression analysis [6].

Let $y_k^*(i)$ be the ground truth time series for the ith hand model parameter in the acceleration domain. Also, let $y_k^0(i)$ be the times series produced by the tracking system without motion prediction, *i.e.* where the motion prediction for the next time step $k+1$ is

$$\hat{\mathbf{x}}_{k+1}^0(i) = \mathbf{x}_k^0(i), \tag{6.15}$$

and thus the corresponding $\hat{y}_{k+1}^0$ is

$$\hat{y}_{k+1}^0(i) = -\mathbf{x}_k^0 + \mathbf{x}_{k-1}^0. \tag{6.16}$$

Similarly, let $y_k^{\mathrm{VAR}}(i)$ be the corresponding time series estimated by the tracking system with motion prediction via a VAR model. Given a video sequence of length τ in total, the $\tilde{R}_y^2(\tau, i)$ measure for the ith hand model parameter in the acceleration domain is defined as

$$\tilde{R}_y^2(\tau, i) := 1 - \frac{\sum_{k=1}^{\tau}(y_k^*(i) - y_k^{\mathrm{VAR}}(i))^2}{\sum_{k=1}^{\tau}(y_k^*(i) - y_k^0(i))^2}, \tag{6.17}$$

evaluated over the entire sequence, from $k = 1, ..., \tau$. $\tilde{R}_y^2(\tau, i)$ rates the improvement in tracking accuracy by comparing the squared sum of residuals between $y_k^{\mathrm{VAR}}(i)$ and the ground truth against the residuals between the no-prediction time series $y_k^0(i)$ and the ground truth.

Measuring these residual errors in the acceleration domain of the hand model parameters is less indicative of tracking performance than measuring residual errors of the hand's joint and fingertip positions in 3D space. Hence, let G be the function that maps the hand model parameters $\mathbf{x}_k$ to the hand joint/fingertip positions of the hand z_k. Specifically,

$$G : \mathbb{R}^{26} \rightarrow \mathbb{R}^{3\times 21} \tag{6.18}$$

$$z_k = G(\mathbf{x}_k). \tag{6.19}$$

The columns of z_k are the joint/fingertip positions in 3D space (there are 16 joints and 5 fingertips in total). Let $z_k(j)$ be the 3D position of the jth joint/fingertip at frame k. Then $\tilde{R}^2(\tau, j)$ is defined as

$$\tilde{R}^2(\tau, j) := 1 - \frac{\sum_{k=1}^{\tau} ||z_k^*(j) - z_k^{\mathrm{VAR}}(j)||^2}{\sum_{k=1}^{\tau} ||z_k^*(j) - z_k^0(j)||^2}. \tag{6.20}$$

As a guideline, $\tilde{R}^2(\tau, j) > 0$ indicates that the tracking estimate with the help of motion prediction is more accurate than that without motion prediction, for the jth joint/fingertip. The converse is true if $\tilde{R}^2(\tau, j) < 0$.

6.2 Experiments on a Synthetic Sequence

The varying types of VAR models (to be discussed in the following sections) are initially tested on a synthetic sequence generated from real hand movements. The synthetic sequence provides ground truth values for a quantitative analysis of tracking performance. It contains elements of typical hand movements such as palm rotations and finger articulation.

Each experiment is repeated over 50 trials. As mentioned previously, the VAR parameters $\hat{\mathbf{B}}_\mathbf{k}$ are evaluated online for each frame k; no offline training is involved. The tracker is manually initialised to a good starting position for the first frame of the video sequence. About 600 sample points are used to track the moving hand. SMD (stochastic meta-descent) [68, 13] is the optimisation algorithm of choice, given its superior performance in the previous experiments in Chapter 5.

Additional computation time for an online VAR-based motion prediction evaluation is comparatively negligible, on average taking 14 milliseconds per frame.

The results of two control experiments are used as benchmarks. The first control experiment is tracking without motion prediction. We denote the multiple time series of the resulting joint/fingertip positions from this experiment as z_k^0. The other control experiment uses the popular decelerating motion prediction model [7, 42]; let $z_k^{\mathrm{D}} = G(\mathbf{x}_k^{\mathrm{D}})$ be the corresponding multiple time series. The predictor $\hat{\mathbf{x}}_{k+1}^{\mathrm{D}}$ is defined as

$$\hat{\mathbf{x}}_{k+1}^{\mathrm{D}} = \mathbf{x}_k^{\mathrm{D}} + \rho(\mathbf{x}_k^{\mathrm{D}} - \mathbf{x}_{k-1}^{\mathrm{D}}), \tag{6.21}$$

where $\rho \in [0, 1]$. Thus the corresponding $\hat{y}_{k+1}^{\mathrm{D}}$ is

$$\hat{y}_{k+1}^{\mathrm{D}} = (\rho - 1)(\mathbf{x}_k^{\mathrm{D}} - \mathbf{x}_{k-1}^{\mathrm{D}}). \tag{6.22}$$

Note that $\rho = 1$ gives the constant velocity model. However, tracking with the constant velocity model is found to be unstable. Testing on a range of ρ values show that setting $\rho = 0.4$ produces the best tracking performance. This will be used in our comparison with the VAR-based prediction models.

Figure 6.5 on page 120 shows the overall mean error, taken to be the average of all joint/fingertip errors over all 50 trials for both the standalone tracker and the tracker with the deceleration predictor. The overall mean error is formulated as follows: Let $p_{i,g}$ and a_i be the predicted and actual 3D positions of the ith joint/fingertip for the gth trial. Then the overall mean error is defined as

$$E_{\mathrm{mean}} := \frac{1}{GI} \sum_{g}^{G} \sum_{i}^{I} ||a_i - p_{i,g}||, \tag{6.23}$$

where I is the total number of joints/fingertips in the hand model and G is the total number of trials.

There are three noticeable error peaks in the video sequence. The first peak (frame 73) represents the tracking inaccuracy due to a misfit of the little finger

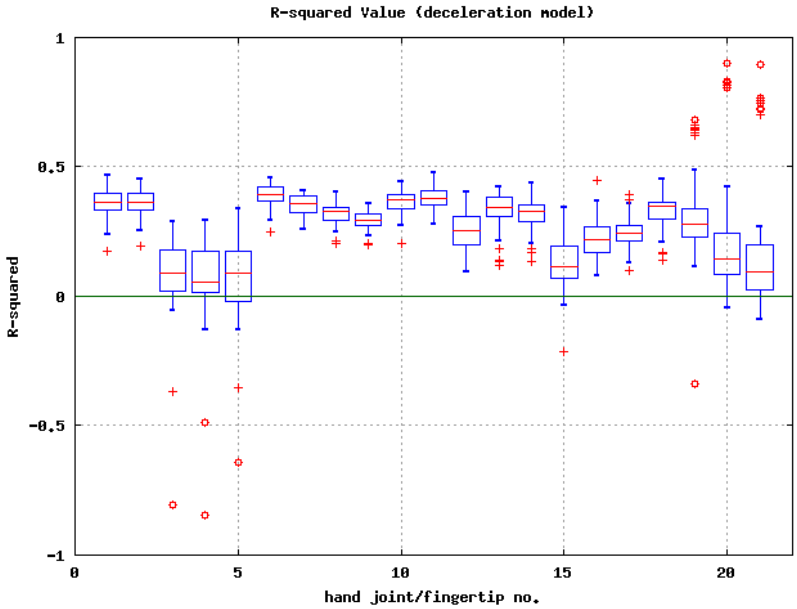

Figure 6.1: $\tilde{R}^2(\tau)$ of z_k^{D} for each hand joint/fingertip over the 50 trials. Outliers that lie between 1-3 times the interquartile range are marked as '+' while those that lie over 3 times the interquartile range are marked as 'o'.

(see Figure 6.2). The second peak (frame 100) corresponds to the part where the hand undergoes a global rotation. The third peak (frame 150) occurs as the thumb moves across the palm, partially occluding the fingers and the palm.

One can already see the improvement with the deceleration prediction model, especially between frames 100 to 130. This improvement is reflected in the $\tilde{R}^2$ value of the entire sequence for z_k^{D} in Figure 6.1. Excluding several outliers for the thumb, there is a noted improvement over tracking without motion prediction.

The following subsections describe our results for the different variations of the VAR (vector autoregressive) model tested, namely the traditional full-VAR model and a structured-VAR model where the kinematic relations of the hand are accounted for.

6.2.1 VAR Model Order Selection

VAR model order selection based on the AIC or BIC measure is a standard procedure [50]. This is meaningful when the training time series $\mathbf{x}_k$ is fixed while the various VAR models are fitted during the criterion evaluation process. A complication in our situation is that the time series is dependent on the VAR

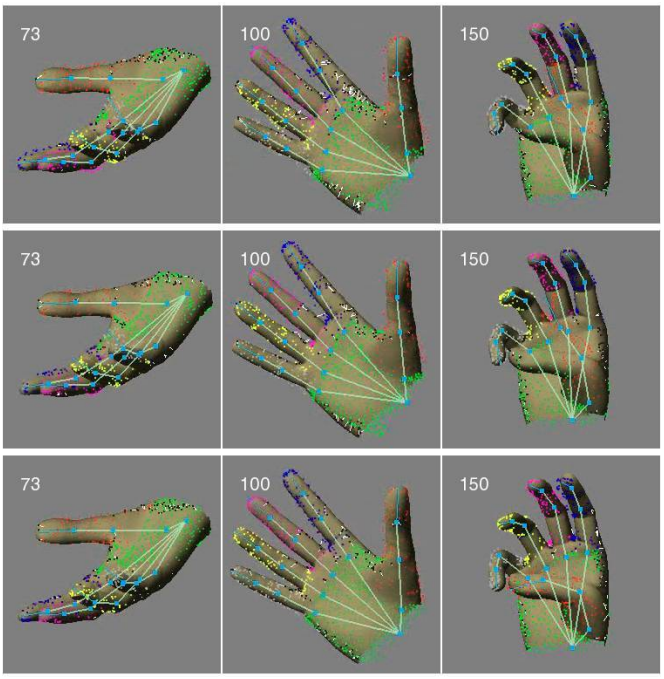

Figure 6.2: Selected frames where the overall mean error peaks. The top row shows tracking results with no motion prediction, the middle row corresponds to tracking with the deceleration predictor, and the last row corresponds to tracking with RVAR motion prediction.

model used. Different VAR models produce different predictions of the initial value for the finite-length optimisation routine, which leads to different evolutions of $\mathbf{x}_k$.

At best, the AIC and BIC criteria can only give a rough initial value for order selection. Initial AIC and BIC tests evaluated on the static ground truth sequence tend to favour VAR models of higher orders (> 5). However, the tracking performance of the tracker with these motion predictors of higher order is substantially worse. Only the results of the first order variants of the VAR model are shown; the results of the higher order models are omitted as they are consistently worse for each case.

6.2.2 Full-VAR model

Recall that the VAR(1) predictor $\hat{y}_{K+1}$ for the hand model parameters in the acceleration domain is given as

$$\hat{y}_{K+1} = \hat{A}_1 y_K, \tag{6.24}$$

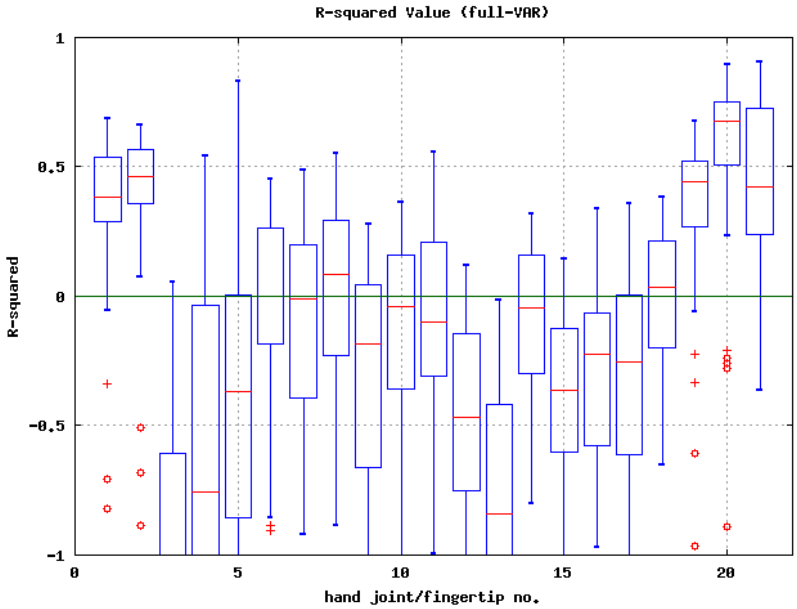

Figure 6.3: $\tilde{R}^2(\tau)$ of z_k^{FV} for each hand joint/fingertip over the 50 trials.

where $\hat{A}_1$ is obtained directly by solving (6.13). We denote the multiple time series generated by the tracking system using this motion predictor in the hand joint/fingertip domain as z_k^{FV}. Initial tests reveal that all the VAR-based prediction models tend to give large error deviations when the change in acceleration is abrupt, leading to gross tracking inaccuracies. To mitigate this, a hard threshold has been set such that the difference between $\hat{\mathbf{x}}_{K+1}$ and $\mathbf{x}_K$ cannot be more than 10 degrees.

The tracking result z_k^{FV} is in general worse than z_k^0, as shown by the corresponding $\tilde{R}^2$ values in Figure 6.3. To understand why this is so, one should note that $\hat{A}_1$ quantifies the correlations not the causations (*i.e.* tendon forces controlling the hand) in the joint movements. The limited-memory sample set (the memory size $N = 150$) used to refine $\hat{A}_1$ online is simply not rich enough and represents a biased sample of all possible hand movements. This leads to inaccurate predictions. Additional prior information, in the form of constraints, is needed for solving $\hat{A}_1$ sensibly.

6.2.3 Structured-VAR model

In this situation, $\hat{A}_1$ is solved under the constraints induced by the hand's kinematic structure. We observe that the movements of joints along the kinematic

chain of each digit are correlated, and that their dependency on each other allows for the flexion of each digit. We then assume that the movement of each digit is independent of other digits. In addition we assume that the rotation and translation movements of the palm are independent of each other and are also independent of the movement of each digit.[†] Applying these priors constrains $\hat{A}_1$ to a block diagonal form

$$\hat{A}_1 = \begin{pmatrix} \mathbf{G_{6\times 6}} & 0 & 0 & \cdots & 0 \\ 0 & \mathbf{R^{Y}_{5\times 5}} & 0 & \cdots & 0 \\ 0 & 0 & \mathbf{R^{1}_{3\times 3}} & 0 & \vdots \\ \vdots & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \mathbf{R^{5}_{3\times 3}} \end{pmatrix}, \tag{6.25}$$

where $\mathbf{G_{6\times 6}}$ is a diagonal matrix that relates to the translation and rotation parameters of the palm. $\mathbf{R^{Y}_{5\times 5}}$ is a diagonal matrix that relates to the rotation parameter of each digit that models the abduction/adduction of each digit. Lastly, $\mathbf{R^{n}_{3x3}}$ relates to the rotation parameters of the nth digit responsible for the flexion of the digit.

The multiple time series in the hand joint/fingertip domain generated by the tracking system using this motion predictor shall be denoted as z_k^{SV}. A limited memory size of $N = 30$ has been used for the structured-VAR model.

Enforcing kinematic constraints on $\hat{A}_i$ results in better tracking performance than when the full-VAR model is used. The first two error peaks that have been observed in both control experiments are dampened when the structured-VAR motion predictor is applied (see Figure 6.5 on page 120). The dampened errors are reflected in the $\tilde{R}^2$ values for z_k^{SV} (see Figure 6.4), where the tracking accuracy of the palm joint (joint/finger no. 1) and the joints on the little finger (joint/fingertip no. 18-21) has drastically improved. Errors for the middle finger have worsened, in particular the fingertip (no. 13), resulting in the enlarged third peak in Figure 6.5.

Compared to the deceleration predictor, adding the structured-VAR predictor to the tracker increases the variance of the tracking performance. This is generally undesirable. Then again, having a larger variance does mean the tracker has a better chance of escaping from local minima. The extended upper tails of the $\hat{R}^2$

[†]These assumptions are rough at best, *e.g.*, it is well known that the flexions of the little finger and the ring finger are not completely independent. [85] would be a more realistic (and sophisticated) model. For our application however, we find that our simple model already works well.

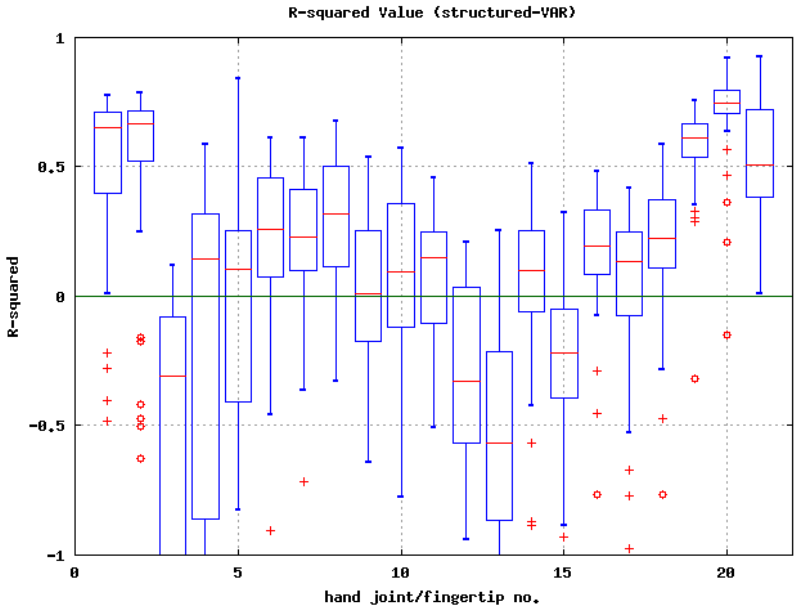

Figure 6.4: $\tilde{R}^2(\tau)$ of z_k^{SV} shows a large improvement for the base joint and the little finger.

distributions for the thumb's PIP joint (no. 4) and the thumb tip (no. 5) attest to this (see Figure 6.4).

6.3 Robust VAR

An ideal motion predictor should have the high performance of the structured-VAR model observed for certain joints, augmented with the general consistency and low variance of the deceleration predictor.

To achieve this, we combine the structured-VAR predictor and the deceleration predictor (with $\rho = 1$), tempered with adaptive trust factors. We shall refer to this adaptive scheme as the RVAR (Robust Vector Autoregressive) prediction model.

The RVAR model interpolates online a weighted combination of $\hat{\mathbf{x}}_{K+1}^{\mathrm{SV}}$, $\hat{\mathbf{x}}_{K+1}^{\mathrm{D}}$ and $\hat{\mathbf{x}}_{K+1}^{0}$ predictions. The interpolation is based on the reliability of the past predictions for each predictor when compared against the past history $\mathbf{x}_{K-N}...\mathbf{x}_K$ of tracking results. The RVAR predictor for the ith hand model parameter $\hat{\mathbf{x}}_{K+1}(i)$ is defined as

$$\hat{\mathbf{x}}_{K+1}^{\mathrm{RVAR}}(i) = (1-\gamma)\hat{\mathbf{x}}_{K+1}^{0}(i) + \gamma(\alpha_i \hat{\mathbf{x}}_{K+1}^{\mathrm{SV}}(i) + \beta_i \hat{\mathbf{x}}_{K+1}^{\mathrm{D}}(i)), \tag{6.26}$$

where $\gamma \in [0,1]$, $\alpha_i + \beta_i \leq 1$, and $\alpha_i, \beta_i \geq 0$.[‡] Using motion prediction to find a good initial value for the optimisation routine can be tricky in that there is always an inherent danger of overshooting. Hence we introduce γ, an upper bound on how much one should trust the motion predictions. γ is fixed throughout the tracking sequence. α_i and β_i are adaptive weights that minimise the following sum for the respective value of i.

$$E_{K,i}(\alpha,\beta) = \sum_{k=K-c}^{K} h_k[\mathbf{x}_k(i) - (\alpha\hat{\mathbf{x}}_k^{SV}(i) + \beta\hat{\mathbf{x}}_k^{\mathrm{D}}(i))]^2, \tag{6.27}$$

where c is a cut-off constant and h_k is a decaying factor defined as

$$h_k = \frac{1}{2^{K-k}}. \tag{6.28}$$

Minimising $E_{K,i}$ gives the optimal α_i and β_i for $\mathbf{x}_k(i)$ at frames $k \in [K-c, K]$. These weights are then used in (6.26) when the prediction $\hat{\mathbf{x}}_{K+1}^{\mathrm{RVAR}}(i)$ for the next unknown frame $K+1$ is made. $E_{K,i}$ is minimised separately for each hand model parameter i to obtain separate sets of α_i and β_i weights.

The absolute interpolation of the predicted model parameters in (6.26) and (6.27) is just one of many interpolation approaches. For example, one can instead choose to interpolate the changes in predicted model parameters relative to the position in the last frame *i.e.* $\hat{\mathbf{x}}_k^{\mathrm{SV}}(i) - \mathbf{x}_{k-1}(i)$. Note that the minimising objective function (6.27) and the update equation (6.26) in the relative interpolation approach will become

$$\begin{aligned}
\hat{\mathbf{x}}_{K+1}^{\mathrm{RVAR}}(i) &= \mathbf{x}_K(i) + \gamma[\alpha_i(\hat{\mathbf{x}}_{K+1}^{\mathrm{SV}}(i) - \mathbf{x}_K(i)) + \beta_i(\hat{\mathbf{x}}_{K+1}^{\mathrm{D}}(i) - \mathbf{x}_K(i))] && (6.29)\\
&= (1 - \gamma(\alpha_i + \beta_i))\hat{\mathbf{x}}_{K+1}^{0}(i) + \gamma(\alpha_i\hat{\mathbf{x}}_{K+1}^{\mathrm{SV}}(i) + \beta_i\hat{\mathbf{x}}_{K+1}^{\mathrm{D}}(i)) && (6.30)
\end{aligned}$$

and

$$\begin{aligned}
E_{K,i}(\alpha,\beta) &= \sum_{k=K-C}^{T} h_k[\mathbf{x}_k(i) - \mathbf{x}_{k-1}(i) && (6.31)\\
&- (\alpha(\hat{\mathbf{x}}_k^{SV}(i) - \mathbf{x}_{k-1}(i)) + \beta(\hat{\mathbf{x}}_k^{\mathrm{D}}(i) - \mathbf{x}_{k-1}(i)))]^2 && (6.32)
\end{aligned}$$

respectively.

Results for both RVAR predictors (with $\gamma = 0.8$, $c = 8$) show a marked improvement over both the deceleration and the structured-VAR predictor (see

[‡]Note again that the no-motion predictor $\hat{\mathbf{x}}_{K+1}^{0}(i) = x_K(i)$.

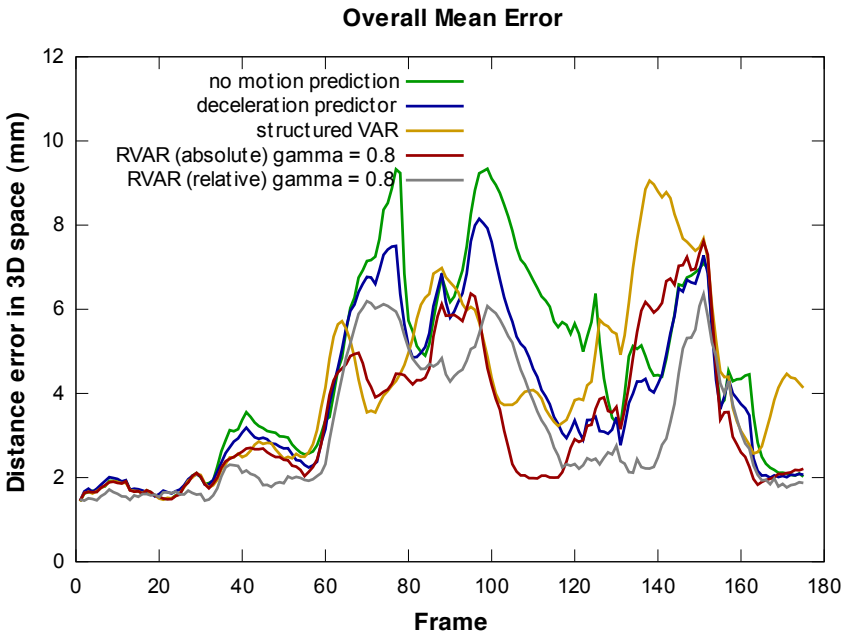

Figure 6.5: Overall mean error for the absolute and relative RVAR models (red and grey respectively) with $\gamma = 0.8$, and cut-off constant $c = 8$. Tracking performance without motion prediction (green), with the deceleration predictor (blue) and with a structured-VAR model (yellow) is shown for comparison.

Figure 6.5). From Figure 6.8 on page 122, one can appreciate that the RVAR predictor has inherited the desired low variance property of the deceleration predictor while retaining the high performance of the structured-VAR model. Figure 6.6 shows the typical α and β values used by the adaptive RVAR model in the video sequence.

The $\tilde{R}^2$ values for most joints/fingertips reside around 0.5. With the exception of the middle finger's PIP joint (no. 12) and possibly the thumb joints/tip (no. 3-5), the absolute RVAR model gives better performance over the deceleration model (see Figure 6.7). Although the variance of the thumb joints is comparatively larger, the median of $\tilde{R}^2$ for the absolute RVAR model is actually slightly higher. Tuning $\gamma = 0.7$ results in a variance (for the thumb joints) that is similar to the variance of the deceleration case, without noticeable change to the statistics of the other joints/fingertips. We prefer keeping the higher variance for the thumb (*i.e.* with $\gamma = 0.8$) since it gives the tracker a better chance to escape from local minima. Frame 150 in Figure 6.2 on page 115 illustrates where using a RVAR model sometimes allows the tracker to escape from a local minimum

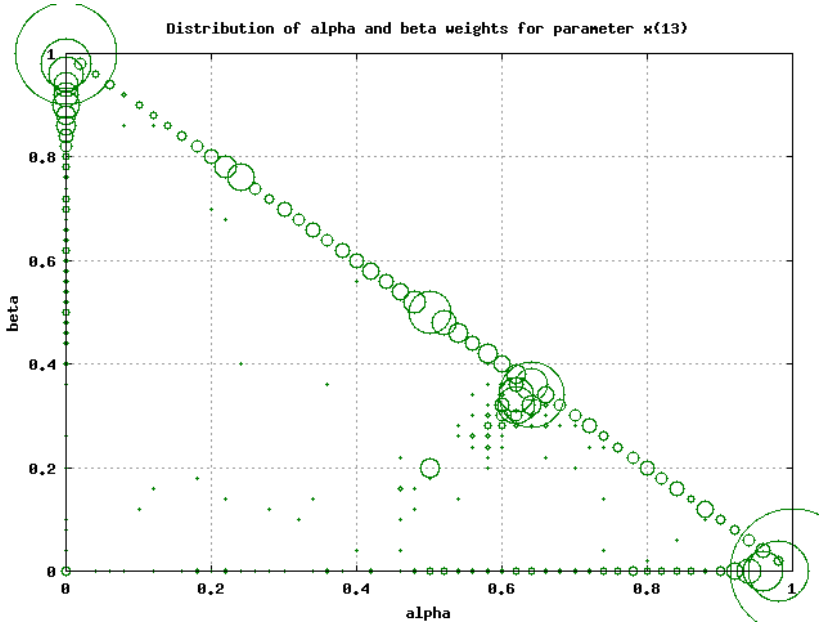

Figure 6.6: Distribution of α and β for the PIP joint of the index finger. The distribution of α and β for all other hand model parameters appear similar to this. The circle radius indicates the frequency of the values. The larger the circle, the higher the frequency.

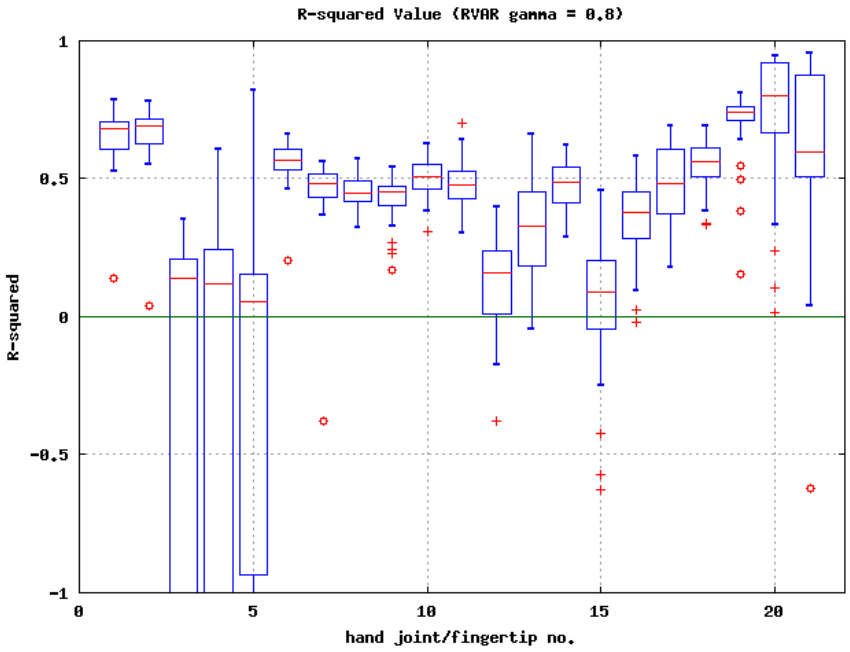

Figure 6.7: $\tilde{R}^2(\tau)$ for the RVAR (absolute) model

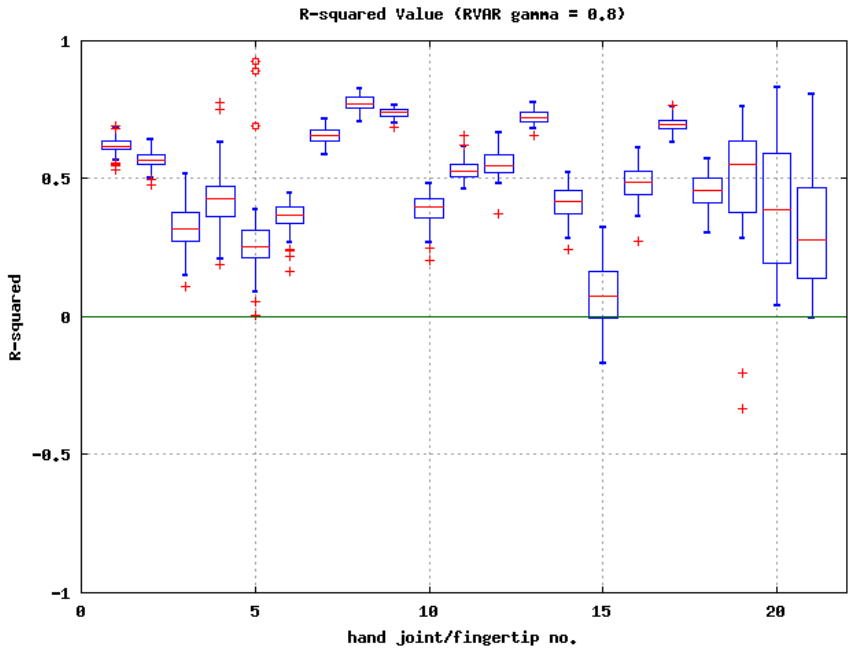

Figure 6.8: $\tilde{R}^2(\tau)$ for the RVAR (relative) model

whereas this is impossible with the deceleration model.

Similarly the relative RVAR model has shown to perform well for this synthetic sequence (see Figure 6.8). One could argue that the relative RVAR model is better than the absolute RVAR model given that the average $\tilde{R}^2$ values over all joints for both RVAR models are similar and that the $\tilde{R}^2$ values for the relative RVAR model have a much smaller variance than those of the absolute RVAR model. A smaller variance implies more consistent tracking results. We will see in the next section that the same relative behaviour of the absolute and relative RVAR models is observed for the real hand sequences.

6.4 Experiments on Real Sequences

The tracking performance of the tracking system using different motion prediction models is evaluated on the same real hand sequences used in Chapter 5. For the deceleration predictor, $\rho = 0.4$ as before, and $\gamma = 0.8, c = 8$ for both the absolute and relative motion RVAR motion predictors. Overall there is an improvement in tracking performance when motion prediction is used, with the relative RVAR motion predictor performing the best out of the deceleration and the RVAR motion predictors. The experimental results also demonstrate the flexibility of the online RVAR model in handling different hand motions. The following subsections examine each of the video sequences in detail.

6.4.1 Sequence 1 - Finger Flexion

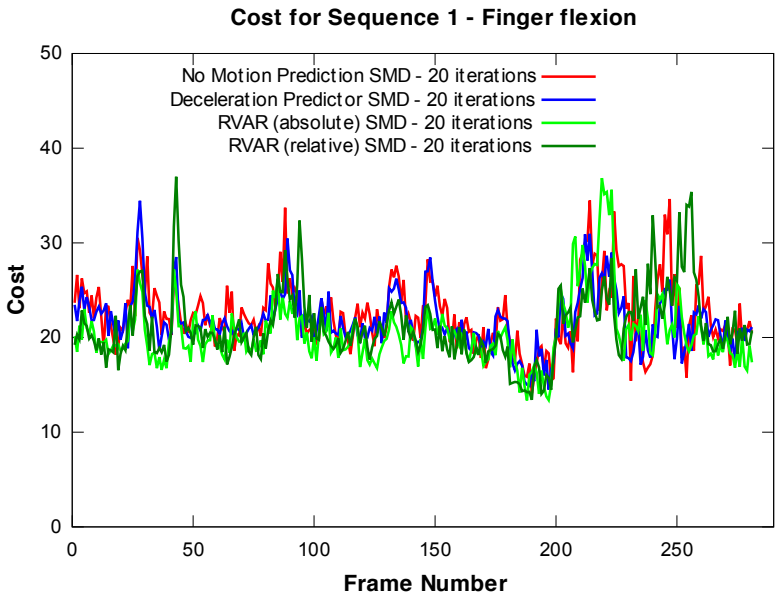

Figure 6.9: Cost per frame for finger flexion. For most frames, motion prediction with RVAR model (absolute and relative) gives better tracking results than no motion prediction or motion prediction with the deceleration predictor.

The improvement after applying motion prediction is not as apparent here since the tracker has already been tracking well previously without motion prediction for this particular sequence. Figure 6.9 shows that tracking with the deceleration predictor is slightly better than no-motion prediction. Tracking with

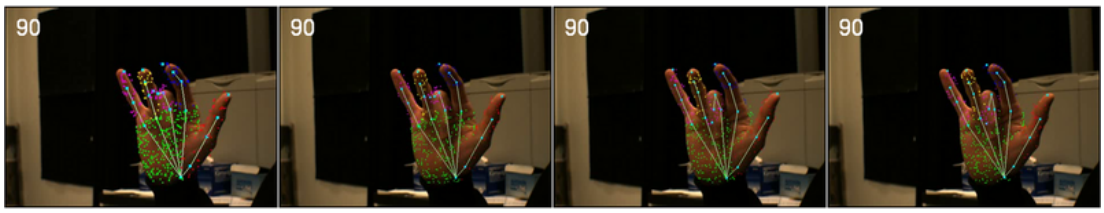

Figure 6.10: An example of the ability of the RVAR model to capture fast finger flexion. The rapid flexion of the middle finger fails to be captured by the tracker using the no-motion and the deceleration predictor (1st and 2nd image respectively). This is not the case when the absolute or relative RVAR model (3rd and 4th image respectively) is used.

the RVAR motion predictors is generally better than with either the deceleration or the no-motion predictors.

Although not immediately obvious from the cost graph in Figure 6.9, tracking with the RVAR predictors better captures fast finger flexions that are otherwise missed by the no-motion and the deceleration predictors. For example frame 90 in the sequence depicts a fast downward flexion of the middle finger (see Figure 6.10). Without the use of either of the RVAR predictors, the tracker loses track of the middle fingertip. This is not the case when using the RVAR predictors.

6.4.2 Sequence 2 - Dial Turning

There is an overall improvement in tracking over the no-motion predictor as shown in the cost graph in Figure 6.11. For example, one can compare the improved pose estimate for frame 56 in Figure 6.12 where the absolute RVAR predictor is used against that in Figure 5.22 on page 90 where there is no motion prediction. Notice also that the crossing of the index and middle fingers observed at frame 112 under the no-motion prediction scheme (see Figure 5.22, Chapter 5) is absent after applying the absolute RVAR prediction model. This crossing over is also absent when the relative RVAR motion predictor is used.

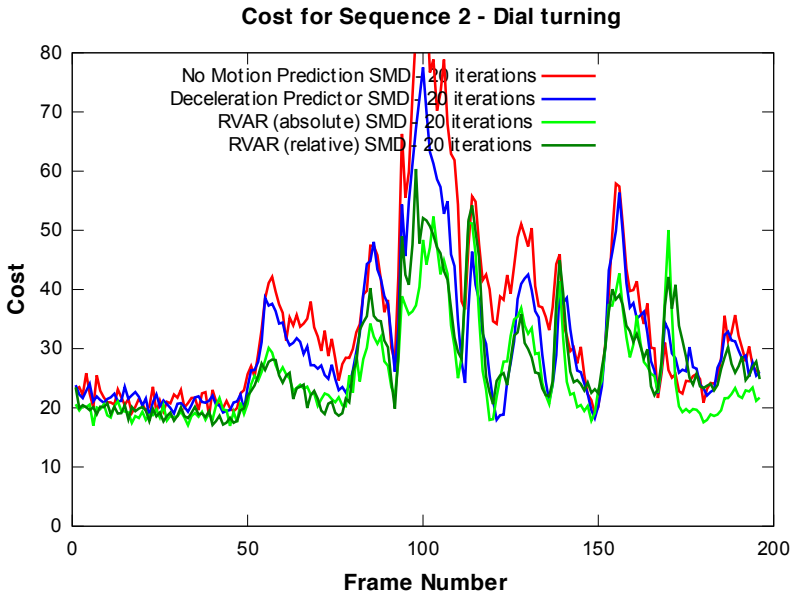

Figure 6.11: Cost per frame for the dial turning motion.

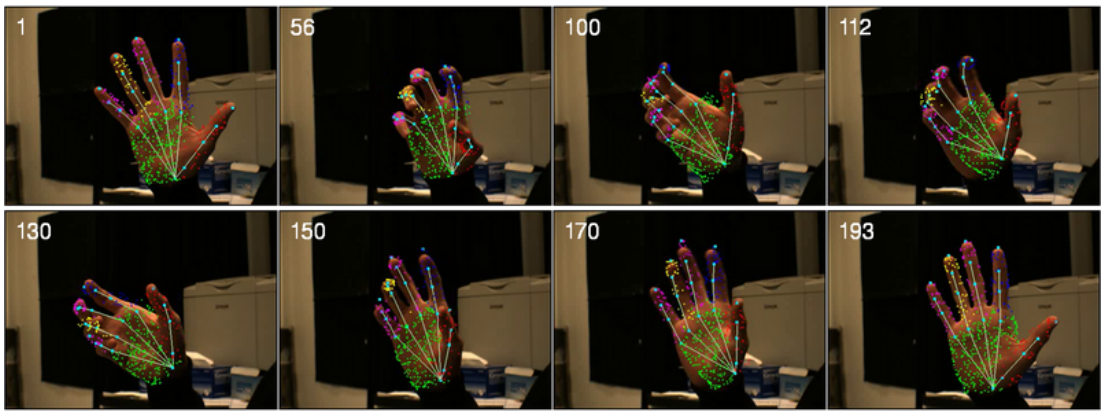

Figure 6.12: Selected frames showing the tracking results with the RVAR (absolute) prediction model applied.

6.4.3 Sequence 3 - Pinch

The addition of motion prediction substantially improves the tracking performance for this sequence. The issue of the index and middle fingers getting mixed up during the pinching motion in the inital experiments (refer back to Figure 5.14 on page 85, Chapter 5 for comparison) is absent when the deceleration or both of the RVAR predictors are used. Figure 6.14 shows the results for the tracker using the relative RVAR predictor. One can observe from frame 74 that the index finger is aligned properly and that the middle, ring and little fingers are now bending correctly compared to without motion prediction.

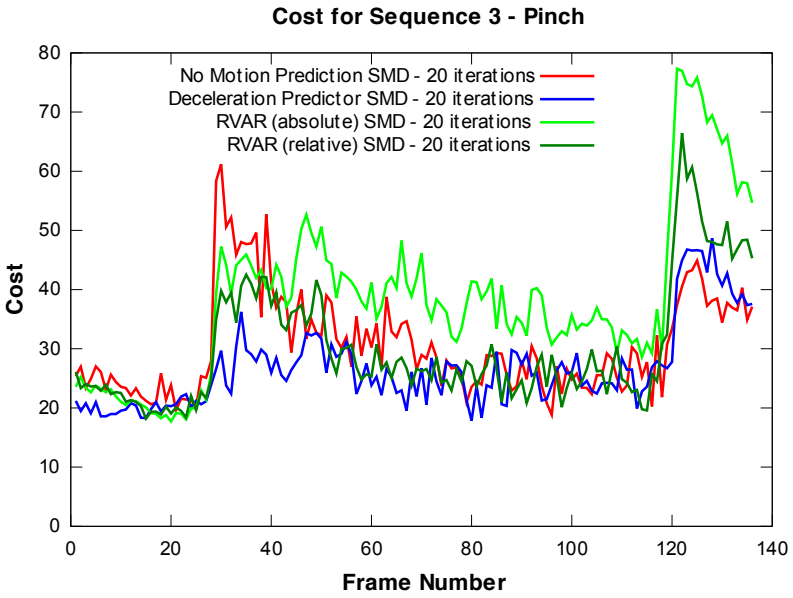

Figure 6.13: Cost per frame for the pinching motion.

The high cost spikes between frames 120-130 for the RVAR predictors and the deceleration motion predictor can be attributed to overshooting in the motion prediction. This leads to the tracker being stuck in a local minimum. Figure 6.15 is an example of the tracker getting stuck at a very bad local minimum. Note that this does not occur very often and can be mitigated by allowing more iterations to run (see Section 6.6).

6.4.4 Sequence 4 - Drag and Drop

Tracking results for the predictor are cleaner with motion prediction. Both the deceleration motion predictor and the relative motion predictor perform equally

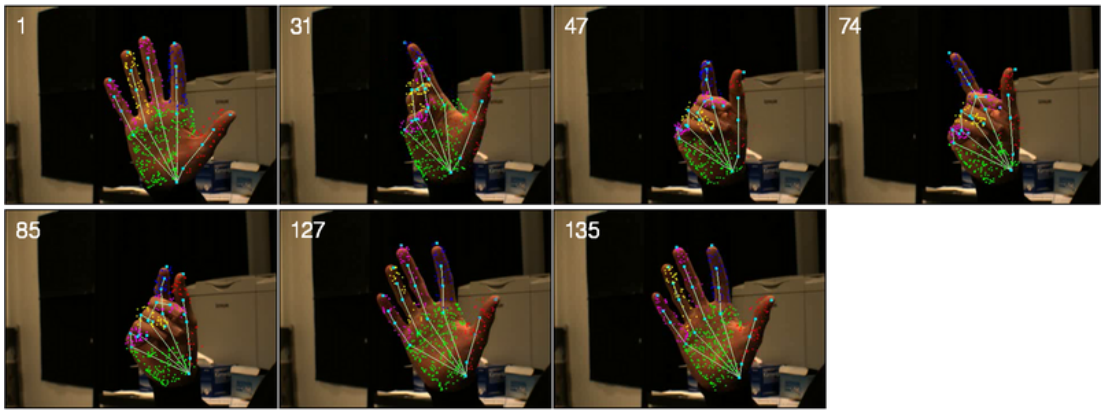

Figure 6.14: Selected frames for the pinching sequence for tracking with the RVAR (relative) prediction model.

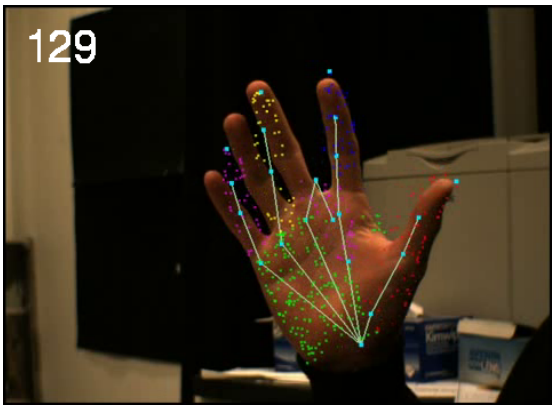

Figure 6.15: Poor pose estimate observed for certain trials for the deceleration motion predictor. This is sometimes observed for the RVAR predictors as well, hence the sharp peak at frame 129.

well (see Figures 6.16 and 6.17). Tracking with the absolute RVAR predictor however seems unstable starting near frame 60 (see Figure 6.18), although the tracking system does recover back to normal as the video sequence progresses. The poor pose estimate at frame 60 introduces noise in the motion prediction scheme for the absolute RVAR motion predictor. This noise results in a poor initial starting point for the next frame thereby causing the tracker to get temporarily stuck in a local minimum in the subsequent frames (*e.g.* frame 63).

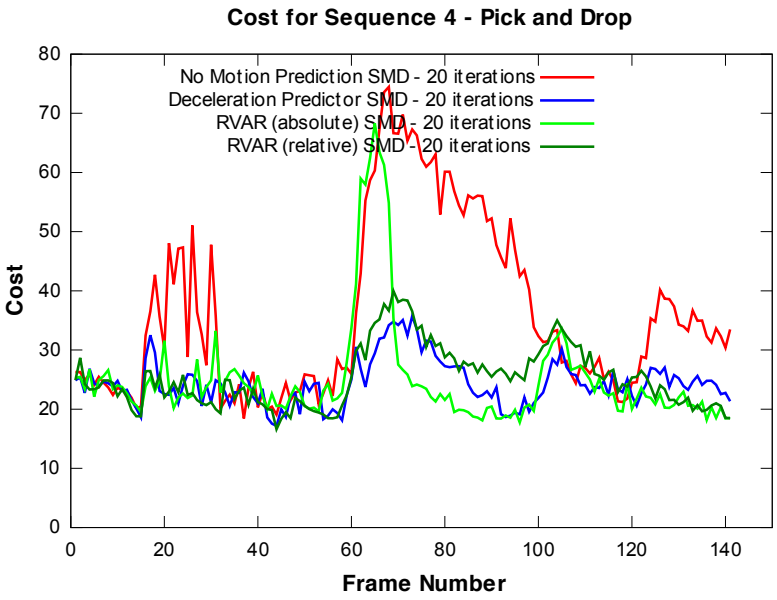

Figure 6.16: Cost per frame for the drag and drop motion.

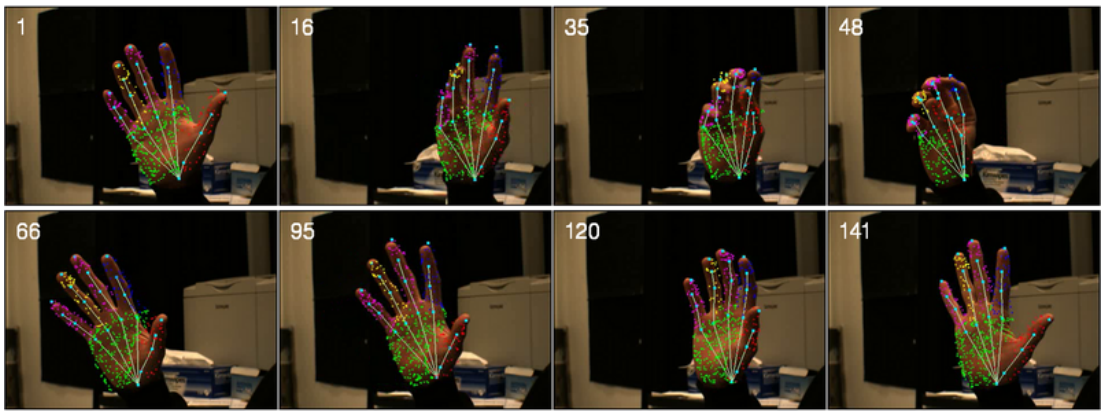

Figure 6.17: Selected frames for the drag and drop sequence for tracking with the RVAR (relative) prediction model.

6.4.5 Sequence 5 - Palm Rotation

The improvement in tracking under the use of the relative RVAR predictor (and to a lesser extent, the absolute RVAR predictor) is best characterised by this sequence. When the relative RVAR motion predictor is used, the tracker can consistently follow through the transitional flipping motion starting from frame 27 to 50 (see Figure 6.20). This is not observed for tracking with the deceleration motion predictor and is only observed in tracking without motion prediction

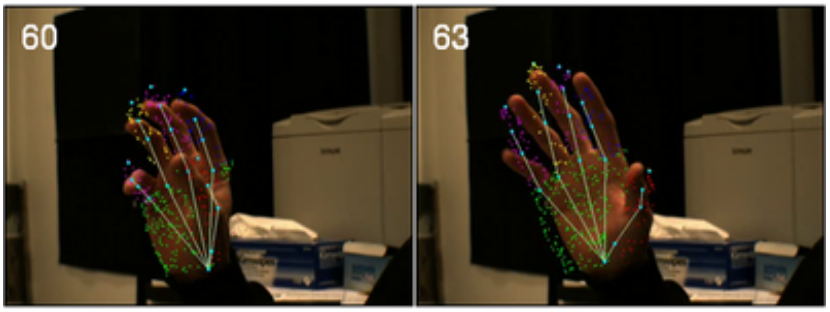

Figure 6.18: Inaccuracy at frames 60 to 63 made by the tracking system using the RVAR (absolute) predictor.

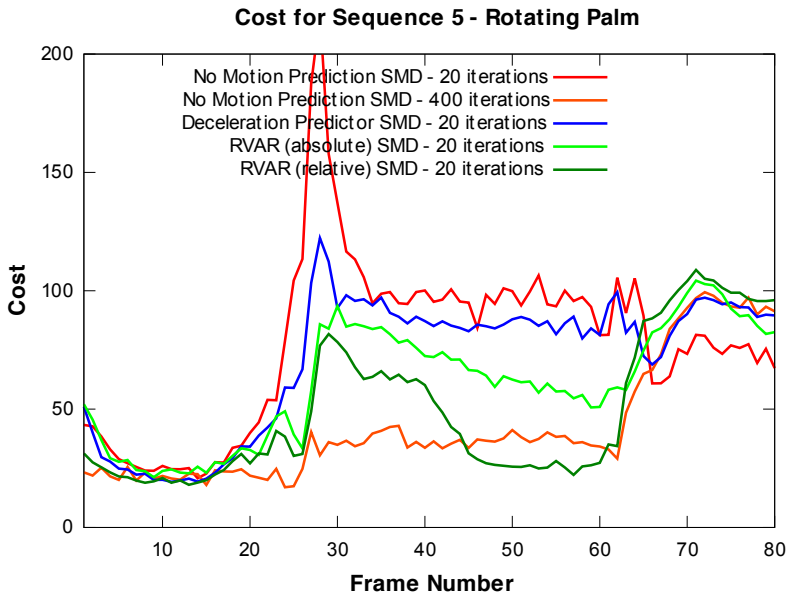

Figure 6.19: Cost per frame for the rotating palm sequence for tracking with the RVAR (relative) prediction model.

under a maximum of 400 allowable iterations (see Figure 5.47 on page 100). This demonstrates the advantage of the RVAR motion predictor, in that the tracking system can now match the accuracy of the 400-iteration sequence under a maximum of just 20 allowable iterations. The tracker with the absolute RVAR predictor is sometimes able to transition through the flipping motion, although without the level of consistency that the tracker with the relative RVAR predictor exhibits.

Despite this, even with the relative RVAR motion predictor, the tracker is

still unable to transition back from the flipping motion starting around frame 61. But by increasing the maximum allowable number of iterations to 400, the tracker with the RVAR (relative) motion predictor is able to transition through both flipping motions (see Figure 6.21).

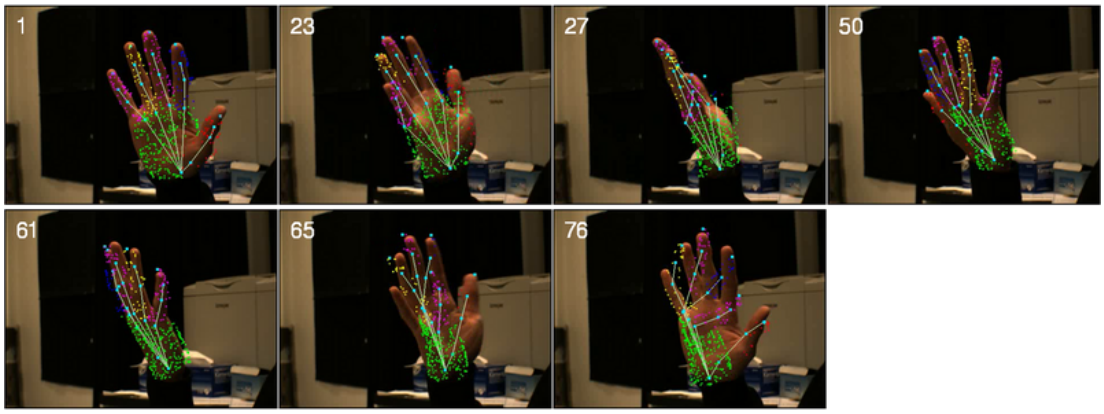

Figure 6.20: Selected frames for the palm rotate sequence for tracking with the RVAR (relative) prediction model.

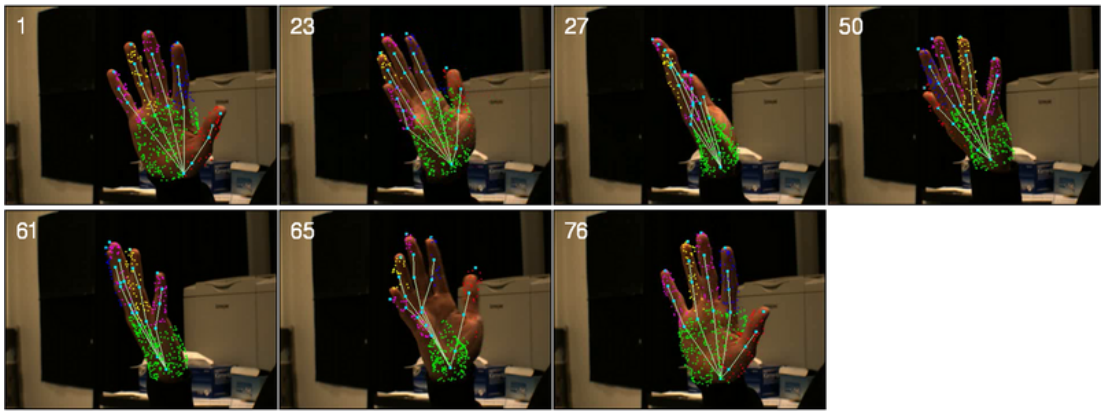

Figure 6.21: Selected frames for the palm rotate sequence for tracking with the RVAR (relative) prediction. When given a maximum allowable threshold of 400 iterations, the tracker is able to occasionally transition through both flipping motions in the video sequence.

6.5 Comparison with a Trained AR Predictor

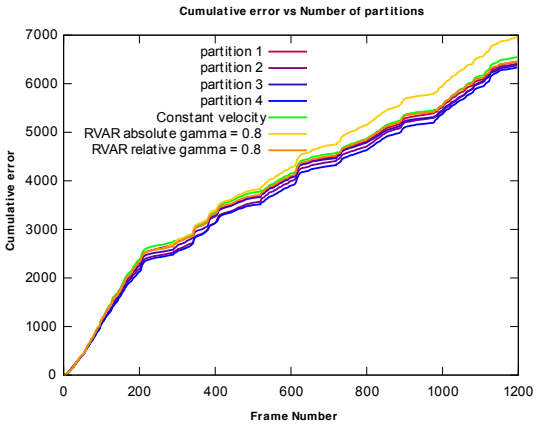

Figure 6.22: Cumulative error on the training set for the trained AR model of a variety of partition numbers.

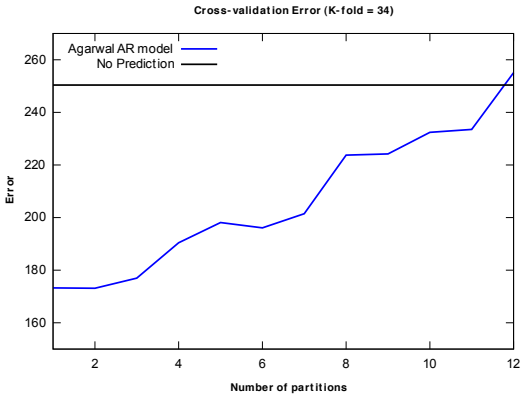

Figure 6.23: Cross-validation error (34-fold) for the trained AR models with different partitions.

The trained mixture of AR (auto-regressive) motion models described by Agarwal *et al* [2] has been implemented as a comparison against the RVAR (relative) motion predictor. For convenience we shall refer to the former as the Agarwal predictor. A brief overview of the Agarwal predictor is as follows: the space of a training set of motion capture data is partitioned into regions with similar dynamical characteristics. For each of these regions, dimensionality reduction via PCA is applied and an AR model is trained on this reduced space for the region. The prediction result made by the Agarwal predictor is the result of a Gaussian mixture model of these trained linear dynamical models.

We use motion capture data obtained from a data glove to train the Agarwal predictor. The motion capture data contains the same type of hand motions observed in the hand video sequences. For the best performance, one should train the motion model using motion data (most likely hand-labelled) obtained from a separate set of video sequences of the same type of hand motions from the same camera setup. Ultimately however, performance comparisons ought to be made in light of the practical limitations of deploying such a tracking system on a commercial production scale. It is our belief that training with video sequences obtained from hand-labelled data from the same set of cameras would be infeasible if the tracking system were a commercial product. Requiring the user to hand-label their own hand motion data for their unique camera setup prior to use is

unrealistic. From a practical standpoint, the better alternative would be for the manufacturer to use a large generic set of motion capture data from a data glove for training.

Figure 6.22 shows the training results of the Agarwal predictor. The cumulative prediction error depicted is obtained by running the Agarwal predictor of varying partition numbers on the training data set and comparing the parameter predictions against the actual parameter values. Prediction results on the training set are better than for the RVAR predictors, with the Agarwal predictor becoming more accurate as the number of partitions is increased. The optimal number of parameter space partitions (4 partitions) appears to be in agreement with the number of distinct hand motions in the training data. Agarwal predictors of a higher partition number are not shown as these results are worse than for the RVAR predictors. Figure 6.23 shows the 34-fold cross-validation error. One can observe that Agarwal predictors with a higher number of partitions do not generalise as well to unseen data as the Agarwal predictors with fewer partitions.

Initial experiments have indicated that the tracking performance with the Agarwal predictor is highly unstable due to the frequent overshooting effect in the predictions. Therefore, in the same spirit as the deceleration motion predictor, a trust factor $\rho = 0.5$ has been applied to dampen predictions made by the Agarwal predictor. Additionally a hard constraint has also been added to clip high angle changes in the predictions.

The following are the results of the tracker using Agarwal predictors of a varying number of partitions.

6.5.1 Sequence 1 - Finger Flexion

Based on the cost graph in Figure 6.24, the performance of the 5-partition Agarwal predictor is the best out of the Agarwal predictors. The performance is on-par with the RVAR (relative) predictor and at times more accurate (see cost at frame 255 in Figure 6.24). Having said that, an inspection in the screenshot reveals that there is still some difficulty in capturing large finger movements, *e.g.* frame 86 in Figure 6.25.

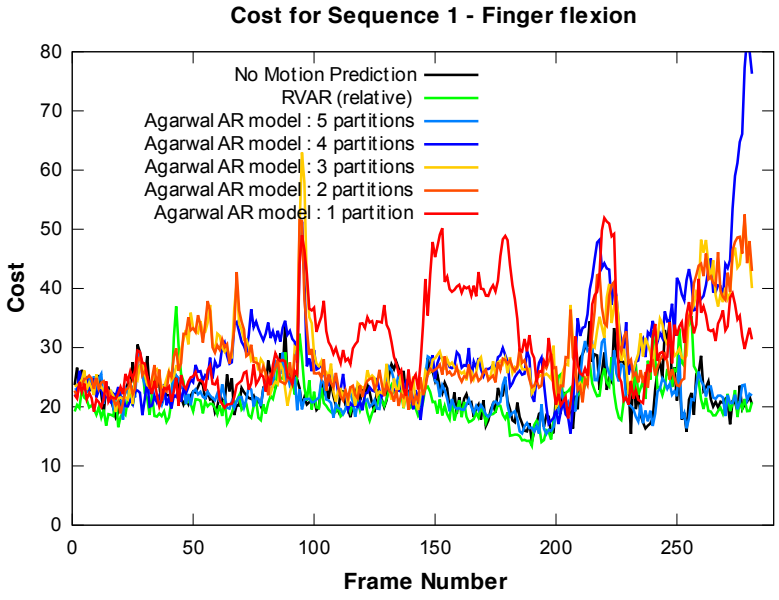

Figure 6.24: Cost per frame for flexion motion.

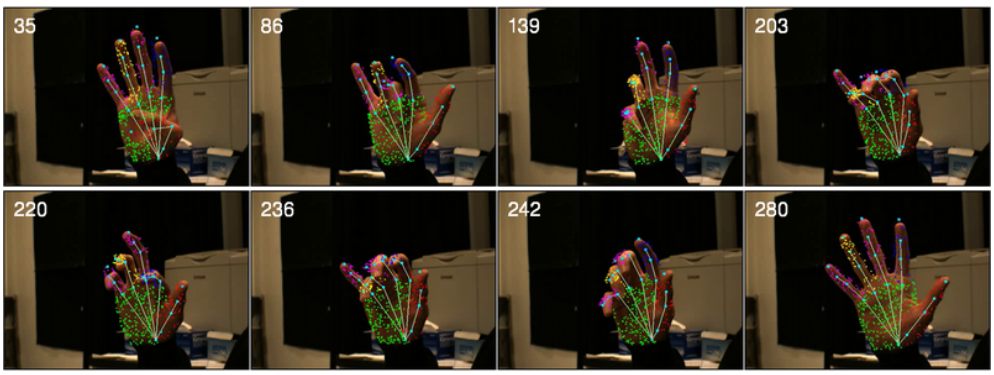

Figure 6.25: Selected frames for the flexion sequence for tracking with the trained AR model (Partition number = 5)

6.5.2 Sequence 2 - Dial Turning

All the Agarwal predictors have performed poorly for this particular sequence (see Figures 6.26 and 6.27). The tracking performance of the best Agarwal predictor (2-partitions) unfortunately is about the same as tracking without motion prediction.

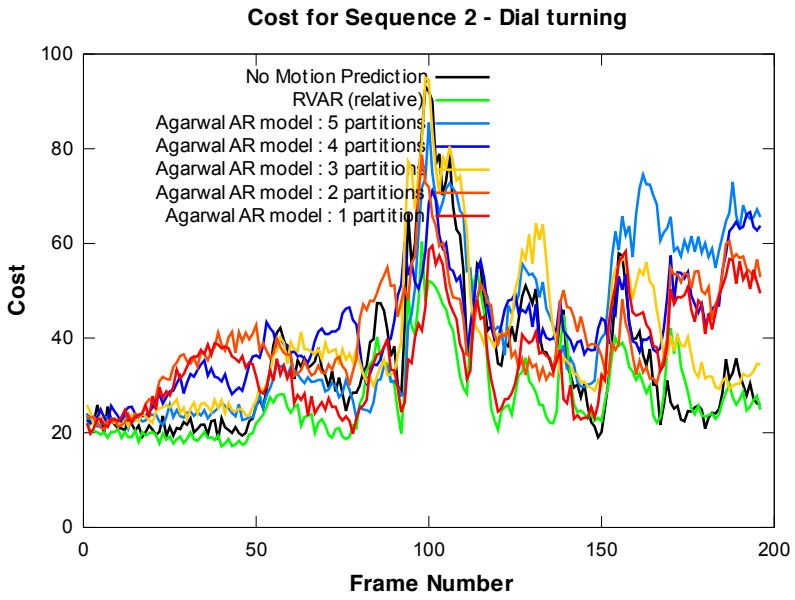

Figure 6.26: Cost per frame for dial turning motion.

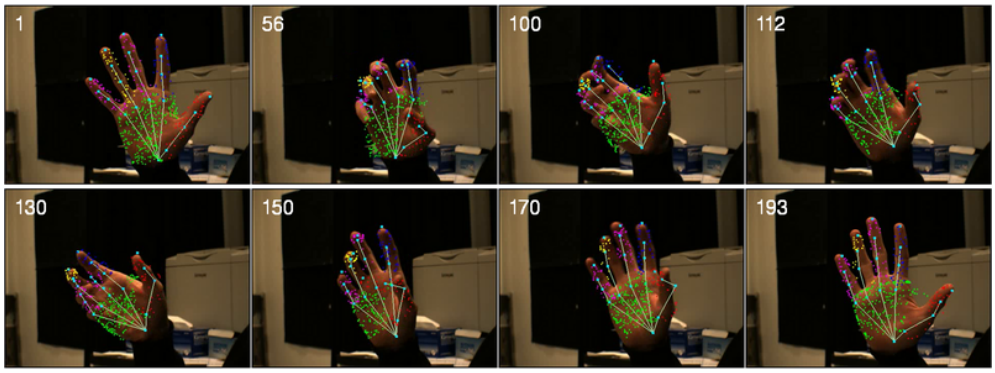

Figure 6.27: Selected frames for the dial turning sequence for tracking with the trained AR model (Partition number = 2)

6.5.3 Sequence 3 - Pinch

The cost graph in Figure 6.28 suggests that the 1-partition and 4-partition Agarwal predictors perform the best and in fact better than the RVAR (relative) predictor in the last portion of the video sequence. However, a closer inspection of the screenshots reveals that the middle and index fingers have been mixed up during the pinching motion (see Figure 6.29).

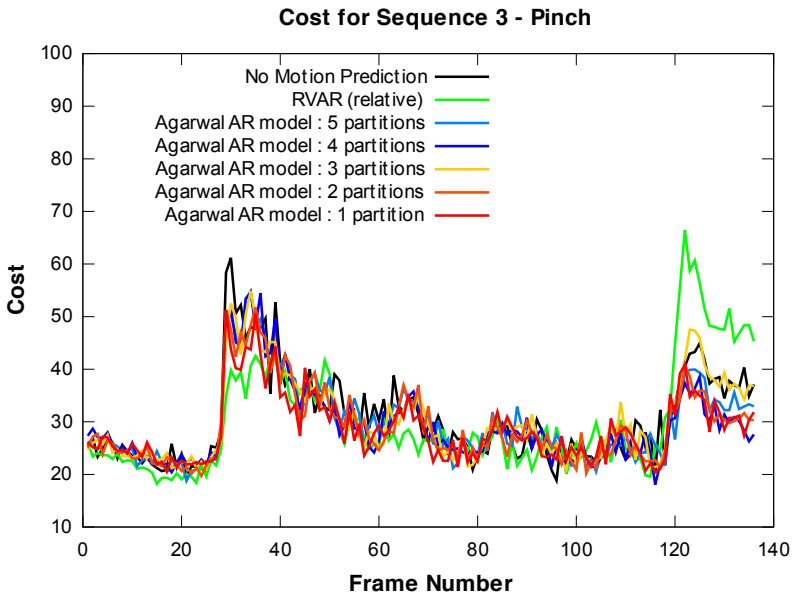

Figure 6.28: Cost per frame for pinch motion.

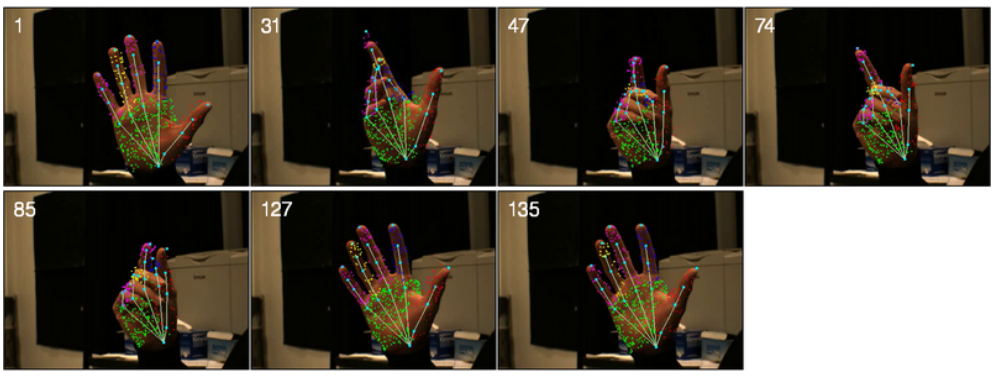

Figure 6.29: Selected frames for the pinch sequence for tracking with the trained AR model (Partition number = 4).

6.5.4 Sequence 4 - Pick and Drop

The 4-partition Agarwal predictor performs the best out of the Agarwal predictors see Figure 6.30. However it is comparatively worse than the RVAR (relative) predictor.

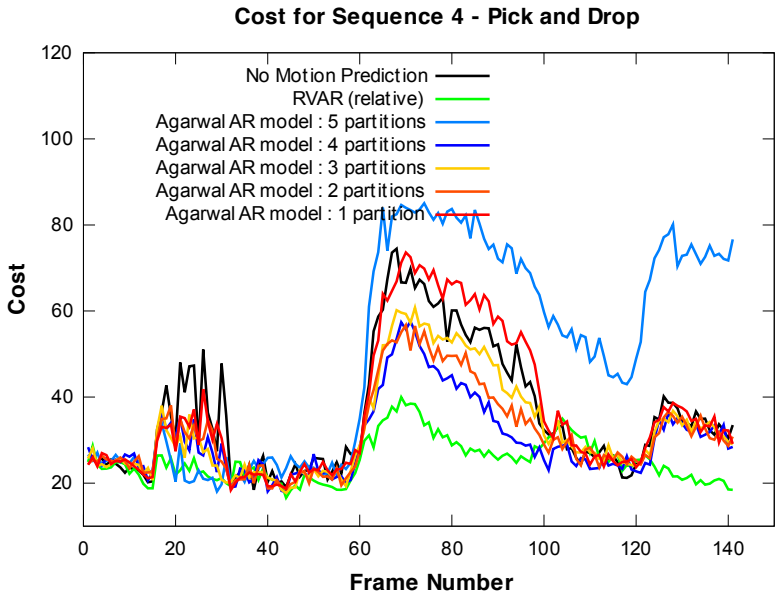

Figure 6.30: Cost per frame for pick and drop motion.

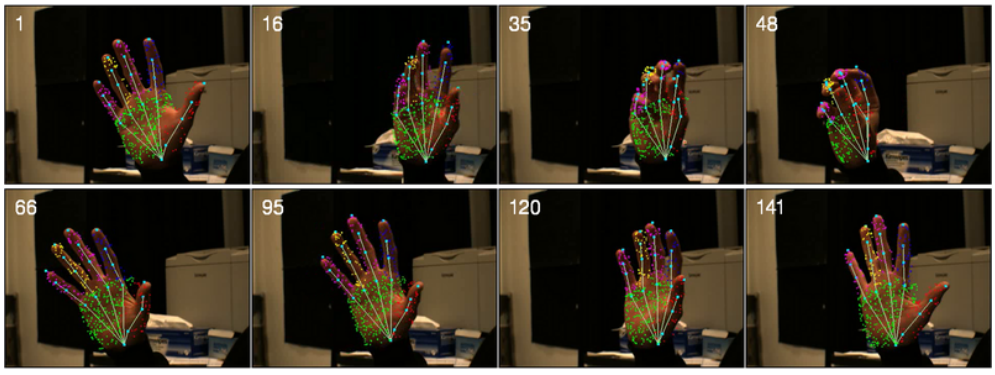

Figure 6.31: Selected frames for the pick and drop sequence for tracking with the trained AR model (Partition number = 4).

6.5.5 Sequence 5 - Palm Rotation

All the Agarwal predictors have performed badly in this difficult sequence (see Figures 6.32 and 6.33). Adding motion prediction helps very little in terms of tracking performance. None of the Agarwal predictors has helped the tracking system to transition through the first palm flipping motion.

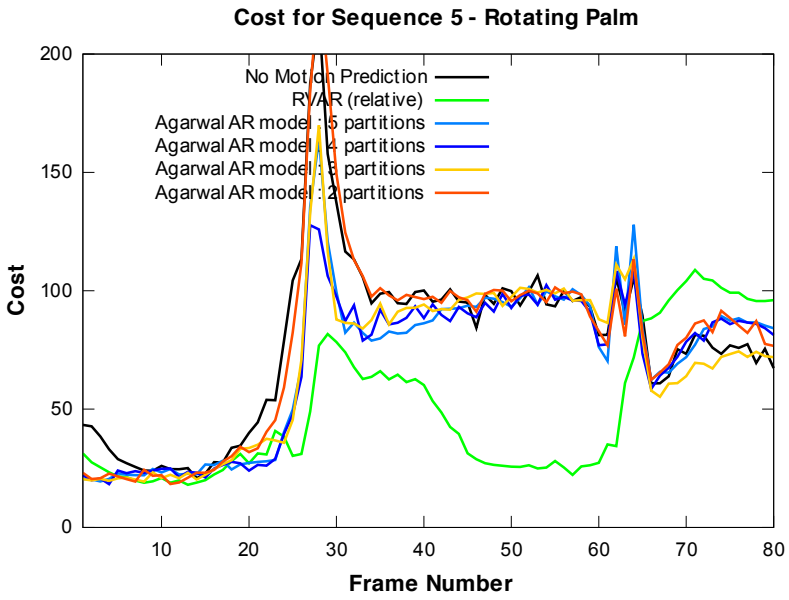

Figure 6.32: Cost per frame for palm rotation motion.

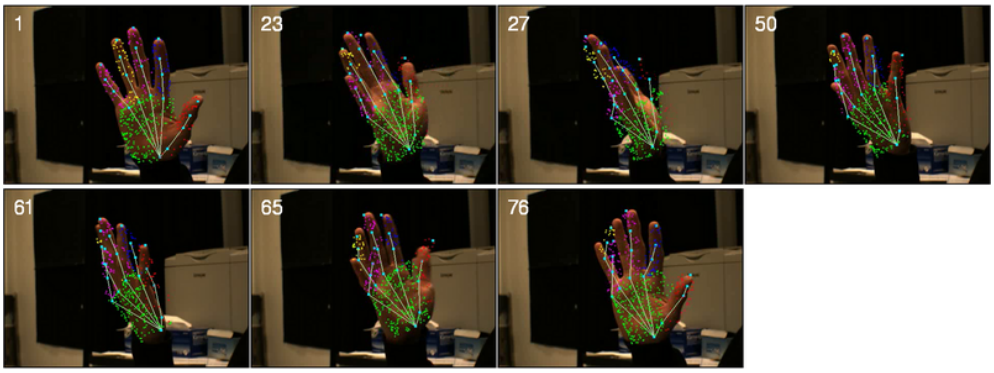

Figure 6.33: Selected frames for the palm rotation sequence for tracking with the trained AR model (Partition number = 4).

6.6 Tracking performance under 10000 iterations

Up to this point, the emphasis in this thesis has been focused towards a compromise between computation speed and tracking accuracy. Nevertheless it is interesting to examine the tracking performance near the limit, *i.e.* for the situation where computation time is not an issue and the tracker is allowed to iterate through a large number of iterations.

In what follows, we repeat the tracking experiments with the RVAR (relative) motion predictor except that the number of iterations for each frame has been increased to 10000. In general the tracking accuracy has visibly improved as shown in the cost graphs in Figures 6.34, 6.35, 6.36, 6.37, 6.38 and their respective screenshots in Figures 6.40, 6.41, 6.42, 6.43, 6.44. A decrease in tracking performance only occurs for a couple of frames in the flexion sequence near frame 260 and in the pinching sequence near frame 85 where the tip of the middle finger and the knuckle of the little finger are misaligned. However the tracker is able resolve this and return accurately back to the open palm pose at the end of the sequence.

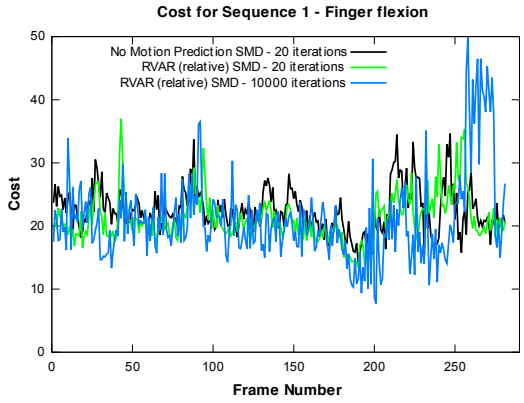

Figure 6.34: Cost per frame for the flexion motion. The blue curve indicates the tracker running with a maximum of 10000 allowable iterations.

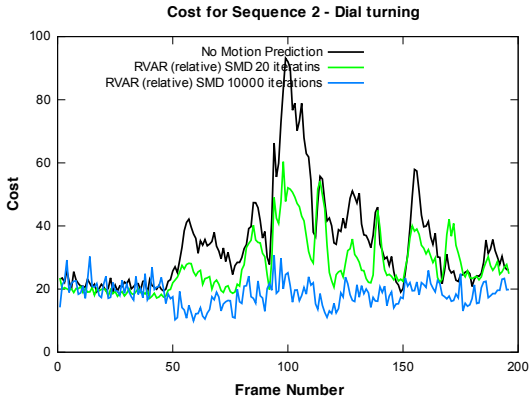

Figure 6.35: Cost per frame for the dial motion. The blue curve indicates the tracker running with a maximum of 10000 allowable iterations.

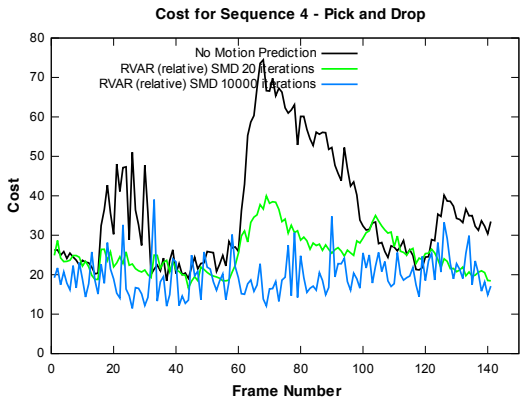

Figure 6.36: Cost per frame for the pick and drop motion.

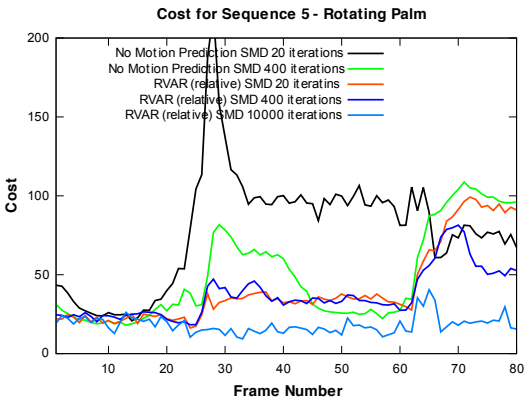

Figure 6.37: Cost per frame for the palm rotation motion.

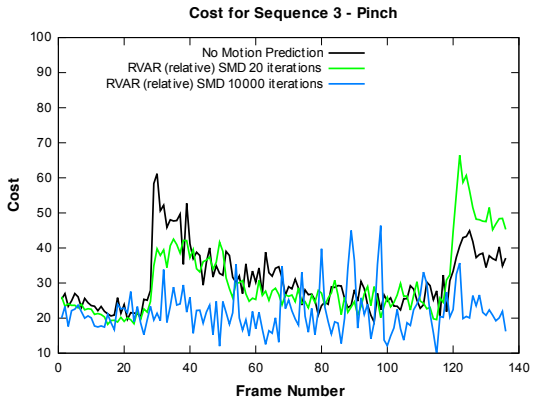

Figure 6.38: Cost per frame for the pinch motion.

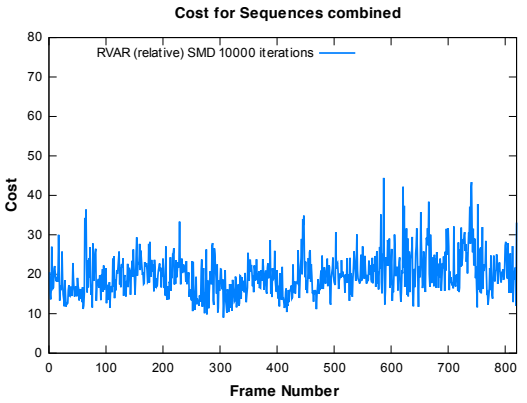

Figure 6.39: Cost per frame for all sequences collated together.

As a final experiment, the tracker is further tested on a longer video sequence (820 frames) to see if the tracker suffers from drifting effects at the limit. This sequence is in fact the five test sequences collated together. One can appreciate from the cost graph in Figure 6.39 that there is minimal if any drifting effects in tracking accuracy, with an average cost of around 22. Selected frames from the sequence in Figure 6.45 show that the tracking system is able to track consistently well throughout the sequence.

In a sense, these results present a stronger empirical confirmation of the main result of the theoretical analysis in Chapter 4, namely that the tracking system has the ability to locally converge to the optimal pose for most cases in spite of noise in the gradient estimates.

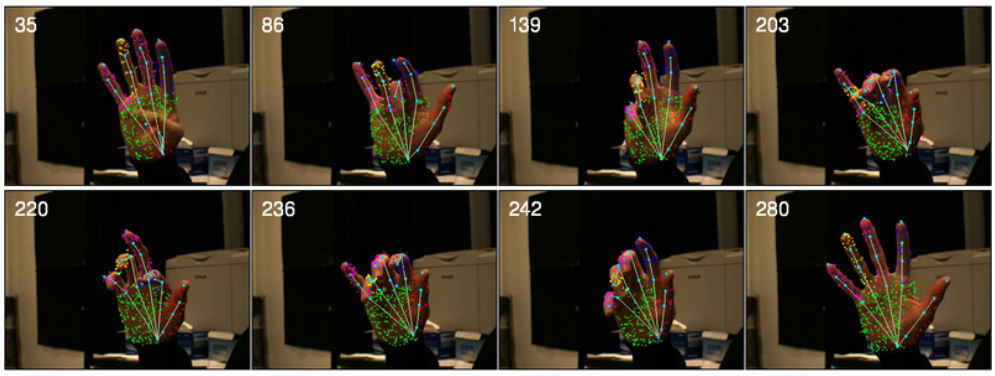

Figure 6.40: Selected frames from the flexion sequence for tracking with the RVAR (relative) prediction model. 10000 iterations are used for each frame.

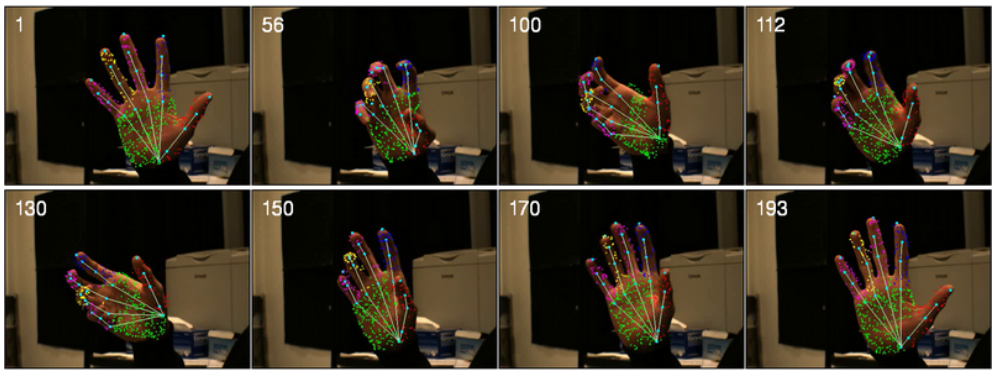

Figure 6.41: Selected frames from the dial turning sequence for tracking with the RVAR (relative) prediction model. 10000 iterations are used for each frame.

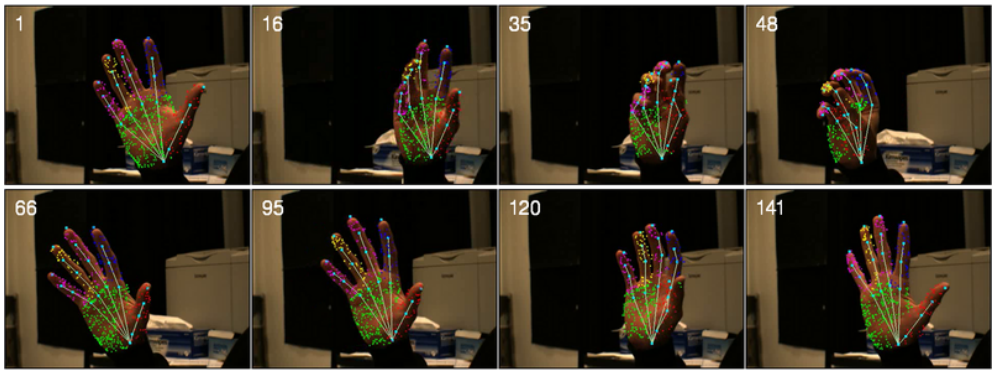

Figure 6.42: Selected frames from the pick and drop sequence for tracking with the RVAR (relative) prediction model. 10000 iterations are used for each frame.

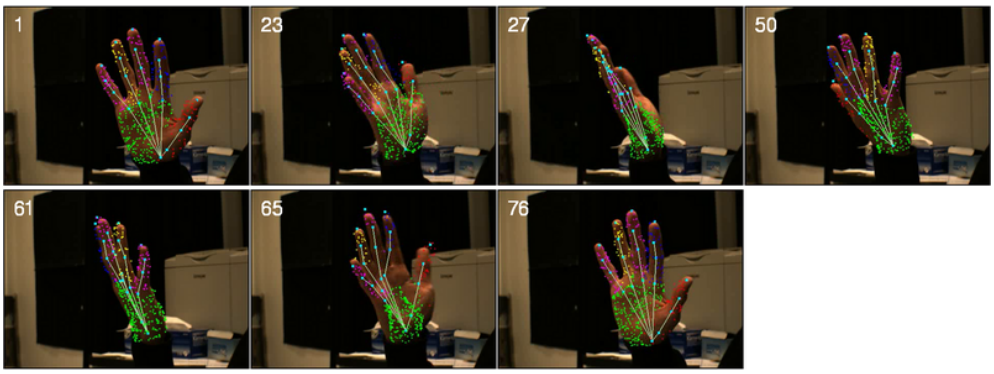

Figure 6.43: Selected frames from the palm rotation sequence for tracking with the RVAR (relative) prediction model. 10000 iterations are used for each frame.

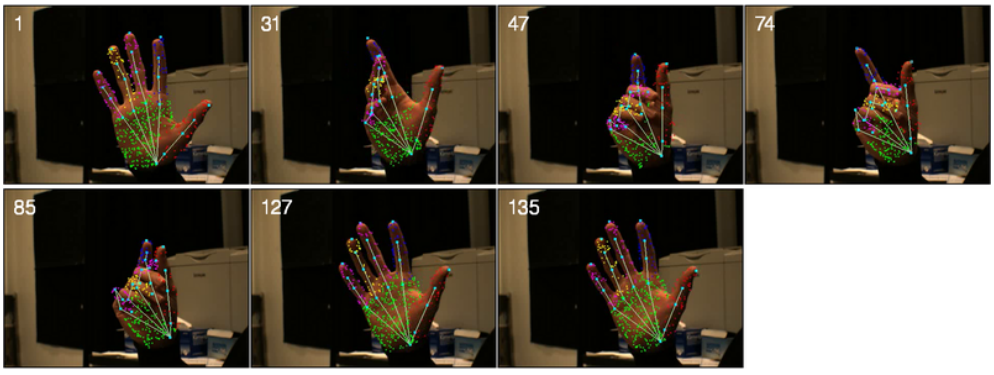

Figure 6.44: Selected frames from the pinch sequence for tracking with the RVAR (relative) prediction model. 10000 iterations are used for each frame.

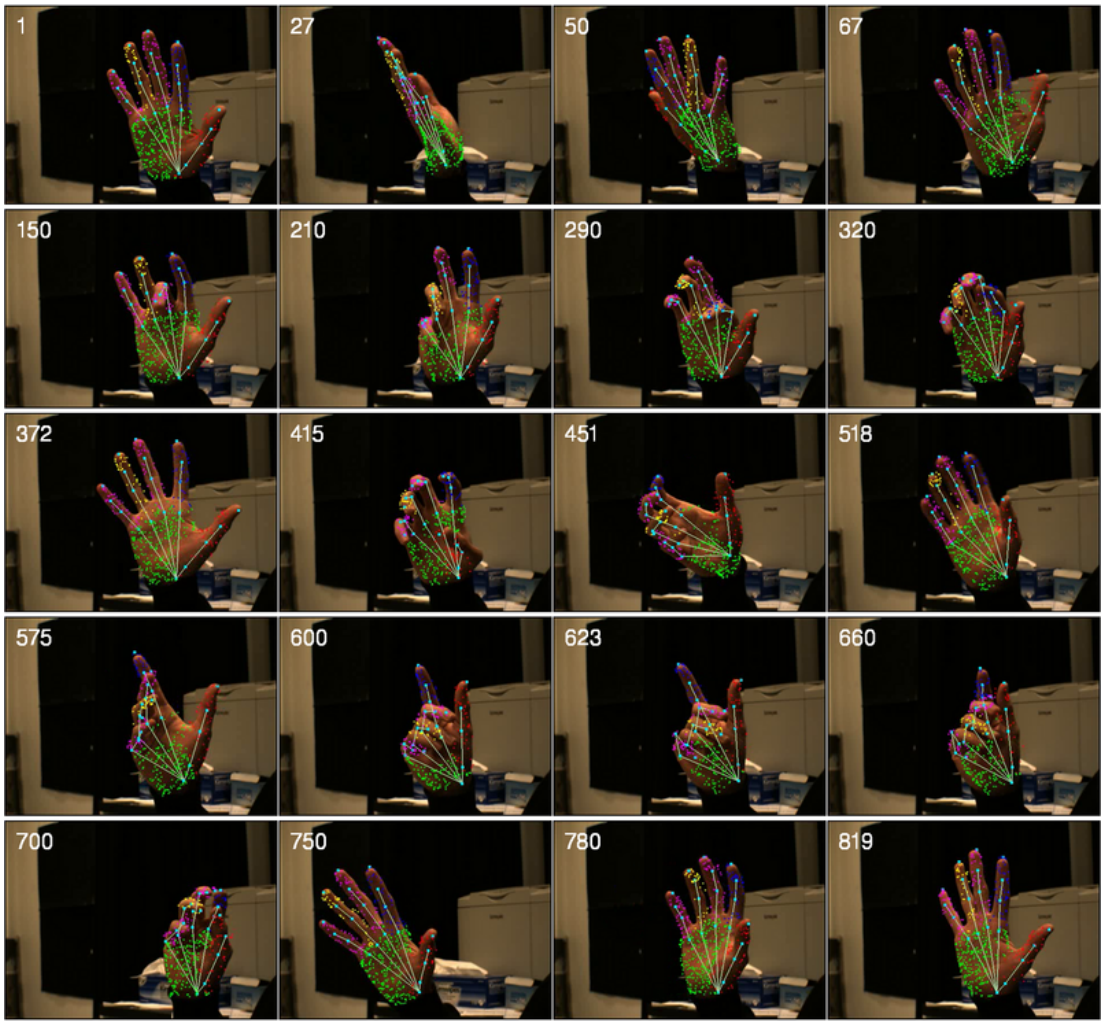

Figure 6.45: Selected frames from the collated sequence for tracking with the RVAR (relative) prediction model. 10000 iterations are used for each frame.

6.7 Summary

It has been demonstrated that motion prediction substantially improves tracking performance. Using a simple deceleration model already improves tracking performance. However, there still remain shortcomings in the deceleration model, as the synthetic and real experimental results have shown. Fast flexion of the digits (*e.g* bending of the middle digit in sequence 1) and motions with heavy self-occlusion such as the palm rotation still prove difficult for a tracker with the deceleration model.

In this chapter an online adaptive VAR prediction model has been presented. The VAR (vector auto-regressive) based predictor is better than the deceleration model in that it has the capacity to model the accelerations of the individual joint movements in an online manner. However the results for the synthetic sequence in Section 6.2 clearly indicate that using an online VAR motion predictor naïvely to find a good initial starting point for the optimisation routine of the tracking system will degrade tracking performance. Applying kinematic-based constraints when solving for the VAR parameters is required for more sensible predictions. The addition of a trust factor and adaptive weights based on the past prediction accuracy has drastically improved results. Experimental results on real sequences support these findings and demonstrate the flexibility of the RVAR model in handling different hand motions. Fast finger flexions are handled well by both the absolute and relative RVAR predictors and the palm rotations are better captured by the RVAR predictors.

Out of the two variants of the RVAR predictors, the relative RVAR predictor performs better overall with a higher consistency in tracking performance. For almost all instances in both the synthetic and real hand sequences, the relative RVAR predictor is arguably superior to the deceleration motion predictor especially for the more challenging hand motions. The experimental results also indicate that the RVAR predictor is more effective at predicting a good starting point for the tracker optimisation routine than the trained AR models. One likely reason why the trained AR models have not performed as well is that the non-linear dynamics of the hand are not adequately captured by a Gaussian mixture of linear dynamic models.

From a practical implementation perspective, the RVAR motion predictor has several advantages over the trained AR models. It is straightforward and substantially quicker to implement than a trained AR model. Unlike the training scheme as described in Section 6.5, one does not have to address the issue of

order selection *i.e.* deciding the number of partitions for the case of the Agarwal predictor. Finally, the RVAR motion predictor is an adaptive prediction scheme that can readily adapt to new primarily unseen hand motions.

Chapter 7

Conclusion

Presented in this thesis is a fast hand tracking system that exhibits promising potential for use as a gestural interface for freeform-modelling applications. The advantages of the tracking system are that it is fast and already flexible enough to handle different hand gestures with minimal offline training. In fact, training is only used to aid in the segmentation of the hand.

The tracking system adopts a gradient-based approach to tracking using hand silhouette and colour intensity information obtained from a pair of cameras. An articulated hand model approximated as a sample point cloud is used to compare the pose estimate to the actual pose seen in the camera images. Mismatch errors from this cost evaluation generate gradients (in the hand parameter space) that are used by a stochastic optimisation algorithm for the refinement of the pose estimate. The use of sparse point sampling allows for a faster evaluation of the pose estimate albeit introducing sampling noise. Noise is also introduced into the tracking system via the camera images. As such, it is appropriate to view the tracking system in a stochastic approximation framework.

This thesis's major contribution is a stochastic convergence analysis of the proposed hand tracking system. The hand tracking system is shown to exhibit properties that allow for local stochastic convergence; namely that the tracker's ideal cost function has a unique global minimum for almost all hand poses and that the Hessian at the unique global minimum is positive definite.

Experimental results however suggest that this unique global minimum has a rather small basin of attraction. For the more challenging sequences, the tracker (without the aid of motion prediction) makes poor pose estimates when the fingers are bending rapidly or exhibiting self-occlusion. As expected, using more sample points for cost evaluation does produce more accurate pose estimates. Since

the tracker's computational time increases linearly with more sample points, one would need to find a compromise between accuracy and speed should one decide to implement an online tracking system. For tracking with a low sample point number, care should be taken to devise a good sampling scheme as the results suggest that the tracking accuracy is sensitive to the sampling scheme under very sparse sampling (*i.e.* ~ 256 sample points).

The experiments have also shown that SMD (Stochastic Meta-Descent) is the optimisation algorithm of choice for a time-constrained tracking system. SMD is able to reach to a good pose estimate in less iterations than other stochastic optimisation algorithms such as stochastic gradient descent and online-BFGS by an order of magnitude. In addition the computational time per iteration for SMD is very low and in fact, just as fast as SG. This is because the cost evaluation procedure is significantly more time consuming to compute than the rest of the update routines in the optimisation algorithm. SMD and SG require the same number of cost evaluations per iteration, and so their computation times are similar.

For the purpose of positioning the tracker within the basin of attraction at the start of each video frame, an online adaptive auto-regressive based motion model has been devised. The online data-driven approach to motion prediction is arguably advantageous when compared to a trained auto-regressive model as it does not involve offline training, is straightforward to implement and, as the experiments have indicated, generalises better to motions that the tracking system has not seen before. The adaptive scheme in the RVAR motion model has been shown to be vital for maintaining reliability in prediction. By actively comparing the recent past predictions against the actual pose estimates at each frame for accuracy, the RVAR motion model obtains a measure of its prediction reliability and is thus able to temper its future predictions accordingly. Finally, by running the tracker at 10,000 allowable iterations per frame with the RVAR (relative) predictor, we have found strong empirical evidence to support the tracker's local convergence property obtained from the theoretical analysis.

7.1 Limitations

The current tracking system is not without its limitations. One such limitation is speed. Despite being able to track with reasonable accuracy at close to 1Hz on a single CPU using a low number of sample points, this is still not fast enough as an

online hand tracking implementation. However, as suggested in Chapter 5, one can readily increase the tracking rate to ∼10Hz by making simple improvements to the tracker implementation such as off-loading image pre-processing tasks onto the GPU and the parallelisation of the cost evaluation routine via threading. With the recent developments in CPU architecture design focusing towards better parallelisation (*e.g.* multi-core processors), it is highly conceivable that computers in the next couple of years will be able to run such a tracking system online at interactive frame rates.

Tracking accuracy also has room for improvement. Even with the aid of a motion predictor there are still instances where the tracking accuracy falters. An example would be the second half of the palm rotating sequence for a low iteration routine *e.g.* the tracker running at a maximum of 20 allowable iterations per frame. Although finger articulations are generally well captured in most cases, one can always add additional visual cues to improve finger alignment (see Future Directions in Section 7.2).

7.2 Future Directions

There are various avenues that one can pursue from this work. On the theoretical side, it has been established that the cost function of this tracking system is well behaved at the unique global minimum. The next logical question to answer is whether the size of the basin of attraction for the unique global minimum can be quantified. This would be interesting to know even if this question can only be answered for a small subset of hand poses. One can imagine that such knowledge will prove extremely helpful in designing a robust gesture alphabet for a gesture interface system. Another challenging theoretical aspect to explore is a stochastic convergence proof for SMD. Empirical results from this thesis and previous work from others [11, 42] suggest that SMD can achieve stochastic convergence. Showing this theoretically, however, has thus far proven to be an elusive problem.

On the practical side, there needs to be more future research on the user-feasibility of articulated 3D hand tracking systems. Previous studies [56, 15] have mainly focused on performing simple tasks on either 2D tabletop interactive systems, or coarse 3D gestures. The potential level of accuracy attainable with an articulated 3D hand tracking system readily lends itself to more complex tasks that require a greater level of control. One such task would be surface modeling

for a 3D software such as Maya. It would be interesting to investigate whether it is indeed more effective for artists to mould surfaces with natural hand gestures as opposed to the traditional keyboard and mouse interface.

Finally, one can explore ways to further improve tracking accuracy of the current tracking system. So far the cost function only exploits the local structure of the hand. Global constraints that make use of the interrelationships between sample points have yet to be exploited. One promising idea (that has not been properly implemented due to lack of time) is the surface normal constraint. Sample points that have the same surface normal on the hand should share the same colour intensity under the uniform texture and the Lambertian surface assumption. With a proper reflectance model, one can formulate this global constraint as another term in the overall cost function. Another tracking aspect well worth exploring is the automatic detection and handling of poor pose estimates. One can envisage that a robust online implementation would be able to automatically detect problematic frames and activate extra subroutine procedures to resolve these frames, *e.g.* increasing the number of iterations and/or sample points.

Appendix A

A.1 Proof for Exception in Case 2

Given that the cylinder lies on the epipolar plane spanned by one end of the cylinder, we want to establish the conditions under which the rotation of a cylinder on the epipolar plane does not cause the cost function $C_{\mathbf{x}^*}$ to increase. This means that for each point on the perturbed cylinder, the projection in both camera views share the same YUV value. Without loss of generality, we assume for convenience that both cameras lie on the same X-Z plane and that the epipolar plane spanned by the cylinder is the same X-Z plane.

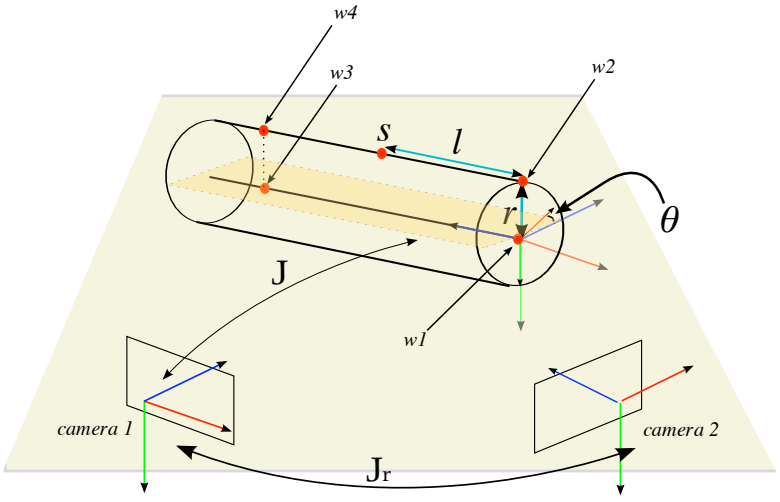

Figure A.1: Rotation of a cylinder on the plane spanned by the x and z axis of the camera. The cylinder's main axis lies on this plane.

Let s be any point on the top edge of the cylinder (see Figure A.1) with its coordinates given as,

$$s := \begin{bmatrix} -l \\ -r \\ 0 \\ 1 \end{bmatrix}.$$

We wish to find an expression for the projection of s in camara 1 in terms of l and r. Let J be the transformation matrix that transforms a point from the local coordinates of the cylinder to the camera coordinates of camera 1. Let θ be the original angle of rotation, $s_\theta := \sin(\theta)$, and $c_\theta := \cos(\theta)$. Note that θ depends on $\mathbf{x}^*$.

$$J := \begin{bmatrix} c_\theta & 0 & s_\theta & k_x \\ 0 & 1 & 0 & k_y \\ -s_\theta & 0 & c_\theta & k_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Let K be the calibration matrix, defined as

$$K := \begin{bmatrix} a_x & 0 & o_x & 0 \\ 0 & a_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Then, with $p := KJs$,

$$p = \begin{bmatrix} -(a_x\, c_\theta - o_x\, , s_\theta))\, l + a_x\, k_x + o_x\, k_z \\ o_y\, s_\theta\, l - a_y\, r + a_y\, k_y + o_y\, k_z \\ s_\theta\, l + k_z \end{bmatrix}.$$

Normalising the depth of p gives the projection p_{norm} of the point s,

$$p_{\text{norm}} = \begin{bmatrix} \dfrac{-(a_x\, c_\theta - o_x\, s_\theta)\, l + a_x\, k_x + o_x\, k_z}{s_\theta\, l + k_z} \\ \dfrac{o_y\, s_\theta\, l - a_y\, r + a_y\, k_y + o_y\, k_z}{s_\theta\, l + k_z} \end{bmatrix}. \tag{A.1}$$

By applying Equation (A.1), the normalised projections of the boundary points w_1, w_2, w_3, w_4 in Figure A.1 can be expressed as

$$\mathrm{w}_1 = \begin{bmatrix} \dfrac{a_x\, k_x + o_x\, k_z}{k_z} \\ \dfrac{a_y\, k_y + o_y\, k_z}{k_z} \end{bmatrix},$$

$$\mathrm{w}_2 = \begin{bmatrix} \dfrac{a_x\, k_x + o_x\, k_z}{k_z} \\ \dfrac{-a_y\, r + a_y\, k_y + o_y\, k_z}{k_z} \end{bmatrix},$$

$$\mathrm{w}_3 = \begin{bmatrix} \dfrac{-(a_x\, c_\theta - o_x\, s_\theta)\, l + a_x\, k_x + o_x\, k_z}{s_\theta\, l + k_z} \\ \dfrac{o_y\, s_\theta\, l + a_y\, k_y + o_y\, k_z}{s_\theta\, l + k_z} \end{bmatrix},$$

and

$$\mathrm{w}_4 = \begin{bmatrix} \dfrac{-(a_x\, c_\theta - o_x\, s_\theta)\, l + a_x\, k_x + o_x\, k_z}{s_\theta\, l + k_z} \\ \dfrac{o_y\, s_\theta\, l - a_y\, r + a_y\, k_y + o_y\, k_z}{s_\theta\, l + k_z} \end{bmatrix}.$$

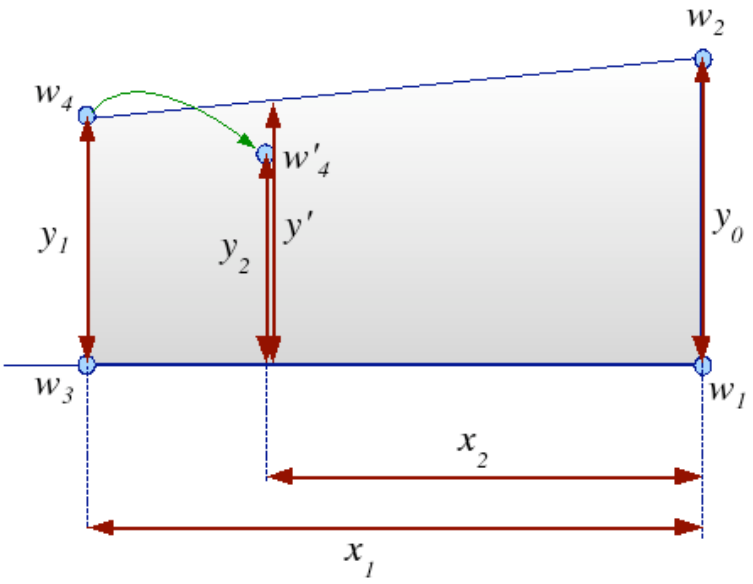

Figure A.2: The projection of the cylinder on the camera image plane.

Figure A.2 shows the projections of these points in the camera image. One should note that the colour distribution on the line connecting w_4 to w_3 is just a scaled version of the colour distribution of the line connecting w_2 to w_1. Hence one can express the colour of a point as the ratio $q := \frac{y}{y'}$ (*e.g.* in Figure A.2 the ratio for w'_4 is $\frac{y_2}{y'}$). Suppose that after a perturbation of $\triangle\theta$, the resulting rotation

angle is $\beta = \theta + \triangle\theta$. Then w_4 will move to a new position w_4'. q is evaluated at w_4' to determine the colour. y' is given as

$$y' = y_0 - (y_0 - y_1)\frac{x_2}{x_1}, \tag{A.2}$$

where

$$y_0 = \frac{a_y\, r}{k_z},$$

$$y_1 = \frac{a_y\, r}{c_\theta\, l + k_z},$$

$$x_1 = -\frac{l\, a_x\, (k_z\, c_\theta + k_x\, s_\theta)}{(s_\theta\, l + k_z)\, k_z},$$

and

$$x_2 = -\frac{l\, a_x\, (k_z\, c_\beta + k_x\, s_\beta)}{(s_\beta\, l + k_z)\, k_z}.$$

y' for w_4' is thus

$$-\frac{a_y\, r\, (-s_\beta\, l\, c_\theta - k_z\, c_\theta - k_x\, s_\theta + s_\theta\, l\, c_\beta)}{s_\beta\, l\, k_z\, c_\theta + s_\beta\, l\, k_x\, s_\theta + {k_z}^2\, c_\theta + k_z\, k_x\, s_\theta},$$

and y_2 is

$$y_2 := \frac{a_y\, r}{s_\beta\, l + k_z}.$$

Therefore, the ratio $q = \frac{y_2}{y'}$ is

$$q = \frac{k_z\, c_\theta + k_x\, s_\theta}{-s_\beta\, l\, c_\theta - k_z\, c_\theta - k_x\, s_\theta + s_\theta\, l\, c_\beta},$$

which simplifies to

$$q = \frac{k_z\, c_\theta + k_x\, s_\theta}{k_z\, c_\theta + k_x\, s_\theta + l s_{\triangle\theta}}. \tag{A.3}$$

For convenience, define T as

$$T := k_{z_i}\, c_\theta + k_{x_i}\, s_\theta,$$

Hence expression (A.3) becomes

$$q = \frac{T}{T + l s_{\triangle\theta}}. \tag{A.4}$$

In the same manner the q ratio for camera 2 can be evaluated. For clarity, from now on, we will denote q_1 as the q ratio for camera 1 and q_2 as the q ratio for camera 2. For the colour of both projections of any given point on the cylinder to be the same in both cameras, the following equality must hold,

$$q_1 = q_2, \tag{A.5}$$

or in the expanded form

$$\frac{T_1}{T_1 + ls_{\triangle\theta}} = \frac{T_2}{T_2 + ls_{\triangle\theta}}. \tag{A.6}$$

There are two scenarios where Equation A.6 holds. One is that there is no perturbation *i.e.* $\triangle\theta = 0$. The other scenario is when $T_1 = T_2$. The latter scenario is the exception set.

T_2 can be broken down and expressed in terms of the original parameters that describe T_1. Let J_1 be the transformation matrix that transforms a point from the local coordinates of the cylinder to the camera coordinates of camera 1. Let J_r be the transformation that takes a point in the coordinates of camera 1 to the camera coordinates of camera 2. Therefore, the transformation J_2 that transforms the point s from the local coordinates to the coordinates of camera 2 can be expressed as $J_2 = J_r J_1$, *i.e.*

$$J_2 = \begin{pmatrix} c_{\theta_r} & 0 & s_{\theta_r} & D_x \\ 0 & 1 & 0 & D_y \\ -s_{\theta_r} & 0 & c\theta_r & D_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_\theta & 0 & s_\theta & kx_1 \\ 0 & 1 & 0 & ky_1 \\ -s_\theta & 0 & c_\theta & kz_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} c_{\theta_r+\theta} & 0 & s_{\theta_r+\theta} & c_{\theta_r} kx_1 + s_{\theta_r} kz_1 + D_x \\ 0 & 1 & 0 & ky_1 + D_y \\ -s_{\theta_r+\theta} & 0 & c_{\theta_r+\theta} & -s_{\theta_r} kx_1 + c_{\theta_r} kz_1 + D_z \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

It is clear that

$$k_{x_2} = c_{\theta_r} k_{x_1} + s_{\theta_r} k_{z_1} + D_x, \tag{A.7}$$

$$k_{z_2} = -s_{\theta_r} k_{x_1} + c_{\theta_r} k_{z_1} + D_z. \tag{A.8}$$

Therefore T_2 can be rewritten as

$$\begin{aligned} T_2 = \quad & (c_{\theta_r} k_{x_1} + s_{\theta_r} k_{z_1} + D_x) s_{\theta+\theta_r} + ... \\ & (-s_{\theta_r} k_{x_1} + c_{\theta_r} k_{z_1} + D_z) c_{\theta+\theta_r} \\ \\ = \quad & k_{x_1}(c_{\theta_r} s_{\theta+\theta_r} - s_{\theta_r} c_{\theta+\theta_r}) + ... \\ & k_{z_1}(s_{\theta_r} s_{\theta+\theta_r} + c_{\theta_r} c_{\theta+\theta_r}) + ... \\ & D_x s_{\theta+\theta_r} + D_z c_{\theta+\theta_r} \\ \\ = \quad & k_{x_1} s_{\theta+\theta_r-\theta_r} + k_{z_1} c_{\theta+\theta_r-\theta_r} + ... \\ & D_x s_{\theta+\theta_r} + D_z c_{\theta+\theta_r} \\ \\ = \quad & k_{x_1} s_\theta + k_{z_1} c_\theta + D_x s_{\theta+\theta_r} + D_z c_{\theta+\theta_r}. \end{aligned}$$

Substituting into the equality $T_1 = T_2$ yields

$$D_x s_{\theta+\theta_r} + D_z c_{\theta+\theta_r} = 0. \tag{A.9}$$

Hence, for the projection of s in both cameras to have the same colour after a perturbation, the original θ has to be

$$\theta = tan^{-1}(-\frac{D_z}{D_x}) - \theta_r. \tag{A.10}$$

□

Appendix B

B.1 Choosing the Rotation Coordinate System in Section 4.5.4

Choosing an appropriate coordinate system for parameterising the rotation of the rigid body simplifies the analysis for showing the positivity of $\hat{H}$. As an illustration, suppose we used the same rotation parameterisation (Euler angles $\theta_x, \theta_y, \theta_z$) as that in the tracker's 3D-2D projection pipeline, *i.e.*

$$\Gamma_i(\omega_i, \mathbf{x}) = R_x R_y R_z \omega_i + t^0_{\text{rigid}}, \tag{B.1}$$

where the rotation matrices R_x, R_y, R_z are parameterised by the Euler angles $\theta_x, \theta_y, \theta_z$, respectively (recall that $\theta_x, \theta_y, \theta_z, t^0_{\text{rigid}}$ are elements of $\mathbf{x}$). The corresponding Jacobian would be

$$J_{\Gamma_i}(\mathbf{x}) = [\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \frac{\partial \Gamma_i}{\partial \theta_x}, \frac{\partial \Gamma_i}{\partial \theta_y}, \frac{\partial \Gamma_i}{\partial \theta_z}], \tag{B.2}$$

where

$$\frac{\partial \Gamma_i}{\partial \theta_x} = \frac{\partial R_x}{\partial \theta_x} R_y R_z \omega_i, \tag{B.3}$$

$$\frac{\partial \Gamma_i}{\partial \theta_y} = R_x \frac{\partial R_y}{\partial \theta_y} R_z \omega_i, \tag{B.4}$$

$$\frac{\partial \Gamma_i}{\partial \theta_z} = R_x R_y \frac{\partial R_z}{\partial \theta_z} \omega_i. \tag{B.5}$$

If the pose $\mathbf{x}$ is at $\theta_x = 0, \theta_y = 0, \theta_z = 0$, then R_x, R_y and R_z would be the identity. One can think of this situation as the rigid body's coordinate system being aligned

to the world coordinate system*. This coordinate system is fixed with respect to the rigid body. Therefore, as the orientation of the rigid body changes, R_x, R_y, R_z changes accordingly. This complicates the evaluation of $\frac{\partial \Gamma_i}{\partial \theta_x}, \frac{\partial \Gamma_i}{\partial \theta_y}, \frac{\partial \Gamma_i}{\partial \theta_z}$.

Instead one can choose to parameterise the rotation of the rigid body such that the coordinate system is fixed with respect to the world/camera 1 instead of the rigid body. One can think of this as merely relabelling the sample points on the rigid body with respect to the camera view 1.

Let $f(\omega_i, \mathbf{x}) := R_x R_y R_z \omega_i$ and $\tilde{\omega}_i := f(\omega_i, \mathbf{x})$. Also let $R_{\tilde{x}}, R_{\tilde{y}}, R_{\tilde{z}}$ be rotation matrices parameterised by the Euler angles $\tilde{\theta}_x = 0, \tilde{\theta}_y = 0, \tilde{\theta}_z = 0$. Then $\Gamma_i(\omega_i, \mathbf{x})$ can be equivalently expressed (in a coordinate system that is aligned with camera 1) as:

$$\Gamma_i(\mathbf{x}) = R_{\tilde{x}} R_{\tilde{y}} R_{\tilde{z}} f(\omega_i, \mathbf{x}) + t^0_{\text{rigid}}. \tag{B.6}$$

The resulting Jacobian that is easier to evaluate for analysis would be

$$J_{\Gamma_i} = [\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \frac{\partial \Gamma_i}{\partial \tilde{\theta}_x}, \frac{\partial \Gamma_i}{\partial \tilde{\theta}_y}, \frac{\partial \Gamma_i}{\partial \tilde{\theta}_z}], \tag{B.7}$$

where

$$\frac{\partial \Gamma_i}{\partial \tilde{\theta}_x} = \frac{\partial R_{\tilde{x}}}{\partial \tilde{\theta}_x} R_{\tilde{y}} R_{\tilde{z}} \tilde{\omega}_i, \tag{B.8}$$

$$\frac{\partial \Gamma_i}{\partial \tilde{\theta}_y} = R_{\tilde{x}} \frac{\partial R_{\tilde{y}}}{\partial \tilde{\theta}_y} R_{\tilde{z}} \tilde{\omega}_i, \tag{B.9}$$

$$\frac{\partial \Gamma_i}{\partial \tilde{\theta}_z} = R_{\tilde{x}} R_{\tilde{y}} \frac{\partial R_{\tilde{z}}}{\partial \tilde{\theta}_z} \tilde{\omega}_i. \tag{B.10}$$

$R_{\tilde{x}}, R_{\tilde{y}}, R_{\tilde{z}} = I$ at all times since $\tilde{\theta}_x, \tilde{\theta}_y, \tilde{\theta}_z$ is fixed at zero at all times. After finding the partial derivatives with respect to $\tilde{\theta}_x, \tilde{\theta}_y, \tilde{\theta}_z$ and some algebraic manipulation, one can show that

$$J_{\Gamma_i} = \begin{bmatrix} 1 & 0 & 0 & 0 & \tilde{\omega}_{i,z} & -\tilde{\omega}_{i,y} \\ 0 & 1 & 0 & -\tilde{\omega}_{i,z} & 0 & \tilde{\omega}_{i,x} \\ 0 & 0 & 1 & \tilde{\omega}_{i,y} & -\tilde{\omega}_{i,x} & 0 \end{bmatrix}. \tag{B.11}$$

This is the Jacobian of the Γ_i function whose rotation is parameterised by a coordinate system that is fixed with respect to camera 1.

*In our case, aligned to the coordinate system of camera 1 since we take the world coordinate system to be the coordinate system of camera 1.

Appendix C

C.1 The existence of $\mathcal{L}_{\text{long}}$

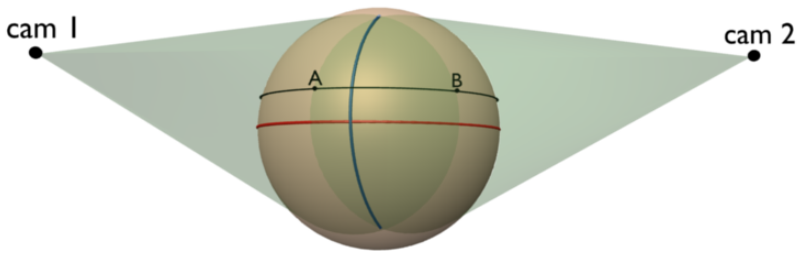

Figure C.1: Trajectories on the spherical surface where $\phi_{i,2} = 0$ are indicated in red and blue. The red trajectory, $\mathcal{L}_{\text{horiz}}$, lies on the equator the sphere. The dark green region on the sphere represents the portion of the sphere visible in both camera views. One can show that the blue trajectory, $\mathcal{L}_{\text{long}}$, always exists within the mutually visible region of the sphere.

Recall from Equation (4.83) on page 55 that

$$\phi_{i,2} \quad = \quad -(s_1 - t_1)^{\mathrm{T}} Q_1 e_2 \left[\frac{1}{\varsigma_{i,1}} - \frac{1}{\varsigma_{i,2}} \right] g_{i,y}, \tag{C.1}$$

where

$$\varsigma_{i,j} := (s_j - t_j)^{\mathrm{T}} Q_j d_j. \tag{C.2}$$

Geometrically speaking, $\varsigma_{i,j}$ is the scaled projection of the surface normal at the sample point ω_i (in the jth camera coordinate frame) on the casting ray d_j (from the jth camera) that intersects ω_i.

Let **A** and **B** be points that lie on the edge of the visible sphere portions for cameras 1 and 2 respectively. **A** and **B** may also lie on an arbitrary latitude **L**

of the sphere (see Figure C.1). Suppose we can pick ω_i along $\mathbf{L}$. As $\omega_i \to \mathbf{A}$, $\varsigma_{i,1}$ goes towards some finite value while $\varsigma_{i,2} \to 0$. The latter is due to the fact that the surface normal at $\mathbf{A}$ will be orthogonal to d_2. Hence

$$\left[\frac{1}{\varsigma_{i,1}} - \frac{1}{\varsigma_{i,2}}\right] \to -\infty. \tag{C.3}$$

Similarly as $\omega_i \to \mathbf{B}$, $\varsigma_{i,1} \to 0$ while $\varsigma_{i,2}$ goes towards some finite value. Hence

$$\left[\frac{1}{\varsigma_{i,1}} - \frac{1}{\varsigma_{i,2}}\right] \to \infty. \tag{C.4}$$

Therefore, using a continuity argument, there will be a point on $\mathbf{L}$ between $\mathbf{A}$ and $\mathbf{B}$ where

$$\left[\frac{1}{\varsigma_{i,1}} - \frac{1}{\varsigma_{i,2}}\right] = 0. \tag{C.5}$$

Repeating this argument for all latitudes on the sphere gives a longitudinal trajectory where $\phi_{i,2} = 0$ for its sample points.

C.2 The slant of $\mathcal{L}_{\text{horiz}}$

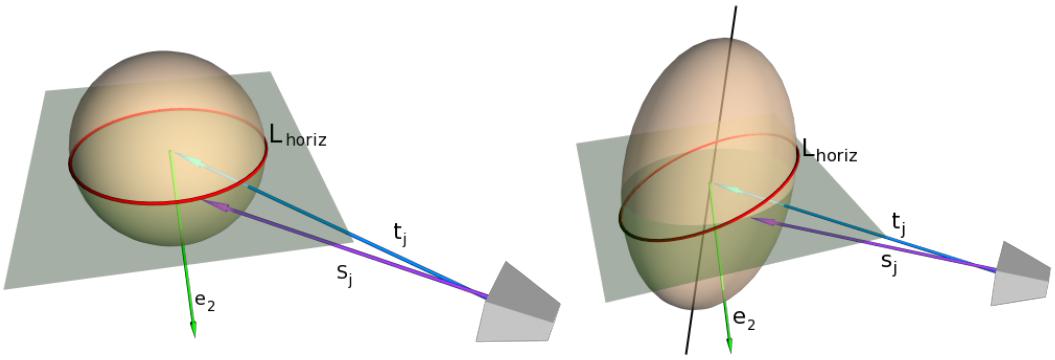

Figure C.2: Left: $\mathcal{L}_{\text{horiz}}$ for a spherical object always lies on the horizontal plane that is orthogonal to e_2, the y-axis of both cameras. Right: $\mathcal{L}_{\text{horiz}}$ for other ellipsoids does not lie on the horizontal plane in general. The skewing is dependent on both the Q matrix and the orientation of the ellipsoid with respect to the cameras.

Recall from Equation (4.83) on page 55 that

$$\phi_{i,2} \quad = \quad -(s_1 - t_1)^{\mathrm{T}} Q_1 e_2 \left[\frac{1}{\varsigma_{i,1}} - \frac{1}{\varsigma_{i,2}}\right] g_{i,y}, \tag{C.6}$$

where

$$\varsigma_{i,j} := (s_j - t_j)^{\mathrm{T}} Q_j d_j. \tag{C.7}$$

$\mathcal{L}_{\text{horiz}}$ describes the path of sample points where $(s_1 - t_1)^{\mathrm{T}} Q_1 e_2 = 0$ and thus $\phi_{i,2} = 0$ as a consequence. Expanding $(s_1 - t_1)^{\mathrm{T}} Q_1 e_2$ gives

$$(s_1 - t_1)^{\mathrm{T}} R_1^{0\,\mathrm{T}} Q R_1^0 e_2. \tag{C.8}$$

$Q = \mathbf{I}$ for the case of a sphere, which simplifies equation (C.8) to

$$(s_1 - t_1)^{\mathrm{T}} e_2. \tag{C.9}$$

Therefore $\mathcal{L}_{\text{horiz}}$ will lie on the horizontal plane (see Figure C.2, left) that contains the sphere's origin and is orthogonal to the y-axes, *i.e.* e_2 of both camera views.

For other ellipsoids, Q is a positive definite diagonal matrix where the diagonal entries are not the same. This has the effect of shearing $\mathcal{L}_{\text{horiz}}$ along the principal directions of the ellipsoid, giving $\mathcal{L}_{\text{horiz}}$ a slant (see Figure C.2, right). Thus $\mathcal{L}_{\text{horiz}}$ lies out of the horizontal plane. The severity of the shear depends on the ratio of the ellipsoid's principal axes. The more anisotropic the ellipsoid, the higher the shear. Note that the shear effect is also dependent on the orientation of the ellipsoid with respect to the camera views. If e_2 aligns exactly to one of the ellipsoid's principal axes, the shearing effect disappears and $\mathcal{L}_{\text{horiz}}$ lies on the horizontal plane as in the spherical case.

Appendix D

D.1 A non-zero Z_3

Lemma D.1.1. *Let $\phi_1, \phi_3 \in \mathbb{R}^3$ be vectors that can be shown to be linearly independent based on their first two entries. In addition let*

$$Z := (\phi_1 \times \phi_3), \tag{D.1}$$

where $\times$ denotes the cross-product operator. Then Z_3 is a non-zero entry.

Proof. Z_3 can be expressed as

$$Z_3 = \phi_{1,1}\phi_{3,2} - \phi_{1,2}\phi_{3,1} = \det(\begin{bmatrix} \phi_{1,1} & \phi_{3,1} \\ \phi_{1,2} & \phi_{3,2} \end{bmatrix}). \tag{D.2}$$

The fact that ϕ_1 and ϕ_3 are linearly independent based on their first two entries imply that $\det(\begin{bmatrix} \phi_{1,1} & \phi_{3,1} \\ \phi_{1,2} & \phi_{3,2} \end{bmatrix}) \neq 0$. Therefore $Z_3 \neq 0$. □

D.2 Structure of $\triangle Y_{i,x}$, $\triangle Y_{i,z}$ and E_i

Recall from Equation (4.95) that $\triangle Y_i$

$$\triangle Y_i = {g_i}^{\mathrm{T}} \left(\hat{\sigma}_1 \begin{bmatrix} 1 & 0 & -\frac{k_{1,x}}{k_{1,z}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \hat{\sigma}_2 {R_1^2}^{\mathrm{T}} \begin{bmatrix} 1 & 0 & -\frac{k_{2,x}}{k_{2,z}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} R_1^2 \right). \tag{D.3}$$

Let $\tilde{g} = g{(R_1^2)}^{\mathrm{T}}$. Then

$$\triangle Y_i = \hat{\sigma}_1 g_{i,x} \begin{bmatrix} 1 & 0 & -\frac{k_{1,x}}{k_{1,z}} \end{bmatrix} - \hat{\sigma}_2 \tilde{g}_{i,x} \begin{bmatrix} 1 & 0 & -\frac{k_{2,x}}{k_{2,z}} \end{bmatrix} R_1^2. \tag{D.4}$$

Factoring out $\frac{\hat{\sigma}_1 g_{i,x}}{k_{1,z}}$ gives

$$\triangle Y_i = \frac{\hat{\sigma}_1 g_{i,x}}{k_{1,z}} \left(\begin{bmatrix} k_{1,z} & 0 & -k_{1,x} \end{bmatrix} - \frac{\hat{\sigma}_2 \tilde{g}_{i,x} k_{1,z}}{\hat{\sigma}_1 g_{i,x}} \begin{bmatrix} 1 & 0 & -\frac{k_{2,x}}{k_{2,z}} \end{bmatrix} R_1^2 \right) \tag{D.5}$$

$$= \frac{\hat{\sigma}_1 g_{i,x}}{k_{1,z}} \left(\begin{bmatrix} k_{1,z} & 0 & -k_{1,x} \end{bmatrix} - \frac{\hat{\sigma}_2 \tilde{g}_{i,x} k_{1,z}}{\hat{\sigma}_1 g_{i,x} k_{2,z}} \begin{bmatrix} k_{2,z} & 0 & -k_{2,x} \end{bmatrix} R_1^2 \right). \tag{D.6}$$

Since K_j is the identity via Assumption 9, $\omega_i^j = k_j$, and so

$$\triangle Y_i = \frac{\hat{\sigma}_1 g_{i,x}}{\omega_{i,z}^1} \left(\begin{bmatrix} \omega_{i,z}^1 & 0 & -\omega_{i,x}^1 \end{bmatrix} - \frac{\hat{\sigma}_2 \tilde{g}_{i,x} \omega_{i,z}^1}{\hat{\sigma}_1 g_{i,x} \omega_{i,z}^2} \begin{bmatrix} \omega_{i,z}^2 & 0 & -\omega_{i,x}^2 \end{bmatrix} R_1^2 \right), \tag{D.7}$$

where ω_i^1 and ω_i^2 are ω_i expressed in the coordinate system of camera 1 and 2 respectively. We define E_i as

$$E_i := -\frac{\hat{\sigma}_2 \tilde{g}_{i,x} \omega_{i,z}^1}{\hat{\sigma}_1 g_{i,x} \omega_{i,z}^2}. \tag{D.8}$$

Note that E_i is a non-linear function of ω_i. Now we wish to express $\begin{bmatrix} \omega_{i,z}^2 & 0 & -\omega_{i,x}^2 \end{bmatrix}$ in terms of $\begin{bmatrix} \omega_{i,z}^1 & 0 & -\omega_{i,x}^1 \end{bmatrix}$. $\begin{bmatrix} \omega_{i,z}^2 & 0 & -\omega_{i,x}^2 \end{bmatrix}^{\mathrm{T}}$ is ω_i^2 that is rotated by $\frac{-\pi}{2}$ on the y-axis and then projected onto the horizontal XZ plane, *i.e.*

$$\begin{bmatrix} \omega_{i,z}^2 & 0 & -\omega_{i,x}^2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \omega_i^2. \tag{D.9}$$

Therefore

$$\left(\begin{bmatrix} \omega_{i,z}^2 & 0 & -\omega_{i,x}^2 \end{bmatrix} R_1^2 \right)^{\mathrm{T}} = (R_1^2)^{\mathrm{T}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \omega_i^2. \tag{D.10}$$

Substituting $\omega_i^2 = (R_1^2)^{\mathrm{T}}(\omega_i^1 + t_2^1)$ gives

$$\left(\begin{bmatrix} \omega_{i,z}^2 & 0 & -\omega_{i,x}^2 \end{bmatrix} R_1^2 \right)^{\mathrm{T}} = (R_1^2)^{\mathrm{T}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} (R_1^2)^{\mathrm{T}} \left(\omega_i^1 + t_2^1 \right), \tag{D.11}$$

which simplifies to

$$\left(\begin{bmatrix} \omega_{i,z}^2 & 0 & -\omega_{i,x}^2 \end{bmatrix} R_1^2 \right)^{\mathrm{T}} = (R_1^2)^{\mathrm{T}} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} (R_1^2)^{\mathrm{T}} \left(\omega_i^1 + t_2^1 \right). \tag{D.12}$$

Since R_1^2 is a rotation on the y-axis (and therefore does not rotate points out of the horizontal plane), one can easily show that

$$(R_1^2)^{\mathrm{T}} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} (R_1^2) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}. \tag{D.13}$$

Thus Equation (D.12) becomes

$$\left(\begin{bmatrix} \omega_{i,z}^2 & 0 & -\omega_{i,x}^2 \end{bmatrix} R_1^2\right)^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \left(\omega_i^1 + t_2^1\right) \tag{D.14}$$

$$= \left(\begin{bmatrix} \omega_{i,z}^1 & 0 & -\omega_{i,x}^1 \end{bmatrix} + \begin{bmatrix} t_{2,z}^1 & 0 & -t_{i,x}^1 \end{bmatrix}\right)^{\mathrm{T}}. \tag{D.15}$$

Substituting this into Equation (D.7) gives

$$\triangle Y_i = \frac{\hat{\sigma}_1 g_{i,x}}{\omega_{i,z}^1} \left((1+E_i) \begin{bmatrix} \omega_{i,z}^1 & 0 & -\omega_{i,x}^1 \end{bmatrix} + E_i \begin{bmatrix} t_{2,z}^1 & 0 & -t_{2,x}^1 \end{bmatrix}\right). \tag{D.16}$$

Therefore in summary,

$$E_i = -\frac{\hat{\sigma}_2 \tilde{g}_{i,x} \omega_{i,z}^1}{\hat{\sigma}_1 g_{i,x} \omega_{i,z}^2}, \tag{D.17}$$

$$\triangle Y_{i,x} = (1+E_i)\omega_{i,z}^1 + E_i t_{2,z}^1, \tag{D.18}$$

$$\triangle Y_{i,z} = -(1+E_i)\omega_{i,x}^1 - E_i t_{2,x}^1. \tag{D.19}$$

Appendix E

E.1 Properties of Ψ and $\bar{\Psi}$

Recall that

$$\phi_i = (\triangle Y_i \begin{bmatrix} 1 & 0 & 0 & 0 & \omega_{i,z} & -\omega_{i,y} \\ 0 & 1 & 0 & -\omega_{i,z} & 0 & \omega_{i,x} \\ 0 & 0 & 1 & \omega_{i,y} & -\omega_{i,x} & 0 \end{bmatrix})^{\mathrm{T}}, \tag{E.1}$$

and by applying Equation (D.17), $\triangle Y_i$ for points on $\mathcal{L}_{\text{horiz}}$ has the following structure

$$\triangle Y_i = \begin{bmatrix} (1+E_i)\omega^1_{i,z} + E_i t^1_{2,z} \\ 0 \\ -(1+E_i)\omega^1_{i,x} - E_i t^1_{2,x} \end{bmatrix}. \tag{E.2}$$

Ψ for the 5DOF case exhibits the following structure (see Equation (4.131) on page 65).

$$\Psi = \{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5\}, \tag{E.3}$$

$$\sim \begin{bmatrix} \phi_{1,1} & \phi_{2,1} & \phi_{3,1} & \phi_{4,1} & \phi_{5,1} \\ 0 & 0 & 0 & 0 & 0 \\ \phi_{1,3} & \phi_{2,3} & \phi_{3,3} & \phi_{4,3} & \phi_{5,3} \\ -\delta_1\phi_{1,3} & -\delta_2\phi_{2,3} & -\delta_3\phi_{3,3} & -N\phi_{4,3} & -(N+\epsilon)\phi_{5,3} \\ * & * & * & * & * \\ \delta_1\phi_{1,1} & \delta_2\phi_{2,1} & \delta_3\phi_{3,1} & N\phi_{4,1} & (N+\epsilon)\phi_{5,1} \end{bmatrix}, \tag{E.4}$$

where $|\delta_i| << 1$ and $N + \epsilon > N >> 0$. From (E.1) it is clear that $\phi_{i,1} = \triangle Y_{i,z}$ and $\phi_{i,3} = \triangle Y_{i,x}$. In our camera setup the hand is located fairly far away from the camera meaning that $|\triangle Y_{i,x}| >> |\triangle Y_{i,z}|$. Therefore $\phi_{i,1}$ will be much larger in magnitude than $\phi_{i,3}$. In addition $\phi_{1,1}, ...\phi_{5,1}$ will be similar in magnitude. The same applies to $\phi_{3,1}, ..., \phi_{3,5}$.

For convenience we will remove row 2 since it does not play a role in showing linear independence for the 5DOF case. We then rearrange the remaining rows of the matrix representation of Ψ to give

$$\bar{\Psi} = \begin{bmatrix} \phi_{1,3} & \phi_{2,3} & \phi_{3,3} & \phi_{4,3} & \phi_{5,3} \\ \phi_{1,1} & \phi_{2,1} & \phi_{3,1} & \phi_{4,1} & \phi_{5,1} \\ * & * & * & * & * \\ -\delta_1\phi_{1,3} & -\delta_2\phi_{2,3} & -\delta_3\phi_{3,3} & -N\phi_{4,3} & -(N+\epsilon)\phi_{5,3} \\ \delta_1\phi_{1,1} & \delta_2\phi_{2,1} & \delta_3\phi_{3,1} & N\phi_{4,1} & (N+\epsilon)\phi_{5,1} \end{bmatrix}. \tag{E.5}$$

Since the magnitude of each column vector does not affect the linear independence of $\bar{\phi}$ we normalise each column based on its respective $\phi_{i,3}$ entry. Thus $\bar{\Psi}$ becomes

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ k_1 & k_2 & k_3 & k_4 & k_5 \\ * & * & * & * & * \\ -\delta_1 & -\delta_2 & -\delta_3 & -N & -(N+\epsilon) \\ \delta_1 k_1 & \delta_2 k_2 & \delta_3 k_3 & Nk_4 & (N+\epsilon)k_5 \end{bmatrix}, \tag{E.6}$$

where $k_1, ..., k_5 >> 1, |\delta_i| << 1$ and $N + \epsilon > N >> 0$.

E.2 A Lower Bound on $|\det(D - CA^{-1}B)|$

$\bar{\Psi}$ exhibits the following structure

$$\bar{\Psi} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \tag{E.7}$$

where

$$A = \begin{bmatrix} 1 & 1 & 1 \\ k_1 & k_2 & k_3 \\ *_1 & *_2 & *_3 \end{bmatrix}, B = \begin{bmatrix} 1 & 1 \\ k_4 & k_5 \\ *_4 & *_5 \end{bmatrix}, \tag{E.8}$$

and

$$C = \begin{bmatrix} \delta_1 & \delta_2 & \delta_3 \\ \delta_1 k_1 & \delta_2 k_2 & \delta_3 k_3 \end{bmatrix}, D = \begin{bmatrix} N & (N+\epsilon) \\ Nk_4 & (N+\epsilon)k_5 \end{bmatrix}. \tag{E.9}$$

Note that $|\delta_i| << 1$ and $N + \epsilon > N >> 0$.

To facilitate analysis, we assume that $k_1 < k_2 < k_3$. This is always achievable by re-ordering the three columns if necessary. In a similar manner, we assume that $k_5 > k_4$. Note that this implies that $\det(D) > 0$. We also assume that the k_i's are similar in value, as are the $*_i$'s. This is valid, as previously justified in Appendix E.1.

To obtain a lower bound on $|\det(D - CA^{-1}B)|$ one needs to determine an upper bound on the elements of the matrix $CA^{-1}B$. A^{-1} is given as

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} k_2 *_3 - k_3 *_2 & -(k_1 *_3 - k_3 *_1) & k_1 *_2 - k_2 *_1 \\ -(*_3 - *_2) & *_3 - *_1 & -(*_2 - *_1) \\ k_3 - k_2 & -(k_3 - k_1) & k_2 - k_1 \end{bmatrix}, \tag{E.10}$$

where $\det(A) = *_1(k_3 - k_2) - *_2(k_3 - k_1) + *_3(k_2 - k_1)$. Since wish to find an upper bound on the elements of $(CA^{-1}B)$, the objective is to first make $|\det(A)|$ as small as possible. It is straightforward to see that the smallest value of $|\det(A)|$ would be of the order $*\triangle k$.

One can expand $A^{-1}B$ to give

$$\frac{1}{\det A} \begin{bmatrix} (k_2 *_3 - k_3 *_2) - k_4(*_3 - *_2) + *_4(k_3 - k_2) & (k_2 *_3 - k_3 *_2) - k_5(*_3 - *_2) + *_5(k_3 - k_2) \\ -(k_1 *_3 - k_3 *_1) + k_4(*_3 - *_1) - *_4(k_3 - k_1) & -(k_1 *_3 - k_3 *_1) + k_5(*_3 - *_1) - *_5(k_3 - k_1) \\ (k_1 *_2 - k_2 *_1) - k_4(*_2 - *_1) + *_4(k_2 - k_1) & (k_1 *_2 - k_2 *_1) - k_5(*_2 - *_1) + *_5(k_2 - k_1) \end{bmatrix}.$$

Using this, one can expand the first element of $CA^{-1}B$ as

$$(CA^{-1}B)_{1,1} = \frac{1}{\det(A)} \begin{pmatrix} \delta_1(*_3(k_3-k_4)+*_2(k_4-k_3)+*_4(k_3-k_2))... \\ +\delta_2(*_3(k_4-k_1)+*_1(k_3-k_4)-*_4(k_3-k_1))... \\ +\delta_3(*_1(k_4-k_2)+*_2(k_1-k_4)+*_4(k_2-k_1)) \end{pmatrix}. \quad \text{(E.11)}$$

Let δ be the upper bound on the set $\{\delta_1, ..., \delta_5\}$. Then

$$(CA^{-1}B)_{1,1} \approx \frac{\delta}{*\triangle k} \begin{pmatrix} (*_3(k_3-k_4)+*_2(k_4-k_3)+*_4(k_3-k_2))... \\ +(*_3(k_4-k_1)+*_1(k_3-k_4)-*_4(k_3-k_1))... \\ +(*_1(k_4-k_2)+*_2(k_1-k_4)+*_4(k_2-k_1)) \end{pmatrix}. \quad \text{(E.12)}$$

Suppose that $k_4 > k_3 > k_2 > k_1$, then one can evaluate from above that the largest possible size of $|(CA^{-1}B)_{1,1}|$ would be in the order of

$$|(CA^{-1}B)_{1,1}| \approx \frac{\delta}{*\triangle k}(5 * \triangle k) \approx 5\delta. \quad \text{(E.13)}$$

The table E.1 shows the size of $|(CA^{-1}B)_{1,1}|$ depending on the relative size of k_4.

Table E.1: Relative size of $|(CA^{-1}B)_{1,1}|$

Relative size of k_4	$\lvert(CA^{-1}B)_{1,1}\rvert$
$k_4 > k_3 > k_2 > k_1$	5δ
$k_3 > k_4 > k_2 > k_1$	7δ
$k_3 > k_2 > k_4 > k_1$	7δ
$k_3 > k_2 > k_1 > k_4$	4δ

Hence the largest value of $|(CA^{-1}B)_{1,1}|$ would be in the order of 7δ. By applying the same method to the other elements of $CA^{-1}B$, one can determine the largest value of $CA^{-1}B$ to be in the order of

$$CA^{-1}B \approx 7\delta \begin{bmatrix} 1 & 1 \\ k & k \end{bmatrix}. \quad \text{(E.14)}$$

Substituting this into $\det(D - CA^{-1}B)$ gives

$$\det(D - CA^{-1)}B) \approx \begin{bmatrix} N & (N+\epsilon) \\ Nk_4 & (N+\epsilon)k_5 \end{bmatrix} - 7\delta \begin{bmatrix} 1 & 1 \\ k & k \end{bmatrix} \tag{E.15}$$

$$\approx N \begin{bmatrix} 1 & 1 \\ k_4 & k_5 \end{bmatrix} - 7\delta \begin{bmatrix} 1 & 1 \\ k & k \end{bmatrix} \tag{E.16}$$

$$\approx (N - 7\delta)(Nk_5 - 7\delta k) - (N - 7\delta)(Nk_4 - 7\delta k) \tag{E.17}$$

$$\approx N^2(k_5 - k_4) - (7\delta N(k_5 - k_4) + 49\delta^2 \triangle k) \tag{E.18}$$

$$\approx N^2(k_5 - k_4) - (7\delta N + 49\delta^2))\triangle k. \tag{E.19}$$

From above, it can be seen that as long as $N^2 >> (7\delta N + 49\delta^2)$, then the $N^2(k_5 - k_4)$ term will dominate and hence $|\det(D - CA^{-1}B)| > 0$.

Based on rough measurements on the palm, N can be made to be ~ -40mm (by choosing the ω_4, ω_5 from the ellipsoid on the side of the palm) and δ can be made to be at most ~ 2.5mm (by choosing the remaining ω's at the top of the palm). Hence

$$N^2 > (7\delta N + 49\delta^2) \tag{E.20}$$

$$1600 > 1006.25, \tag{E.21}$$

at worst and so one can safely say that $|\det(D - CA^{-1}B)| > 0$ in practice.

Bibliography

[1] Opencv. http://opencv.willowgarage.com/wiki/. 76

[2] A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages II: 882–888, 2004. 7, 14, 131

[3] A. Agarwal and B. Triggs. Tracking articulated motion using a mixture of autoregressive models. In *European Conference on Computer Vision*, pages Vol III: 54–65, 2004. 13, 107

[4] E. D. Aguiar, C. Stoll, C. Theobalt, N. Ahmed, S. H., and T. S. Performance capture from sparse multi-view video. *SIGGRAPH*, 2008. 19

[5] V. Athitsos and S. Sclaroff. Estimating 3D hand pose from a cluttered image. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages II: 432–439, 2003. 8, 9

[6] A. Atkinson and M. Riani. *Robust Diagnostic Regression Analysis*. Springer, 1st edition, 2000. 111

[7] A. O. Balan, L. Sigal, and M. J. Black. A quantitative evaluation of video-based 3D person tracking. In *International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 349–356, 2005. 2, 3, 8, 12, 14, 113

[8] J. R. Blum. Multidimensional stochastic approximation methods. *Annals of Mathematical Statistics*, 25:737–744, 1954. 29, 30

[9] G. Borgefors. Distance transformations in digital images. *Comput. Vision Graph. Image Process.*, 34(3):344–371, 1986. 22, 32, 76

[10] J.-Y. Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/. 75

[11] M. Bray, E. Koller-Meier, P. Müller, N. N. Schraudolph, and L. V. Gool. Stochastic Optimization for High-Dimensional Tracking in Dense Range Maps. *IEE Proceedings Vision, Image & Signal Processing*, 152(4):501–512, 2005. 7, 8, 10, 11, 13, 25, 26, 31, 147

[12] M. Bray, E. Koller-Meier, and L. Van Gool. Smart particle filtering for 3D hand tracking. *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 675–680, 2004. 12

[13] M. Bray, E. K. Meier, N. N. Schraudolph, and L. J. V. Gool. Fast stochastic optimization for articulated structure tracking. *Image and Vision Computing*, 25(3):352–364, Mar. 2007. 19, 25, 113

[14] P. Brockwell and R. Davis. *Introduction to Time Series and Forecasting*. Springer, 2nd edition, 2002. 109

[15] M. Cabral, C. Morimoto, and M. Zuffo. On the usability of gesture interfaces in virtual reality environments. In *Proceedings of the 2005 Latin American conference on Human-computer interaction*, pages 100–108. ACM New York, NY, USA, 2005. 80, 147

[16] F. Caillette, A. Galata, and T. Howard. Real-time 3-d human body tracking using learnt models of behaviour. *Comput. Vis. Image Underst.*, 109(2):112–125, 2008. 7

[17] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986. 8

[18] J. Carranza, C. Theobalt, M. Magnor, and H. Seidel. Freeviewpoint video of human actors. In *ACM Transactions on Graphics*, pages 569–577, San Diego, USA, 2003. ACM SIGGRAPH. 7

[19] W. Chang, C. Chen, and Y. Hung. Appearance-guided particle filtering for articulated hand tracking. *IEEE International Conference on Computer Vision and Pattern Recognition*, 1:235–242, 2005. 12

[20] F. Chen, E. Choi, J. Epps, S. Lichman, N. Ruiz, Y. Shi, R. Taib, and M. Wu. A study of manual gesture-based selection for the pemmi multimodal transport management interface. In *ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces*, pages 274–281, New York, NY, USA, 2005. ACM. 1

[21] D. Chik, J. Trumpf, and N. N. Schraudolph. 3d hand tracking in a stochastic approximation setting. In *2nd Human Motion Workshop - Understanding, Modeling, Capture and Animation*, pages 136–151, 2007. 3, 4, 29

[22] D. Chik, J. Trumpf, and N. N. Schraudolph. Using an adaptive VAR model for motion prediction in 3d hand tracking. In *8th Intl. Conf. Automatic Face & Gesture Recognition (FG)*, Amsterdam, Netherlands, 2008. IEEE. 4, 108

[23] T. Cootes, G. Edwards, C. Taylor, et al. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001. 72

[24] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. *Computer Vision and Pattern Recognition*, 2:126–133, 2000. 12

[25] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002. 23, 31

[26] M. Gorce, N. Paragios, and D. J. Fleet. Model-based hand tracking with texture, shading and self-occlusion. In *IEEE Computer Vision and Pattern Recognition or CVPR*, 2008. 2, 7, 8, 10, 11, 13

[27] H. Guan, J. S. Chang, L. Chen, R. S. Feris, and M. Turk. Multi-view appearance-based 3D hand pose estimation. In *Vision for Human-Computer Interaction, CVPRW*, page 154, 2006. 8

[28] H. Guan, R. S. Feris, and M. Turk. The isometric self-organizing map for 3D hand pose estimation. In *International Conference on Automatic Face and Gesture Recognition*, pages 263–268, 2006. 2

[29] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004. 21, 35

[30] D. Heckenberg. Performance evaluation of vision-based high DOF human movement tracking: A survey and human computer interaction perspective. In *Vision for Human-Computer Interaction*, page 156, 2006. 2

[31] K. Hinckley, R. Pausch, J. Goble, and N. Kassell. A survey of design issues in spatial input. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 213–222. ACM Press New York, NY, USA, 1994. 79, 80

[32] E. J. Holden, G. Lee, and R. Owens. Automatic recognition of colloquial australian sign language. In *IEEE Workshop on Motion and Video Computing*, pages II: 183–188, 2005. 1

[33] S. Hou, A. Galata, F. Caillette, N. Thacker, and P. Bromiley. Real-time body tracking using a gaussian process latent variable model. In *International Conference on Computer Vision*, 2007. 2, 3, 12, 14, 107

[34] M. Hu. Visual pattern recognition by moment invariants. *Information Theory, IEEE Transactions on*, 8(2):179–187, 1962. 9

[35] M. Isard and A. Blake. CONDENSATION Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. 14

[36] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96, Jan. 2002. 75

[37] P. Kaimakis and J. Lasenby. Gradient-Based Hand Tracking Using Silhouette Data. *LECTURE NOTES IN COMPUTER SCIENCE*, 4841:24, 2007. 7, 11

[38] P. Kan-John. Advanced human model with dynamic constraints. In *Honours Thesis, CECS, ANU*, 2006. 19, 71

[39] M. Karam and M. Schraefel. A taxonomy of gestures in human computer interactions. Technical report, TR ECSTR-IAM05-009, University of Southampton, 2005. 79

[40] M. Kato, Y. W. Chen, and G. Xu. Articulated hand tracking by PCA-ICA approach. In *International Conference on Automatic Face and Gesture Recognition*, pages 329–334, 2006. 8, 13, 107

[41] R. Kehl, M. Bray, and L. V. Gool. Full body tracking from multiple views using stochastic sampling. In *CVPR*, volume 2, pages 129–136, Washington, DC, USA, 2005. IEEE Computer Society. 3, 7

[42] R. Kehl and L. J. V. Gool. Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding*, 103(2-3):190–209, Nov. 2006. 2, 7, 8, 10, 11, 13, 14, 18, 25, 31, 113, 147

[43] H. Kushner and G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2003. 29

[44] C. S. Lee and A. M. Elgammal. Modeling view and posture manifolds for tracking. In *International Conference on Computer Vision*, pages 1–8, 2007. 13

[45] S.-W. Lee and X. Zhang. Biodynamic modeling, system identification, and variability of multi-finger movements;. *Journal of Biomechanics*, 40(14):3215–3222, 2007. 2

[46] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00*, pages 165–172, New York, NY, USA, 2000. 20

[47] J. Lin, Y. Wu, and T. Huang. 3D model-based hand tracking using stochastic direct search method. *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 693–698, 2004. 10, 12

[48] I. Llamas, B. Kim, J. Gargus, J. Rossignac, and C. Shaw. Twister: a space-warp operator for the two-handed editing of 3D shapes. *ACM Transactions on Graphics (TOG)*, 22(3):663–668, 2003. 1

[49] S. Lu, D. Metaxas, D. Samaras, and J. Oliensis. Using multiple cues for hand tracking and model refinement. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages II: 443–450, 2003. 8, 10, 11, 72

[50] H. Lutkepohl. *New Introduction to Multiple Time Series Analysis*. Springer, 1st edition, 2006. 109, 110, 114

[51] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. *Proc. ECCV*, 2:3–19, 2000. 12

[52] J. McDonald, J. Toro, K. Alkoby, A. Berthiaume, R. Carter, P. Chomwong, J. Christopher, M. Davidson, J. Furst, B. Konie, et al. An improved articulated model of the human hand. *The Visual Computer*, 17(3):158–166, 2001. 19, 20

[53] C. Menier, E. Boyer, and B. Raffin. 3D skeleton-based body pose recovery. In *3DPVT06*, pages 389–396, 2006. 7, 8, 10

[54] C. Meyer. *Matrix analysis and applied linear algebra*. Society for Industrial Mathematics, 2000. 65

[55] B. Michoud, E. Guillou, H. Briceno Pulido, and S. Bouakaz. Real-Time Marker-free Motion Capture from multiple cameras. In *ICCV'2007 : Eleventh IEEE International Conference on Computer Vision*, Oct. 2007. 2, 7, 10

[56] M. Nielsen, M. Storring, T. Moeslund, and E. Granum. A Procedure for Developing Intuitive and Ergonomic Gesture Interfaces for HCI. *Lecture Notes In Computer Science*, pages 409–420, 2004. 79, 80, 147

[57] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999. 26

[58] V. Pavlovic, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):677–695, 1997. 79

[59] A. P. Pentland, J. Weaver, and T. E. Starner. Real-time american sign language recognition using desk and wearable computer based video. In *Massachusetts Institute of Technology, Media Lab*, 1998. 1

[60] F. Quek. Toward a Vision-Based Hand Gesture Interface. In *Virtual Reality Software & Technology: Proceedings of the VRST'94 Conference, 23-26 August 1994, Singapore*. World Scientific, 1994. 79

[61] F. Quek, D. McNeill, R. Bryll, S. Duncan, X. Ma, C. Kirbas, K. McCullough, and R. Ansari. Multimodal Human Discourse: Gesture and Speech. *ACM Transactions on Computer-Human Interaction*, 9(3):171–193, 2002. 1, 79

[62] J. Rehg and T. Kanade. DigitEyes: Vision-Based Human Hand Tracking. *School of Computer Science Technical Report CMU-CS-93-220, Carnegie Mellon Univ., Dec*, 1993. 10

[63] J. M. Rehg and T. Kanade. Visual tracking of high doF articulated structures: An application to human hand tracking. In *European Conference on Computer Vision*, pages B:35–46, 1994. 2, 8, 10

[64] J. M. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *International Conference on Computer Vision*, pages 612–617, 1995. 2, 10

[65] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951. 18, 29, 30

[66] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3D hand pose reconstruction using specialized mappings. *Proc. 8th Int. Conf. on Computer Vision*, 1:378–385, 2001. 9

[67] S. Schkolne. Drawing with the Hand in Free Space: Creating 3D Shapes with Gesture in a Semi-Immersive Environment. *Leonardo*, 35(4):371–375, 2002. 1

[68] N. N. Schraudolph. Local gain adaptation in stochastic gradient descent. In *ICANN*, pages 569–574, Edinburgh, Scotland, 1999. IEE, London. 11, 25, 113

[69] N. N. Schraudolph. Fast curvature matrix-vector products. In *ICANN*, volume 2130 of *Lecture Notes in Computer Science*, pages 19–26, Vienna, Austria, 2001. Springer Verlag, Berlin. 26, 27

[70] N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002. 43

[71] N. N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-Newton method for online convex optimization. In M. Meila and X. Shen, editors, *Proc. 11th Intl. Conf. Artificial Intelligence and Statistics (AIstats)*, pages 433–440, San Juan, Puerto Rico, 2007. Society for Artificial Intelligence and Statistics. 26, 27

[72] N. Shimada. Real-time 3-D hand posture estimation based on 2-D appearance retrieval using monocular camera. In *International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, 2001. 2, 9

[73] H. Sidenbladh, M. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *European Conference on Computer Vision*, pages 784–800, 2002. 107

[74] C. Sminchisescu and B. Triggs. Covariance scaled sampling for monocular 3D body tracking. In *CVPR*, pages I:447–454, 2001. 7

[75] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2007. 31

[76] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla. Filtering using a tree-based estimator. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1063–1070, 2003. 8, 9, 13

[77] B. D. R. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 28, pages 1372–1384, Sept. 2006. 3, 7, 8, 9, 10, 13, 14, 18, 107

[78] H. Stern, J. Wachs, and Y. Edan. Hand gesture vocabulary design: a multi-criteria optimization. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 1, 2004. 79

[79] H. Stern, J. Wachs, and Y. Edan. Optimal Hand Gesture Vocabulary Design Using Psycho-Physiological and Technical Factors. In *7th International Conference Automatic Face and Gesture Recognition, FG2006. Southampton, UK, April*, pages 10–12, 2006. 79, 80

[80] E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. Visual hand tracking using nonparametric belief propagation. In *Workshop on Generative Model Based Vision*, page 189, 2004. 3, 10, 12, 14

[81] A. Sundaresan and R. Chellappa. Multi-camera tracking of articulated human motion using motion and shape cues. In *ACCV*, pages II:131–140, 2006. 7, 8

[82] P. Sunehag, J. Trumpf, S. Vishwanathan, and N. N. Schraudolph. Variable metric stochastic approximation theory. In *Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 560–566, 2009. 31

[83] D. Thelen, F. Anderson, and S. Delp. Generating dynamic simulations of movement using computed muscle control. *Journal of Biomechanics*, 36(3):321–328, 2003. 108

[84] C. Theobalt, J. Carranza, M. A. Magnor, and H.-P. Seidel. Combining 3d flow fields with silhouette-based human motion capture for immersive video. *Graph. Models*, 66(6):333–351, 2004. 7, 8

[85] W. Tsang, K. Singh, and E. Fiume. Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion. In *SCA '05:ACM*

SIGGRAPH/Eurographics symposium on Computer animation, pages 319–328, 2005. 13, 107, 108, 117

[86] D. Vlasic, I. Baran, W. Matusik, and J. Popovic. Articulated Mesh Animation from Multi-view Silhouettes. *SIGGRAPH*, 2008. 7, 19

[87] M. Wasan. *Stochastic Approximation*. Cambridge University Press, 2004. 29, 31

[88] R. Weinstein, E. Guendelman, and R. Fedkiw. Impulse-based control of joints and muscles. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):37–46, 2008. 108

[89] A. Wexelblat. Research Challenges in Gesture: Open Issues and Unsolved Problems. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 1–12, 1998. 79

[90] Y. Wu, J. Lin, and T. Huang. Capturing natural hand articulation. *International Conference on Computer Vision*, pages 426–432, 2001. 13

[91] Y. Wu, J. Lin, and T. S. Huang. Analyzing and capturing articulated hand motion in image sequences. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(12):1910–1922, Dec. 2005. 2, 8, 10, 12, 13

[92] X. Y. Xu and B. X. Li. Learning motion correlation for tracking articulated human body with a rao-blackwellised particle filter. In *International Conference on Computer Vision*, pages 1–8, 2007. 3

[93] H. N. Zhou and T. S. Huang. Tracking articulated hand motion with eigen dynamics analysis. In *International Conference on Computer Vision*, pages 1102–1109, 2003. 2, 3, 13, 14, 107