Model-Based Hand Tracking Using A Hierarchical Bayesian Filter

Björn Dietmar Rafael Stenger

St. John's College

March 2004

UNIVERSITY OF CAMBRIDGE

Dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

Model-Based Hand Tracking Using A Hierarchical Bayesian Filter

Björn Stenger

Abstract

This thesis focuses on the automatic recovery of three-dimensional hand motion from one or more views. A 3D geometric hand model is constructed from truncated cones, cylinders and ellipsoids and is used to generate contours, which can be compared with edge contours and skin colour in images. The hand tracking problem is formulated as state estimation, where the model parameters define the internal state, which is to be estimated from image observations.

In the first approach, an *unscented Kalman filter* is employed to update the model's pose based on local intensity or skin colour edges. The algorithm is able to track smooth hand motion in front of backgrounds that are either uniform or contain no skin colour. However, manual initialization is required in the first frame, and no recovery strategy is available when track is lost.

The second approach, *tree-based filtering*, combines ideas from detection and Bayesian filtering: Detection is used to solve the initialization problem, where a single image is given with no prior information of the hand pose. A hierarchical Bayesian filter is developed, which allows integration of temporal information. This is more efficient than applying detection at each frame, because dynamic information can be incorporated into the search, which helps to resolve ambiguities in difficult cases. This thesis develops a likelihood function, which is based on intensity edges, as well as pixel colour values. This function is obtained from a similarity measure, which is compared with a number of other cost functions. The posterior distribution is computed in a hierarchical fashion, the aim being to concentrate computation power on regions in state space with larger posterior values. The algorithm is tested on a number of image sequences, which include hand motion with self-occlusion in front of a cluttered background.

Acknowledgements

First and foremost, I wish to thank my supervisors Roberto Cipolla and Philip Torr who have both continuously shared their support and enthusiasm. Working with them has been a great pleasure.

Over the past years I have also enjoyed working with a number of fellow PhD students, in particular Paulo Mendonça and Arasanathan Thayananthan, with whom the sharing of authorship in several papers reflects the closeness of our cooperation. Present and former colleagues in the vision group have made the lab an excellent place to work, including Tom Drummond, Paul Smith, Adrian Broadhurst, Kenneth Wong, Anthony Dick, Duncan Robertson, Yongduek Seo, Björn Johansson, Martin Weber, Oliver Williams, Jason McEwen, George Vogiatzis, Jamie Shotton, Ramanan Navaratnam, and Ognjen Arandjelović.

The financial support of the Engineering and Physical Sciences Research Council, the Gottlieb-Daimler and Karl-Benz Foundation, the Cambridge European Trust and Toshiba Research have made my stay in Cambridge possible. I also thank St. John's College and the Department of Engineering for supporting my participation in conferences and seminars and for providing a fine work environment.

A very big 'thank you' goes to Sofya Poger for help with the data glove, to Nanthan, Ram, Ollie, Martin, Oggie, Phil, Bill Gibson and Simon Redhead for the time and effort put into proof-reading, and to Rei for the *Omamori* charm.

Finally, heartfelt thanks go to my family for their immense support along the way.

Björn Stenger

Contents

1 Introduction **2**

1.1 Motivation 2

1.1.1 Limitations of existing systems 3

1.1.2 Approach 5

1.1.3 Limiting the scope 5

1.2 Thesis overview 6

1.2.1 Contributions of this thesis 6

1.2.2 Thesis outline 7

2 Related Work **8**

2.1 Hand tracking 8

2.1.1 2D hand tracking 9

2.1.2 Model-based hand tracking 10

2.1.3 View-based gesture recognition 14

2.1.4 Single-view pose estimation 16

2.2 Full human body tracking 18

2.3 Object detection 21

2.3.1 Matching shape templates 22

2.3.2 Hierarchical classification 24

2.4 Summary 26

3 A 3D Hand Model Built From Quadrics **27**

3.1 Projective geometry of quadrics and conics 27

3.2 Constructing a hand model 33

3.3 Drawing the contours 35

3.4 Handling self-occlusion 38

3.5 Summary 39

4 Model-Based Tracking Using an Unscented Kalman Filter **40**
4.1 Tracking as probabilistic inference 40
4.2 The unscented Kalman filter 43
4.3 Object tracking using the UKF algorithm 45
4.4 Experimental results 48
4.5 Limitations of the UKF tracker 51
4.6 Summary 52

5 Similarity Measures for Template-Based Shape Matching **55**
5.1 Shape matching using distance transforms 56
5.1.1 The Hausdorff distance 60
5.2 Shape matching as linear classification 61
5.2.1 Comparison of classifiers 62
5.2.2 Experimental results 64
5.2.3 Discussion 67
5.3 Modelling skin colour 68
5.3.1 Comparison of classifiers 71
5.3.2 Experimental results 72
5.4 Combining edge and colour information 75
5.4.1 Experimental results 76
5.5 Summary 78

6 Hand Tracking Using A Hierarchical Filter **82**
6.1 Tree-based detection 82
6.2 Tree-based filtering 83
6.2.1 The relation to particle filtering 87
6.3 Edge and colour likelihoods 89
6.4 Modelling hand dynamics 90
6.5 Tracking rigid body motion 91
6.5.1 Initialization 92
6.6 Dimensionality reduction for articulated motion 94
6.6.1 Constructing a tree for articulated motion 96
6.7 Summary 99

7 Conclusion **103**
7.1 Summary 103

7.2 Limitations 104
7.3 Future work 104

List of Figures

1.1 Problem statement 3

2.1 Components of a model-based tracking system 11
2.2 Components of a view-based recognition system 15
2.3 Components of a pose estimation system 17
2.4 Cascade of classifiers 25

3.1 Examples of quadratic surfaces 28
3.2 A truncated ellipsoid 31
3.3 Projecting a quadric 31
3.4 The geometric hand model 33
3.5 Finger blueprints 34
3.6 Projecting cones and cylinders 37
3.7 Projected contours of the hand model 39

4.1 Flowchart of the UKF tracker 45
4.2 Tracking a pointing hand using local edge features 47
4.3 Tracking a pointing hand in two views 48
4.4 Tracking a pointing hand with thumb motion in two views 49
4.5 Tracking an open hand using local edge features 50
4.6 Error comparison for different features 51
4.7 Tracking a pointing hand using skin colour edges 51

5.1 Computing the distance transform 57
5.2 Chamfer cost function in translation space 58
5.3 Chamfer matching using multiple orientations 60
5.4 Templates and feature maps used for edge-based classification 63
5.5 Examples of training and test images 64
5.6 Including edge orientation improves classification performance 65

5.7 ROC curves for edge-based classifiers 66
5.8 Colour likelihood in translation space 70
5.9 Templates and feature maps used for colour-based classification 72
5.10 Colour information helps to exclude background regions 73
5.11 ROC curves for colour-based classifiers 73
5.12 Efficient evaluation of colour likelihoods 75
5.13 Determining the weighting factor in the cost function 76
5.14 Detection with integrated edge and colour features 77
5.15 Detection results using edge and colour information 80
5.16 Error comparison between UKF tracking and detection 81

6.1 Hierarchical partitioning of the state space 85
6.2 Discretizing the filtering equations 86
6.3 Tree-based estimation of the posterior density 88
6.4 Tracking a pointing hand 92
6.5 Error performance 92
6.6 Tracking out-of-image-plane rotation 93
6.7 Fast motion and recovery 93
6.8 Automatic initialization 94
6.9 Search results at different levels of the tree 95
6.10 Variation along principal components 97
6.11 Paths in the eigenspace 97
6.12 Partitioning the state space 98
6.13 Tracking articulated hand motion 100
6.14 Tracking a hand opening and closing with rigid body motion 102

1 Introduction

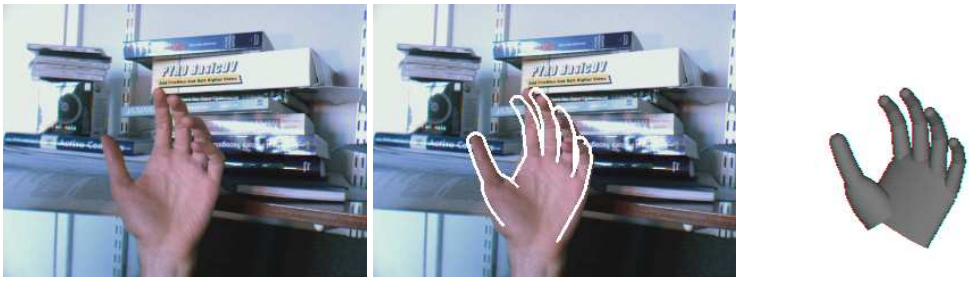
But finding a hand-detector certainly did not allow us to program one.

David Marr (1945 – 1980)

1.1 Motivation

The problem addressed in this thesis is the automatic recovery of the three-dimensional hand pose from a single-view video sequence. Additionally, extensions to multiple views are shown for a few cases. An example image from an input sequence is given in figure 1.1, along with a 3D hand model in the recovered pose.

Tracking objects through image sequences is one of the fundamental problems in computer vision, and recovering articulated motion has been an area of research since the 1980s [59, 102]. The particular problem of hand tracking has been investigated for over a decade [29, 30, 109], and continues to attract research interest as the potential applications are appealing: A vision of computers or robots that are able to make sense of human hand motion has inspired researchers and film directors alike [15, 31, 88, 127, 128]. At the centre of interest is the development of human computer interfaces (HCI), where a computer can interpret hand motion and gestures and attach certain actions to them. Some examples are the manipulation of virtual objects, and the control of a virtual hand (an avatar) or a mechanical hand. If the pose of a *pointing* hand is recovered accurately, this can be interpreted by the computer as "selecting" a particular object, either on a screen or in the environment, and an action can be easily associated. A "digital desk" [146] may be a useful input device in museums or other public places, and touch-free input is also interesting for computer games, where it is already commercially available in a relatively simple form (e.g. the *EyeToy* system for the *Sony Playstation 2*).



(a) (b) (c)

Figure 1.1: **The problem addressed in this thesis.** *The three-dimensional pose of a hand is recovered from an input image.* **(a)** *The input image of a hand in front of cluttered background,* **(b)** *with the projected model contours superimposed, and* **(c)** *the corresponding 3D model in the recovered pose.*

1.1.1 Limitations of existing systems

Given the complexity of the task, most approaches in the past have made a number of assumptions that greatly simplify the problem of detecting or tracking a hand. These include the following restrictions:

- **Uniform or static background:** The use of a simple uniform background (e.g. black or blue) allows clean foreground segmentation of the hand [5, 113, 134]. If the camera is static, background subtraction can be used, assuming that the foreground appearance is sufficiently different from the background [90, 136]. In the case of edge-based trackers, the feature detection stage is made robust by the absence or elimination of background edges [109, 110]. However, using a uniform background is considered too restrictive for a general system, and the technique of background subtraction is unreliable when there are strong shadows in the scene, when background objects move or if the illumination changes [136].

- **Clean skin colour segmentation:** The goal of skin colour segmentation is to obtain the hand silhouette, which can then be used for shape-based matching [88, 119]. It has been shown that in many settings skin colour has a characteristic distribution, which can be distinguished from the colour distribution of background pixels [74]. However, one difficulty of this method is that not only hand regions are obtained, but also other skin coloured regions, such as the arm, the face, or background objects, for example wooden surfaces. The segmentation quality can be significantly improved when special hardware is available such as infrared lights [119].

- **Delimitation of the wrist:** Often it is assumed that the hand in the image can be easily segmented from the arm. If a person is wearing long non-skin coloured sleeves, the extraction of the hand silhouette is much simplified [113, 119, 134]. Lockton and Fitzgibbon [88] make this assumption explicit by requiring the user to wear a wristband. If such an "anchor feature" is supplied, the hand shape can be projected into a canonical frame, thus obtaining the orientation, translation and scale parameters.

- **Manual initialization:** If a three-dimensional model is used for pose estimation, it is often assumed that the model is correctly aligned in the first frame [109, 110]. The problem can then be posed as an optimization problem, wherein the model parameters are adapted in each frame in order to minimize an error function. For user interfaces, however, it is desirable to make the initialization process automatic.

- **Slow hand motion:** Given that the hand is moving slowly, a sequential optimization approach can be taken in which the parameters of a hand model are adapted in each frame. The solution from the previous frame is used to initialize the optimization process in the current frame [109, 110]. However, due to the number of local minima this approach does not work when hand motion is fast or when feature association is difficult due to self-occlusion.

- **Unoccluded fingertips:** If the fingertips can be located in the image, the hand pose can be reconstructed by solving an inverse kinematics problem [149]. However, extracting the fingertip positions is not an easy task, and in many poses not all fingers are visible.

Even though these assumptions hold in certain cases, they are considered too restrictive in the general case. For example, when a single input image is given, no knowledge of the background scene can be assumed. Similarly, it is not guaranteed that no other skin coloured objects are in the scene. Usually people are able to estimate the pose of a hand, even when viewed with one eye. This fact is proof of the existence of a solution to the problem. However, the underlying mechanisms of biological visual information processing, particularly at higher levels of abstraction, are currently not fully understood [81]. In 1972 Gross *et al.* [53] discovered neurons in the inferotemporal cortex of macaque monkeys that showed the strongest response when a hand shape was present in their visual field. However, the fact that such cells should exist does not tell us anything about how they do it. This is the context of David Marr's quote at the beginning of this chapter.

1.1.2 Approach

In this thesis a model-based approach is taken. By "model" we a mean geometric three-dimensional hand model, which has the advantages of compact representation, correspondence with semantics, and the ability to work in continuous space. In the more general sense, our "model" encodes information about the shape, motion and colour that is expected in an image containing a hand. State variables are associated with the geometric model, which are estimated by a filter allowing integration of temporal information and image observations. In the case of hands, edge contours and skin colour features have been used as observations in the past [5, 90, 110, 119] and both of these features are used in this work.

Two solutions are suggested for the estimation problem. The first solution is based on recursive Bayesian filtering using an extension of the Kalman filter, the *unscented Kalman filter* [78]. This approach makes use of a dynamic model to predict the hand motion in each frame and combines the prediction with an observation to obtain a pose estimate. It is shown that this approach works under certain assumptions, but it is too restrictive in the general setting.

The second approach combines ideas from detection techniques and Bayesian filtering. A detection approach is used to solve the initialization problem, where a single image is given with no prior information of the hand pose. This is a hard task and raises a number of issues, such as how to find the location of a hand and how to discriminate between different poses. Perhaps the more important question is whether these two tasks should be separated at all. In this work, template matching is used to solve this problem: A large number of templates are generated by the 3D model and a similarity function based on edge and colour features scores the quality of a match. This is shown to work for a single frame, but using detection in every frame is not efficient. Furthermore, in certain hand poses, e.g. if a flat hand is viewed from the side, there is little image information that can be used for reliable detection. Thus, the idea is to use temporal information in order to increase the efficiency, as well as to resolve ambiguities in difficult cases. This leads to a hierarchical Bayesian filter. The proposed algorithm is shown to successfully recover the hand pose in challenging input sequences, which include hand motion with self-occlusion in front of a cluttered background, and as such represents the current state-of-the-art.

1.1.3 Limiting the scope

Two important applications are explicitly not considered within this thesis. The first is marker-based motion capture, widely used for animation purposes in the film or games

industry, or for biomechanical analysis. Commercial motion capture systems use markers on the object in order to obtain exact point locations. They also work with multiple cameras to resolve ambiguities, which occur in a single view. The use of markers solves one of the most difficult tasks, which is the feature correspondence problem. The number of possible correspondences is reduced drastically, and once these are found, 3D pose recovery becomes relatively easy. In this thesis only the case of markerless tracking is considered. The second application, which we do not attempt to solve, is that of automatic sign language interpretation. Even though recovering 3D pose reliably could be used to interpret finger spelling, such a level of detail may not be required for these applications. For example, the American Sign Language recognition system by Starner *et al.* [128], relies on temporal information and only recovers the hand position and orientation. Also the finger spelling system of Lockton and Fitzgibbon [88], which can recognize 46 different hand poses, does not attempt to recover 3D information.

1.2 Thesis overview

This section states the main contributions of this thesis, and gives a brief outline of the following chapters.

1.2.1 Contributions of this thesis

The main contributions of this thesis are:

- a hierarchical filtering algorithm, which combines robust detection with temporal filtering;
- the formulation of a likelihood function that fuses shape and colour information, significantly improving the robustness of the tracker.

Minor contributions include:

- the construction of a three-dimensional hand model using truncated quadrics as building blocks, and the efficient generation of its projected contours;
- the application of the *unscented Kalman filter* to three-dimensional hand tracking in single and dual views.

1.2.2 Thesis outline

Chapter 2. This chapter presents a literature review of hand tracking, where the main ideas of model-based tracking, view-based recognition and single view pose estimation are discussed. The similarities and differences to full human body tracking are made explicit. Finally, it gives a brief introduction to object detection methods, which are based on hierarchical approaches.

Chapter 3. A method to construct a three-dimensional hand model from truncated cones, cylinders and ellipsoids is presented in this chapter. It starts with an introduction to the geometry of quadric surfaces and explains how such surfaces are projected into the image plane. The hand model, which approximates the anatomy of a real human hand, is then constructed using truncated quadrics as building blocks. It is also explained how self-occlusions can be handled when projecting the model contours.

Chapter 4. This chapter first formulates the task of tracking as a Bayesian inference problem. The unscented Kalman filter is introduced within the context of Bayesian filtering and is applied to 3D hand tracking. Experiments using single and dual views demonstrate the performance of this method, but also reveal its limitations.

Chapter 5. In this chapter similarity measures are examined, which can be used in order to find a hand in images with cluttered backgrounds. The suggested cost function integrates both shape and colour information. The *chamfer distance function* [9] is used to measure shape similarity between a hand template and an image. The colour cost term is based on a statistical colour model and corresponds to the likelihood of the colour data in an image region.

Chapter 6. This chapter describes one of the main contributions of this work, which is the development of an algorithm, which combines robust detection with Bayesian filtering. This allows for automatic initialization and recovery of the tracker, as well as the incorporation of a dynamic model. It is shown how the dimensionality of articulated hand motion can be reduced by analyzing joint angle data captured with a data glove, and examples of 3D pose estimation from a single view are presented.

2 Related Work

Study the past if you would define the future.

Confucius (551 BC–479 BC)

This chapter gives an overview of the developments and the current state-of-the-art in hand tracking. Earlier reviews have been published by Pavlović *et al.* [105] and Wu and Huang [150, 152]. The next section introduces the ideas behind model-based tracking, view-based recognition, and single-view pose estimation, highlighting the merits, as well as the limits of each approach for hand tracking. In section 2.2 the common ground as well as the differences to full human body tracking are pointed out. Finally, section 2.3 gives a brief introduction to general object detection methods, which have yielded good results for locating deformable objects in images.

2.1 Hand tracking

For a long time the work on hand tracking could be divided into two streams of research, *model-based* and *view-based* approaches [105]. Model-based approaches use an articulated three-dimensional (3D) hand model for tracking. The model is projected into the image and an error function is computed, scoring the quality of the match. The model parameters are then adapted such that this cost is minimized. Usually it is assumed that the model configuration at the previous frame is known, and only a small parameter update is necessary. Therefore, model initialization has been a big obstacle. In most cases the model is aligned manually in the first frame. Section 2.1.2 reviews methods for model-based hand tracking.

In the view-based approach, the problem is formulated as a classification problem. A set of hand features is labelled with a particular hand pose, and a classifier is learnt

Model-based 3D tracking	View-based pose estimation
Estimation of continuous 3D parameters	Discrete number of poses
Initialization required	Estimation from a single image
Single or multiple views	Single view
Tracking features reliably is difficult	Estimating nonlinear 3D–2D mapping is difficult

Table 2.1: **Comparison of model-based and view-based approaches.**

from this training data. These techniques have been employed for gesture recognition tasks, where the number of learnt poses has been relatively limited. Section 2.1.3 gives an overview of methods for view-based gesture recognition. A summary of properties of the two approaches is listed in table 2.1.

More recently, the boundaries between model-based and view-based methods have been blurred. In several papers [4, 5, 6, 113, 119] 3D models have been used to generate a discrete set of 2D appearances, which are matched to the image. These appearances are used to generate an arbitrary amount of training data, and a correspondence between 3D pose vector and 2D appearance is given automatically by the model projection. The inverse mapping can then simply be found by view-based recognition, i.e. the estimated pose is given by the 2D appearance with the best match. These methods have the potential to solve many of the problems inherent in model-based tracking.

A number of tracking systems have been shown to work for 2D hand tracking. In the following section, some of these methods are reviewed.

2.1.1 2D hand tracking

Tracking the hand in 2D can be useful in a number of applications, for example simple user interfaces. In this context 2D tracking refers to methods, which use a single camera to track a hand, possibly with some scale changes, and do not recover 3D information about out-of-image-plane rotation or finger configuration. This is a much simpler problem than full 3D tracking for a number of reasons. First of all, the dimension of the search space is much smaller, i.e. there are only four degrees of freedom (DOF) for global motion, which are translation, scale and rotation in the image plane, whereas 3D tracking has six DOF for rigid body motion. Another factor is that hand poses can be easily recognized when

the hand is fronto-parallel to the camera, because the visible area is relatively large and self-occlusion between different fingers is small.

One example of a human computer interface (HCI) application is the vision-based drawing system presented by MacCormick and Isard [90]. The hand shape was modelled using a B-spline, and a particle filter was used to track the hand contour. The state space was partitioned for efficiency, decoupling the motions of index finger and thumb. Background estimation was used to constrain the search region, and seven DOF were tracked in real-time. The tracker was initialized by searching regions which were classified as skin colour [67]. A limitation of this method is the fact that the contour alone is not a reliable feature during fast hand movements.

Laptev and Lindeberg [84] and Bretzner *et al.* [20] presented a system that could track an open hand and distinguish between five different hand states. Scale-space analysis was used to find intensity or colour blob features at different scales, corresponding to palm and fingers. Likelihood maps were generated for these features and were used to evaluate five hypothesized hand states that corresponded to certain fingers being bent or extended. Tracking was effected using a particle filter, and the system operated at 10 frames per second. It was used within a vision-based television remote control application.

Von Hardenberg and Bérard [143] showed the use of an efficient but simple fingertip detector in some HCI applications. The system adaptively estimated the background and searched the foreground region for finger-like shapes at a single scale. This method only works for applications with a steady camera and relies on clean foreground segmentation, which is difficult to obtain in a general setting [136]. However, it requires no initialization and allows for fast hand motion.

To summarize, a number of 2D tracking systems have been shown to work in HCI applications in real time. However, they are difficult to extend to 3D tracking without the use of a geometric hand model. The following section deals with 3D tracking, which has many more potential applications, but at the same time is a very challenging task.

2.1.2 Model-based hand tracking

A model-based tracking system generally consists of a number of components, as depicted in figure 2.1. The geometric hand model is usually created manually, but can also be obtained by reconstruction methods. Models that have been used for tracking are based on planar patches [153, 155], deformable polygon meshes [57] or generalized cylinders [36, 108, 109, 110, 118, 120, 121]. In some works the hand shape parameters are estimated together with the motion parameters, thus adapting the hand model to each user [118, 120,

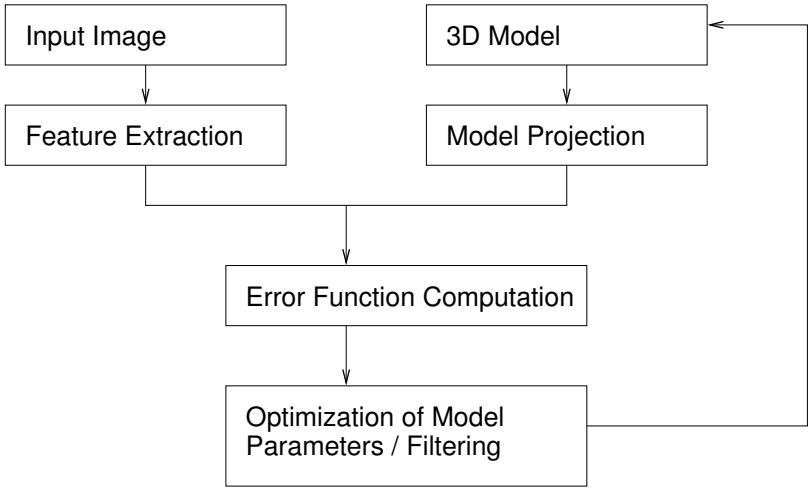


Figure 2.1: **Components of a model-based tracking system.** *A model-based tracking system employs a geometric 3D model, which is projected into the image. An error function between image features and model projection is minimized, and the model parameters are adapted.*

121, 149].

The underlying kinematic structure is usually based on the biomechanics of the hand. Each finger can be modelled as a four DOF kinematic chain attached to the palm, and the thumb may be modelled similarly with either four or five DOF. Together with rigid body motion of the hand, there are 27 DOF to be estimated. Working in such a high dimensional space is particularly difficult because feature points that can be tracked reliably are sparse: the texture on hands is weak and self-occlusion occurs frequently. However, the anatomical design places certain constraints on the hand motion. These constraints have been exploited in different ways to reduce the search space. One type of constraint are the joint angle limits, defining the range of motion. Another very significant type is the correlation of joint movement. Angles of the same finger, as well as of different fingers, are not independent. These constraints can be enforced in different ways, either by parameter coupling or by working in a space of reduced dimension, e.g. one found by PCA [57, 153, 155].

Different dynamic models have been used to characterize hand motion. Standard techniques include extended Kalman filtering with a constant velocity model [120]. A more complex model is the *eigendynamics* model of Zhu *et al.* [155], where the motion of each finger is modelled as a 2D dynamic system in a lower dimensional eigenspace. For 2D motions Isard and Blake [68] have used model switching to characterize different

types of motion. However, global hand motion can be fast in terms of image motion. Even though it is usually smooth, it may be abrupt and difficult to predict from frame to frame. Furthermore, Tomasi *et al.* [134] pointed out that finger flexion or extension can be as short as 0.1 seconds, corresponding to only three frames in an image sequence. These considerations increase the difficulty of modelling hand dynamics in a general setting. An approach suggested in [134] is to identify a number of key poses and interpolate the hand motion between them.

Arguably the most important component is the observation model, in particular the selection of features that are used to update the model. Possible features are intensity edges [57, 109, 110], colour edges [90] or silhouettes derived from colour segmentation [87, 153]. Identification of certain key points has been attempted [118, 120, 121], e.g. looking for fingertips by searching for protrusions in the silhouette shape. In some cases, features have been used that require more computational effort, such as a stereo depth map [36] or optical flow [89]. Several systems combine multiple cues into a single cost function [87, 89, 153, 155].

One of the first systems for markerless hand tracking was that of Cipolla and Hollinghurst [29, 30], who used B-spline snakes to track a pointing hand from two uncalibrated views. In each view the affine transformation of the snake was estimated, and with a ground plane constraint the indicated target position was found. The system required a simple background, but it could operate in real-time.

Three-dimensional articulated hand tracking was pioneered by Rehg and Kanade [109, 110] with their *DigitEyes* tracking system. The system used a 3D hand model, which was constructed from truncated cylinders and had an underlying kinematic model with 27 degrees of freedom. Tracking was performed by minimizing a cost function, which measured how well the model is aligned with the image. Two possible cost functions were suggested, the first is based on edge features [109], the second on template registration [110]. In the first method, the model cylinder axes were projected into the image and local edges were found. Similarly, fingertip positions were found. The error term was the sum of squared differences between projected points and image feature points, and it was minimized using the Gauss-Newton algorithm. The second method of template registration was used in order to deal with self-occlusion. An edge template was associated with each finger segment, together with a window function masking the contribution of occluded segments. Using two cameras and dedicated hardware, the system achieved a rate of ten frames per second when using local edge features. Off-line results were shown for the case of self-occlusion. A neutral background was required for reliable feature

extraction.

Heap and Hogg [57] used a 3D point distribution model [32], which was constructed from a 3D magnetic resonance image data set. The possible hand shapes were a linear combination of the five most significant eigenvectors added to the mean shape. This reduction in the number of parameters allowed for tracking from a single view. However, natural hand motion was not modelled accurately because the 3D shape changes are nonlinear. The model was first deformed, then rotated, scaled and translated, so that the contours matched the edges in the image. The system tracked 12 DOF in a limited range at 10 frames per second.

Delamarre and Faugeras [36] pursued a stereo approach to hand tracking. A stereo-correlation algorithm was used to estimate a dense depth map. In each frame a 3D model was fitted to this depth map using the *iterative closest points* (ICP) algorithm. Using stereo information helped segment the hand from a cluttered background. However, the disparity maps obtained with the correlation algorithm were not very accurate as there are not many strong features on a hand.

Shimada *et al.* [118, 120, 121] used a silhouette-based approach to tracking from single camera input. The positions of finger-like shapes were extracted from the silhouette and matched with the projection of a 3D model. A set of possible pose vectors was maintained and each one was updated using an extended Kalman filter. An optimal path was then found for the whole image sequence. The difficulty in this method is to identify finger-like shapes during unconstrained hand motion, particularly in cases of self-occlusion.

Wu and Huang [149] used different techniques to reduce the dimensionality of the search space. Firstly, constraints on the finger motion were imposed, such as linear dependence of finger joint angles. Furthermore, the state space was partitioned into global pose parameters and finger motion parameters. These sets of parameters were estimated in a two-step iterative algorithm. The limitation of this method is that the fingertips have to be visible in each frame.

In another approach, Wu, Lin and Huang [87, 153] learnt the correlations of joint angles from data using a data glove. It was found that hand articulation is highly constrained and that a few principal components capture most of the motion. For the training data the state space could be reduced from 20 to seven dimensions with loss of only five percent of the information, and finger motion was described by a union of line segments in this lower dimensional space. This structure was used for generating a proposal distribution in a particle filter framework.

Zhou and Huang [155] analyzed hand motion in more detail, and introduced an *eigendynamics* model. As previously, the dimensionality of joint angle data was reduced using PCA. *Eigendynamics* are defined as the motion of a single finger, modelled by a 2D linear dynamic system in this lower dimensional eigenspace, reducing the number of parameters from 20 to 10. As in previous work [149], the global pose and finger motion parameters were estimated separately in an iterative scheme. By using a cost function based on intensity edges as well as colour, the robustness towards cluttered backgrounds was increased.

Lu *et al.* [89] suggested that the integration of multiple cues can assist tracking of articulated motion. In their approach, intensity edges and optical flow were used to generate forces, which act on a kinematic hand model. One limitation of the method, however, is the assumption that illumination does not change over time.

To summarize, 3D hand tracking has been shown to work in principle, but only for relatively slow hand motion and usually in constrained environments, such as plain backgrounds or controlled lighting. The problem is ill-posed in a single view, because in some configurations the motion is very difficult to observe [15, 100]. In order to deal with these ambiguities, most systems use multiple cameras.

A further issue is the initialization of the 3D model. When starting to track or when track is lost, an initialization or recovery mechanism is required, as presented by Williams *et al.* for region-based tracking [147]. This problem has been solved either by manual initialization or by finding the hand in a fixed pose. Generally, this task can be formulated as a hand detection problem, which has been addressed by view-based methods. The next section gives a short review of this approach.

2.1.3 View-based gesture recognition

View-based methods have traditionally been used for gesture recognition. This is often posed as a pattern recognition problem, which may be partitioned into components such as shown in figure 2.2. These methods are often described as “bottom-up” methods, because low-level features are used to infer higher level information. The main problems, which need to be solved in these systems are how to segment a hand from a general background scene and which features to extract from the segmented region. The classification itself is mostly done using a nearest neighbour classifier or other standard classification methods [42]. Generally, the classifier is learnt from a set of training examples and assigns the input image to one of the possible categories. In the following paragraphs only a few selected papers in gesture recognition are described. More thorough reviews can be found in [105, 152].

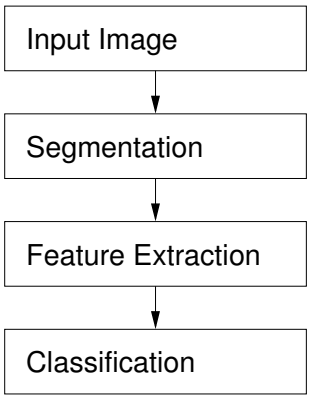


Figure 2.2: **Components of a view-based recognition system.** *View-based recognition is often treated as a pattern recognition problem and a typical system consists of components as shown above. At the segmentation stage the hand is separated from the background. Features are then extracted, and based on these measurements the input is classified as one of many possible hand poses. The classifier is designed using a set of training patterns.*

Cui and Weng [34, 35] presented a system for hand sign recognition, which integrated segmentation and recognition. In an off-line process, a mapping was learnt from local hand subwindows to a segmentation mask. During recognition, regions of image motion were identified, and local subwindows of different sizes were used to hypothesize a number of segmentations. Each hypothesis was then verified by using a classifier on the segmented region. Features were selected by discriminant analysis and a decision tree was used for classification. The system recognized 28 different signs with a recognition rate of 93%. The segmentation step was the computational bottleneck and took approximately one minute per frame on an Indigo workstation, whereas the classification required only about 0.1 seconds. Another limitation of this approach is the assumption that the hand is the only moving object in the scene.

Triesch and Von der Malsburg [137] built a hand gesture interface for robot control. Elastic 2D graph matching was employed to compare twelve different hand gestures to an image. Gabor filter responses and colour cues were combined to make the system work in front of cluttered backgrounds. This system achieved a recognition rate of 93% in simple backgrounds and 86% in cluttered backgrounds.

Wu and Huang [151] introduced a method based on the EM algorithm to combine labelled and unlabelled training data. Using this approach, they classified 14 different hand postures in different views. Two types of feature vectors were compared: a combination of texture and shape descriptors, and a feature vector obtained by principal component analysis (PCA) of the hand images. The features obtained by PCA yielded better perfor-

mance, and a correct classification rate of 92% was reported.

Lockton and Fitzgibbon [88] built a real-time gesture recognition system for 46 different hand poses, including letter spellings of American Sign Language. Their method is based on skin colour segmentation and uses a boosting algorithm [46] for fast classification. The user was required to wear a wristband so that the hand shape can be easily mapped to a canonical frame. They reported a recognition rate of 99.87%; this is the best recognition result achieved so far, however, a limitation of the system is that controlled lighting conditions are needed.

Tomasi *et al.* [134] used a classification approach together with parameter interpolation to track hand motion. Image intensity data was used to train a hierarchical nearest neighbour classifier (24 poses in 15 views), classifying each frame as one of 360 views. A 3D hand model was then used for visualizing the motion. The parameters for the 24 poses were roughly fixed by hand and the model parameters were interpolated for transitions between two poses. The system could handle fast hand motion, but it relied on clean skin colour segmentation and controlled lighting conditions, in a manner similar to that of Lockton and Fitzgibbon.

It may also be noted that many other papers in gesture recognition, such as the sign language recognition system by Starner *et al.* [128], make heavy use of temporal information. American Sign Language has a vocabulary of approximately 6,000 temporal gestures, each representing one word. This is different from finger spelling, which is used for dictating unknown words or proper nouns. The hand tracking stage in [128] does not attempt to recover detail; it recovers only a coarse description of the hand shape, as well as its position and orientation. Methods from automated speech recognition, such as hidden Markov models, can be used for these tasks. Starner *et al.* reported a recognition rate of 98% using a vocabulary of 40 words.

To summarize, view-based methods have been shown to be effective at discriminating between a certain number of hand poses, which is satisfactory for a number of applications in gesture recognition. One of the main problems in view-based methods is the segmentation stage. This is often done by skin colour segmentation, which requires the user to wear long sleeves or a wristband. Another option is to test several possible segmentations, which increase the recognition time.

2.1.4 Single-view pose estimation

More recently, a number of methods have been suggested for hand pose estimation from a single view. A 3D hand model is used to generate 2D appearances, which are matched

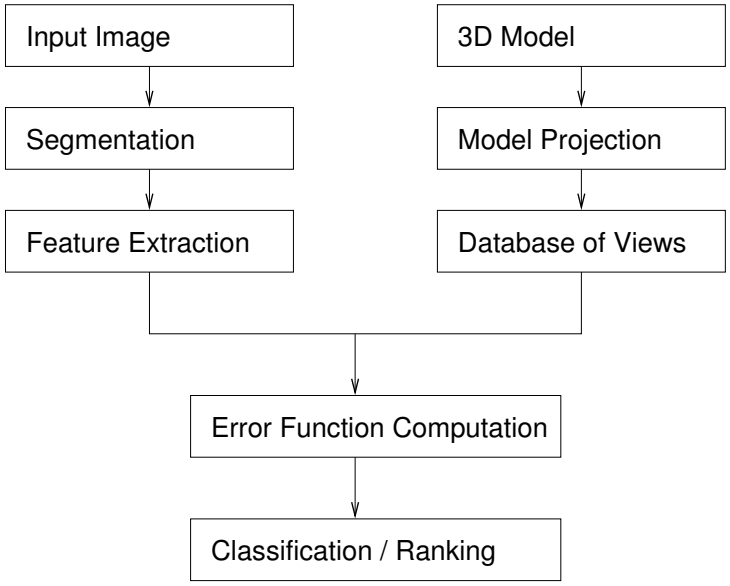


Figure 2.3: **Components of a pose estimation system.** *This figure shows the components of a system for pose estimation from a single frame. A 3D geometric model is used to generate a large number of 2D appearances, which are stored in a database. At each frame, the input image is compared to all or a number of these views. Either a single pose or a ranking of possible poses is returned based on a matching error function.*

to the image, and the mapping between 3D pose and 2D appearance is given directly by projecting the model into the image, see figure 2.3.

Shimada, Kimura and Shirai [119] presented a silhouette-based system for hand pose estimation from a single view. A shape descriptor was extracted and matched to a number of pre-computed models, which were generated from a 3D hand model. A total of 125 poses, each in 128 views was stored, giving a total of 16,000 shapes. In order to avoid exhaustive search at each frame, a transition network between model shapes was constructed. At each time step a set of hypotheses was kept, and only the neighbourhood of each hypothesis was searched. Real-time performance was achieved on a 6 personal computer (PC) cluster. Infra-red illumination and a corresponding lense filter were used to improve colour segmentation.

Rosales *et al.* [113] also estimated the 3D hand pose from the silhouette in a single view. The silhouette was obtained by colour segmentation, and shape moments were used as features. The silhouette was matched to a large number of models, generated by projecting a 3D model in 2,400 different configurations from 86 viewpoints (giving a total of 206,400 shapes). The mapping from feature space to state space was learnt from

example silhouettes generated by a 3D model. A limitation of the two previous methods is that the silhouette shape is not a very rich feature, and many different hand poses can give rise to the same silhouette shape in cases of self-occlusion.

Athitsos and Sclaroff [4, 5] followed a database retrieval approach. A database of 107,328 images was constructed using a 3D model (26 poses in 4128 views). Different similarity measures were evaluated, including chamfer distance, edge orientation histograms, shape moments and detected finger positions. A weighted sum of these was used as a global matching cost. Retrieval times of 3-4 seconds per frame on a 1.2 GHz PC were reported in the case of a neutral background. In [6] the approach was extended to cluttered backgrounds by including a line matching cost term in the similarity measure. Processing time increased to 15 seconds per frame. Retrieval efficiency was addressed in [3], where the symmetric chamfer distance was used as a similarity measure. Fast approximations to the nearest neighbour classifier were explored to reduce the search time. This was done by learning an embedding from the input data into Euclidean space, where distances can be rapidly computed, while trying to preserve the relative proximity structure of the data.

Pose estimation techniques from a single view are attractive, because they potentially solve the initialization problem in model-based trackers. In fact, if they were efficient enough, they could be used for pose estimation in each input frame independently. The observation model is a key component when adopting a database-retrieval method. Ideally, the similarity function should have a global minimum at the correct pose, and it should also be smooth around this minimum to tolerate some discretization error. As in view-based recognition, segmentation is a difficult problem. The methods presented above require either a clean background [4, 5, 113, 119] or a rough segmentation [6] of the hand.

The following section gives a brief review of the related field of full human body tracking.

2.2 Full human body tracking

Both hand tracking and full human body tracking can be viewed as special cases of tracking non-rigid articulated objects. Many of the techniques for human body tracking described in the literature can be applied to hand tracking as well. Therefore this section gives a brief review of methods used for tracking people, which is still a very active research area. For more extensive reviews the reader is referred to the papers by Aggarwal and Cai [1] and Moeslund and Granum [98].

Despite the obvious similarities between the two domains, there are a number of key

differences. First of all, the observation model is usually different for hand and body tracking. Due to clothing, full body trackers have to handle a larger variability, but texture or colour of clothing provide more reliable features, which can be used for tracking [107]. Another option is to find people in images by using a face detector. In hand tracking, it is difficult to find features that can be tracked reliably due to weak texture and self-occlusion. On the other hand, skin colour is a discriminative feature, which is often used to segment the hand from the background.

Another difference is the system dynamics in both cases. In human body motion, the relative image motion of the limbs is usually much slower than for hands. Global hand motion can be very fast and also finger motion may only take a fraction of a second [134]. These factors make the dynamics of the hand much harder to model than regular motion patterns such as walking or running. However, hand tracking is usually done in the context of HCI, and feedback from higher levels can be used to interpret the motion, for example in sign language recognition systems.

As pointed out by Moeslund and Granum [98], most body motion capture systems make various assumptions related to motion and appearance. Typical assumptions are that the subject remains inside the camera view, that the range of motion is constrained, that lighting conditions are constant, or that the subject is wearing tight-fitting clothes. Most of these conditions can be made to hold in controlled environments, e.g. for motion capture applications. For robustness in less restrictive environments, such as HCI or surveillance applications, an initialization and recovery mechanism needs to be available.

Trackers based on 3D models have a relatively long tradition in human body tracking. The first model-based trackers were presented by O'Rourke and Badler [102] and Hogg [59]. The system of Hogg already contained all the elements of figure 2.1, namely a 3D geometrical model with body kinematics, an observation model, and a method to search for the 3D parameters, which optimally align the model with the image. These issues need to be addressed in every model-based tracker.

The first geometrical model, suggested by Marr and Nishihara [92], is a hierarchical structure wherein each part is represented by a cylinder. This model is used in Hogg's tracking system [59]. Other primitives have been used such as boxes or quadrics, which include cylinders, cones and ellipsoids [19, 48, 95], deformable models [79] or implicit surfaces [106]. The underlying kinematical model is arranged in a hierarchical fashion, the torso being the root frame of reference, relative to which the coordinate systems of the body limbs are defined.

A number of different observation models have been suggested to match the model

projection to the image. Examples of observations used are: edge contours [41, 48, 59, 125], foreground silhouettes [37, 106], optical flow [19, 125], dense stereo depth maps [54, 106], or multi-view voxel data [95]. Several works combine a number of these features into one cost term.

In some cases, a model with weak or no dynamics is used, and the model parameters are optimized at each frame, such that they minimize a cost function based on observations [37, 59, 106, 125]. When a dynamic model is used, the motion in consecutive frames can be predicted and integrated with the observation within a filtering framework. Possible choices are extended Kalman filtering [79, 95], or versions of particle filtering [38]. The human body dynamics have been modelled with constant velocity or constant acceleration models [48, 79], switched linear dynamic models [104] or example-based models [122].

The majority of model-based 3D body trackers require multiple views in order to reduce ambiguity. The high dimensionality of the state space, together with self-occlusions and depth ambiguity make tracking in a single view an ill-posed problem [100]. The cost function in the monocular estimation problem is highly multi-modal and there exist many possible inverse kinematic solutions for one observation. Choosing the wrong minimum leads to loss of track, therefore reliable tracking requires the maintenance of multiple hypotheses. Sminchisescu and Triggs [125] address this problem by using a cost function, which includes multiple cues, such as edge intensity and optical flow, and adopting a sophisticated sampling and search strategy in the 30 dimensional state space. Another, complementary strategy to resolve this ambiguity is to learn dynamic priors from training data of motions [18, 60, 122].

For simple HCI applications blob trackers may be adequate, for example the *Pfinder* system [148]. Typically a person is modelled by a number of ellipsoids, which are matched from frame to frame using an appearance descriptor, such as a colour histogram. Isard and MacCormick [69] combined blob tracking with particle filtering in their *BraMBLe* system. The body shape was modelled as a single generalized cylinder, and the system could track the position of multiple people in a surveillance application, where a static camera was assumed.

There have been only few papers on 3D body pose estimation from a single image. One example is the method of Mori and Malik [99] who used a *shape context* descriptor [11] to match the input image to a number of exemplar 2D views. The exemplars were labelled with key points, and after registering the input shape, these points were used to reconstruct a likely 3D pose. This estimation algorithm was applied to each frame in a

video sequence independently.

Another method was presented by Shakhnarovich *et al.* [117] who generated a large number of example images using 3D animation software. The technique of *parameter sensitive hashing* was introduced to learn a hash function in order to efficiently retrieve approximate nearest neighbours with respect to both feature and parameter values. This approach is very similar to the method of Athitsos and Sclaroff [4] for hand pose estimation, where the problem was solved by a fast nearest neighbour search in the appearance domain.

In a parts-based approach to people detection, such as the one by Felzenszwalb and Huttenlocher [44], "bottom-up" and "top-down" methods are combined. The body is represented by a collection of body parts, e.g. torso, head and limbs, and each of these is matched individually. The body pose is found by optimizing a cost function which takes the global configuration into account. These methods are promising as they reduce the search complexity significantly and they have recently been extended to three-dimensional body tracking from multiple views [123].

2.3 Object detection

One of the main challenges in tracking is initialization and recovery. This is necessary in HCI applications, where the user can move in and out of the camera view, and manual re-initialization is not a desirable option. Hand or full body tracking systems that rely on incremental estimation and do not have a recovery mechanism have been demonstrated to track accurately, but not over very long image sequences.

Using a filter that supports multiple hypotheses is necessary to overcome ambiguous frames so that the tracker is able to recover. Another way to overcome the problem of losing lock is to treat tracking as object detection at each frame. Thus if the target is lost at one frame, subsequent frames are not affected. For example, current face detection systems work reliably and fast [141] without relying on dynamic models. However, detectors tend to give multiple positive responses in the neighbourhood of a correct match, and temporal information is useful to render such cases unambiguous.

The following subsections review template matching techniques, which have been introduced in the context of object detection. Some of these methods have become so efficient that they can be implemented in real time systems, e.g. automated pedestrian detection in vehicles [47].

2.3.1 Matching shape templates

Template-based methods have yielded good results for locating deformable objects in a scene with no prior knowledge, e.g. for hands or pedestrians [6, 47]. These methods are made robust and efficient by the use of distance transforms such as the chamfer or Hausdorff distance between template and image [9, 16, 62, 64], and were originally developed for matching a single template.

Chamfer matching for a single shape was introduced by Barrow *et al.* [9]. It is a technique for finding the best match of two sets of edge points by minimizing a generalized distance between them. A common application is template-to-image matching, where the aim is to locate a previously defined object in the image. In most practical applications, the shape template is allowed to undergo a certain transformation, which needs to be dealt with efficiently. No explicit point correspondences are determined in chamfer matching, which means that the whole parameter space needs to be searched for an optimum match. In order to do this efficiently, the smoothness property of the chamfer distance can be exploited. If the change in the transformation parameters is small, the chamfer distance change is also small. This motivated the multi-resolution approaches of Borgefors for chamfer matching [16] and Huttenlocher and Rucklidge for Hausdorff matching [65]. Borgefors introduced the hierarchical chamfer matching algorithm [16], where a multi-resolution pyramid of edge images is built and matching proceeds in a coarse-to-fine manner. At the coarsest level, the transformation parameters are optimized, starting from points, which are located on a regular grid in parameter space. From these optima the matches at the next level of resolution are computed, and locations where the cost increases beyond a threshold are rejected and not further explored. Shape matching is demonstrated for Euclidean and affine transformations. However, a careful adjustment of step lengths for optimization and rejection thresholds is required and the algorithm only guarantees a locally optimal match.

Huttenlocher and Rucklidge [65] used a similar approach to search the 4D transformation space of translation and independent scaling of the x– and y–axis. The parameter space is tesselated at multiple resolutions and the centre of each region is evaluated. Bounds on the maximum change in cost are derived for the transformation parameters and regions in parameter space are eliminated in a coarse-to-fine search. Within this particular 4D transformation space the method is guaranteed not to miss any match, however, it is not straightforward to extend to other transformations, such as rotations.

Both methods were developed for matching a single shape template. Three-dimensional objects can be recognized by representing each object by a set of 2D views. This multi-

view based representation has been used frequently in object recognition [83]. These views, together with a similarity transformation can then be used to approximate the full 6D transformation space. In the case when these shapes are obtained from training examples, they are often referred to as *exemplars*.

Breuel [21, 22] proposed an alternative shape matching algorithm by subdividing the transformation space. The quality of a match assigned to a transformation is the number of template points that are brought within an error bound of some image point. For each sub-region of parameter space an upper bound on this quality of match is computed, and the global optimum is found by recursive subdivision of these regions.

A key suggestion was that multiple templates could be dealt with efficiently by using a tree structure [47, 101]. Olson and Huttenlocher [101] used agglomerative clustering of shape templates. In each iteration the two closest models in terms of chamfer distance are grouped together. Each node only contains those points, which the templates in the subtree have in common. During evaluation the tree is searched from the root and at each node model points, which are too far away from an image edge are counted. This count is propagated to the children nodes and the search is stopped once a threshold value is reached. The threshold value corresponds to an upper bound on the partial Hausdorff distance between templates. This allows for early rejection of models, however, the gain in speed is only significant if the models have a large number of points in common. For this to be likely, a large number of model views is necessary and it is not clear which and how many views to use.

Gavrila [47, 49] suggested forming the template hierarchy by optimizing a clustering cost function. His idea is to group similar templates and represent them with a single prototype template together with an estimate of the variance of the error within the cluster, which is used to define a matching threshold. The prototype is first compared to the image and only if the error is below the threshold are the templates within the cluster compared to the image. This clustering is done at various levels, resulting in a hierarchy, in which the templates at the leaf level cover the space of all possible templates. The method is used in the context of pedestrian detection from a moving vehicle. A three-level tree is built from approximately 4,500 templates, which includes a search over five scales for each template. Detection rates of 60–90% are reported by using the chamfer distance alone. An extension of the system includes a further verification step based on a nonlinear classifier [47].

An approach to embed exemplar-based matching in a probabilistic tracking framework was proposed by Toyama and Blake [135]. In this method, templates are clustered

using a k-medioid clustering algorithm. A likelihood function is estimated as a Gaussian mixture model, where each cluster prototype defines the centre of a mode. The dynamics are encoded as transition probabilities between prototype templates, and a first order dynamic model is used for the continuous transformation parameters. The observation and dynamic models are integrated in a Bayesian framework and the posterior distribution is estimated with a particle filter. A problem with template sets is that their size can grow exponentially with object complexity.

2.3.2 Hierarchical classification

This section presents another line of work in the area of object detection, which has been shown to be succesful for face detection. The general idea is to slide windows of different sizes over the image and classify each window as face or non-face. Alternatively, by scaling the image, a window of the same size, typically of size 20×20 pixels, can be used.

The task thus is formulated as a pattern recognition problem, and the methods differ by which features and which type of classifier they use. Most of the papers deal with the case of single view detection. Multiple views can be handled by separately training detectors for images taken from other views. Variation in lighting and contrast is either handled by a normalization step during pre-processing, or by explicitly including a large number of training examples in different lighting conditions.

Rowley, Baluja and Kanade [114] used a neural network-based classifier. After pre-processing, the sub-window were fed into a multi-layer neural network, which returns a real value between $+1$, corresponding to a face, and -1, corresponding to no face. The outputs of different filters were then combined in order to eliminate overlapping detections.

Schneiderman and Kanade [115] learnt the appearance of faces and of non-face patches in frontal and side-view by representing them as histograms of wavelet coefficients. The detection decision for each window was based on the maximum likelihood ratio.

Romdhani *et al.* [112] introduced a cascade of classifiers where regions which are unlikely to be faces were discarded early, and more computational resources was spent in the remaining regions (see figure 2.4). This was done by sequential evaluation of support vectors in an SVM classifier.

Viola and Jones [141] introduced rectangle features, which are differences of sums of intensities over rectangular regions in the image patch. These can be computed very efficiently by using *integral images* [33, 141]. A cascade of classifiers was constructed,

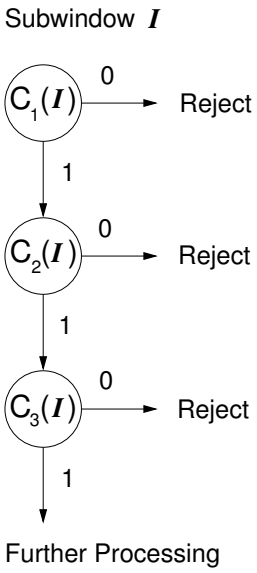


Figure 2.4: **Cascade of classifiers.** *A cascade of classifiers C_1, C_2, and C_3, with increasing complexity. Each classifier has a high detection rate and a moderate false positive rate. Subwindows I which are unlikely to contain an object are rejected early, and more computational effort is spent on regions which are more ambiguous.*

arranged in successively more discriminative, but also computationally more expensive, order. Feature selection and training was done using a version of the AdaBoost algorithm [46].

Li *et al.* [86] extended the approach of Viola and Jones to multi-view detection. The range of head rotation was partitioned successively into smaller sub-ranges, discriminating between seven distinct head rotations. An improved boosting technique was used in order to reduce the number of necessary features.

It is difficult to compare the performance of the methods presented, as in some cases different testing data was used, in other cases only the detection rate at a fixed false detection rate was reported. The detection rates of all systems are fairly comparable, however, the best performance has been reported for the Schneiderman-Kanade detector. The Viola-Jones detector is the fastest system for single-view detection, running in real-time.

The question arises whether these methods could be applied for detecting or tracking a hand. An important difference between hands and faces is that hand images can have a much larger variation in appearance due to the large number of different poses. It would be surprising to find that only a few simple features could be used to discriminate hands from other image regions. Thus it is necessary to search over different poses as well as over views. This also means that a very large training set is necessary; Viola and

Jones used nearly 5,000 face images for single-view detection [141]. There is also a large variation in hand shape, and in most poses, a search window containing the hand will include a larger number of background pixels. It may be necessary to use more discriminative features than rectangle filters based on intensity differences. For example, Viola *et al.* [142] included simple motion features to apply their framework to pedestrian detection. If shape is to be used as a feature, then this framework becomes very similar to the hierarchical shape matching methods described in section 2.3.1.

2.4 Summary

This chapter has introduced a number of different approaches to detecting, tracking and interpreting hand motion from images. Accurate 3D tracking of hand or full body requires a geometric model, as well as a scheme to update the model given the image data. A range of hand tracking algorithms have been surveyed, using single or multiple views. The problem is ill-posed when using a single camera, and a number of methods have been suggested to reduce the number of parameters that need to be estimated. A shortcoming of these methods is the lack of an initialization and recovery mechanism. View-based methods could be used to solve this problem. However, view-based gesture recognition systems generally assume that the segmentation problem has already been solved, and most of them discriminate between a few gestures only. Single-view pose estimation attempts to recover the full 3D hand pose from a single image. This is done by matching the image to a large database of model-generated hand shapes. However, the mapping from a single 2D view to 3D pose is inherently ambiguous, and methods that are based on detection alone ignore temporal information, which could resolve ambiguous situations.

In order to be able to accurately recover 3D information about the hand pose a 3D geometric hand model is essential, and the following chapter presents a method to construct such a model using truncated quadrics as building blocks.

3 A 3D Hand Model Built From Quadrics

Treat nature by means of the cylinder, the sphere, the cone, everything brought into proper perspective.

Paul Cézanne (1839–1906)

In this work a model-based approach is pursued to recover the three-dimensional hand pose. The model is used to encode prior knowledge about the hand geometry and valid hand motion. It also allows the pose recovery problem to be formulated as a parameter estimation task and is therefore central to this work. In chapter 4 the model is used to generate contours for tracking with an *unscented Kalman filter*, and in chapters 5 and 6 it is used to generate templates, which are used for shape-based matching.

This chapter explains how to construct a three-dimensional hand model using truncated quadrics as building blocks, approximating the anatomy of a real human hand. This approach uses results from projective geometry in order to generate the model projection as a set of conics, while handling cases of self-occlusion. As a theoretical background, a brief introduction to the projective geometry of quadrics is given in the next section. For textbooks on this topic, the reader is referred to [28, 56, 116].

3.1 Projective geometry of quadrics and conics

A quadric Q is a second degree implicit surface in 3D space. It can be represented in homogeneous coordinates as a symmetric 4×4 matrix $\mathbf{Q}$. The surface is defined by the points $\mathbf{X}$, which satisfy the equation

$$\mathbf{X}^{\mathrm{T}}\mathbf{Q}\,\mathbf{X} \;=\; 0\,. \tag{3.1}$$

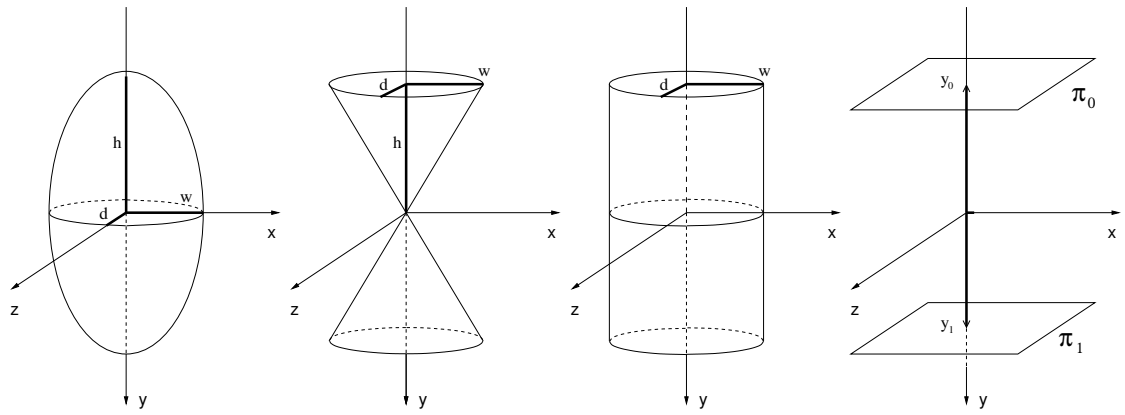


Figure 3.1: **Examples of quadratic surfaces.** *This figure shows different surfaces corresponding to the equations in the text: (a) ellipsoid, (b) cone, (c) elliptic cylinder, (d) pair of planes.*

where $\mathbf{X} = [x, y, z, 1]^{\mathrm{T}}$ is a homogeneous vector representing a 3D point. A quadric has 9 degrees of freedom. Equation (3.1) is simply the quadratic equation:

$$q_{1,1}\,x^2 + q_{2,2}\,y^2 + q_{3,3}\,z^2 + 2q_{1,2}\,xy + 2q_{1,3}\,xz + 2q_{2,3}\,yz + 2q_{1,4}\,x + 2q_{2,4}\,y + 2q_{3,4}\,z + q_{4,4} = 0 \tag{3.2}$$

Different families of quadrics are obtained from matrices $\mathbf{Q}$ of different ranks. Figure 3.1 shows examples of their geometric interpretation. These particular cases of interest are:

- **Ellipsoids**, represented by matrices $\mathbf{Q}$ with full rank. The normal implicit form of an ellipsoid centred at the origin and aligned with the coordinate axes is given by

$$\frac{x^2}{w^2} + \frac{y^2}{h^2} + \frac{z^2}{d^2} = 1 \ , \tag{3.3}$$

  and the corresponding matrix is given by

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{w^2} & 0 & 0 & 0 \\ 0 & \frac{1}{h^2} & 0 & 0 \\ 0 & 0 & \frac{1}{d^2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \ . \tag{3.4}$$

  Other examples of surfaces which can be represented by matrices with full rank are paraboloids and hyperboloids. If the matrix $\mathbf{Q}$ is singular, the quadric is said to be *degenerate*.

- **Cones and cylinders**, represented by matrices $\mathbf{Q}$ with $\mathrm{rank}(\mathbf{Q}) = 3$.

  The equation of a cone aligned with the y-axis is given by

$$\frac{x^2}{w^2} - \frac{y^2}{h^2} + \frac{z^2}{d^2} = 0 \,, \tag{3.5}$$

  and the corresponding matrix is given by

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{w^2} & 0 & 0 & 0 \\ 0 & -\frac{1}{h^2} & 0 & 0 \\ 0 & 0 & \frac{1}{d^2} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} . \tag{3.6}$$

  An elliptic cylinder, aligned with the y-axis, is described by

$$\frac{x^2}{w^2} + \frac{z^2}{d^2} = 1 \,, \tag{3.7}$$

  and the corresponding matrix is given by

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{w^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d^2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} . \tag{3.8}$$

- **Pairs of planes $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}_1$**, represented as $\mathbf{Q} = \boldsymbol{\pi}_0\boldsymbol{\pi}_1^{\mathrm{T}} + \boldsymbol{\pi}_1\boldsymbol{\pi}_0^{\mathrm{T}}$ with $\mathrm{rank}(\mathbf{Q}) = 2$. For example, a pair of planes, which are parallel to the xz-plane can be described by the equation:

$$(y - y_0)\,(y - y_1) = 0 \,, \tag{3.9}$$

  in which case $\boldsymbol{\pi_0} = [0, 1, 0, -y_0]^{\mathrm{T}}$ and $\boldsymbol{\pi_1} = [0, 1, 0, -y_1]^{\mathrm{T}}$.

  The corresponding matrix is given by

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\frac{(y_0+y_1)}{2} \\ 0 & 0 & 0 & 0 \\ 0 & -\frac{(y_0+y_1)}{2} & 0 & y_0\, y_1 \end{bmatrix} . \tag{3.10}$$

Coordinate transformations Under a Euclidean transformation $\mathbf{T}$ the shape of a quadric is preserved, but the matrix representation changes. Given a point $\mathbf{X}$ on quadric $\mathbf{Q}$, the equation of quadric $\widehat{\mathbf{Q}}$, on which the transformed point $\widehat{\mathbf{X}} = \mathbf{T}\mathbf{X}$ lies can be

derived as follows:

$$\mathbf{X}^{\mathrm{T}}\mathbf{Q}\mathbf{X} = 0 \tag{3.11}$$

$$\Leftrightarrow \quad \mathbf{X}^{\mathrm{T}}\left(\mathbf{T}^{\mathrm{T}}\mathbf{T}^{-\mathrm{T}}\right)\mathbf{Q}\left(\mathbf{T}^{-1}\mathbf{T}\right)\mathbf{X} = 0 \tag{3.12}$$

$$\Leftrightarrow \quad (\mathbf{T}\mathbf{X})^{\mathrm{T}}\left(\mathbf{T}^{-\mathrm{T}}\mathbf{Q}\mathbf{T}^{-1}\right)(\mathbf{T}\mathbf{X}) = 0 \tag{3.13}$$

$$\Leftrightarrow \quad \widehat{\mathbf{X}}^{\mathrm{T}}\widehat{\mathbf{Q}}\widehat{\mathbf{X}} = 0\,. \tag{3.14}$$

Thus the matrix $\mathbf{Q}$ is mapped to a matrix $\widehat{\mathbf{Q}} = \mathbf{T}^{-\mathrm{T}}\mathbf{Q}\mathbf{T}^{-1}$. The matrix $\mathbf{T}$ represents a Euclidean transformation, and $\mathbf{T}$ and $\mathbf{T}^{-1}$ can be written in homogeneous coordinates as

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R}^{\mathrm{T}} & -\mathbf{R}^{\mathrm{T}}\mathbf{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}\,. \tag{3.15}$$

Equivalently, this implies that any quadric matrix $\widehat{\mathbf{Q}}$, representing an ellipsoid, cone or cylinder can be factored into components representing its shape, given in normal implicit form as in section 3.1, and motion in form of a Euclidean transformation.

Truncated quadrics By truncating quadrics, they can be used as building blocks for modelling more complex shapes [109, 130]. For any quadric $\mathbf{Q}$ the truncated quadric $\mathbf{Q}_{\mathbf{\Pi}}$ can be obtained by finding points $\mathbf{X}$ satisfying:

$$\mathbf{X}^{\mathrm{T}}\mathbf{Q}\mathbf{X} = 0 \tag{3.16}$$

$$\text{and} \qquad \mathbf{X}^{\mathrm{T}}\mathbf{\Pi}\mathbf{X} \geq 0\,, \tag{3.17}$$

where $\mathbf{\Pi}$ is a matrix representing a pair of clipping planes, see figure 3.2. It can be ensured that $\mathbf{X}^{\mathrm{T}}\,\mathbf{\Pi}\,\mathbf{X} \geq 0$ for points between the clipping planes, by evaluating this equation for a point at infinity in the direction of one of the plane normal vectors, and if it is positive, the sign of $\mathbf{\Pi}$ is changed.

Projecting a quadric In order to determine the projected outline of a quadric for a normalized projective camera $\widetilde{\mathbf{P}} = [\,\mathbf{I} \mid \mathbf{0}\,]$, the quadric matrix $\mathbf{Q}$ is written as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^{\mathrm{T}} & c \end{bmatrix}. \tag{3.18}$$

The camera centre $\mathbf{0}$ and a point $\mathbf{x}$ in image coordinates define a ray $\mathbf{X}(\zeta) = [\mathbf{x}, \zeta]^{\mathrm{T}}$ in 3D space. The inverse depth of a point on the ray is determined by the free parameter ζ , such that the point $\mathbf{X}(0)$ is at infinity and $\mathbf{X}(\infty)$ is at the camera centre (see figure 3.3). The point of intersection of the ray with the quadric $\mathbf{Q}$ can be found by solving the equation

$$\mathbf{X}(\zeta)^{\mathrm{T}}\,\mathbf{Q}\,\mathbf{X}(\zeta) = 0\,, \tag{3.19}$$

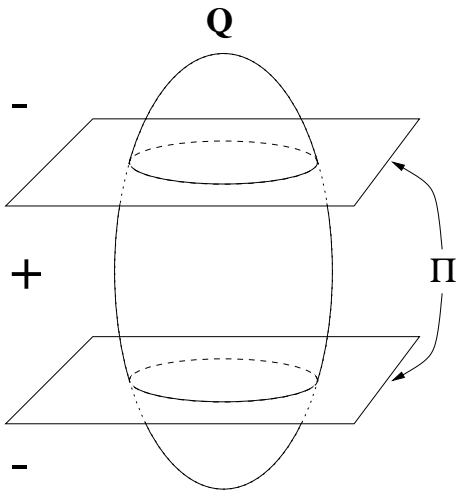


Figure 3.2: **A truncated ellipsoid.** *The truncated quadric* $\mathbf{Q}_\Pi$*, e.g. a truncated ellipsoid, can be obtained by finding points on the quadric* $\mathbf{Q}$ *which satisfy* $\mathbf{X}^{\mathrm{T}}\mathbf{\Pi}\mathbf{X} \geq 0$*.*

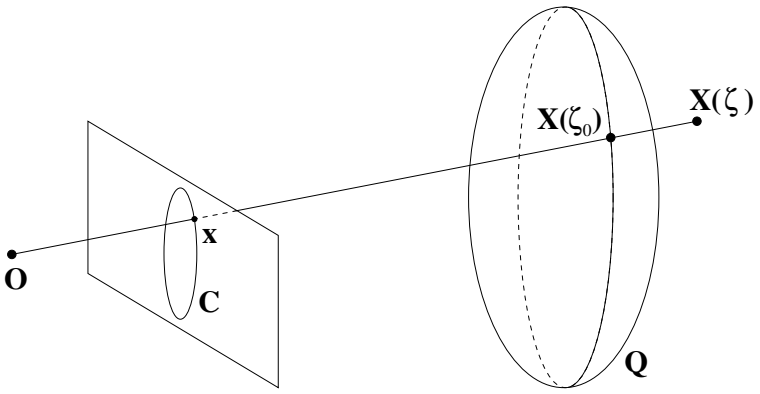


Figure 3.3: **Projecting a quadric.** *When viewed through a normalized camera, the projection of a quadric* $\mathbf{Q}$ *onto the image plane is a conic* $\mathbf{C}$*. The ray* $\mathbf{X}(\zeta)$ *defined by the origin and a point* $\mathbf{x}$ *on the conic is parameterized by the inverse depth parameter* ζ*, and intersects the contour generator at* $\mathbf{X}(\zeta_0)$*.*

which can be written as a quadratic equation in ζ [28]:

$$c\,\zeta^2 \;+\; 2\mathbf{b}^{\mathrm{T}}\mathbf{x}\,\zeta \;+\; \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} \;=\; 0\,. \tag{3.20}$$

When the ray represented by $\mathbf{X}(\zeta)$ is tangent to $\mathbf{Q}$, equation (3.20) will have a unique real solution ζ_0. Algebraically, this condition is satisfied when the discriminant is zero, i.e.

$$\mathbf{x}^{\mathrm{T}}\left(\mathbf{b}\mathbf{b}^{\mathrm{T}} - c\mathbf{A}\right)\mathbf{x} = 0\,. \tag{3.21}$$

Therefore, the contour of quadric $\mathbf{Q}$ is given by all points $\mathbf{x}$ in the image plane satisfying $\mathbf{x}^{\mathrm{T}}\mathbf{C}\mathbf{x} = 0$, where

$$\mathbf{C} = c\mathbf{A} - \mathbf{b}\mathbf{b}^{\mathrm{T}}\,. \tag{3.22}$$

The 3×3 matrix $\mathbf{C}$ represents a conic, which is an implicit curve in 2D space. For points $\mathbf{x}$ on the conic $\mathbf{C}$, the unique solution of equation (3.20) is

$$\zeta_0 = -\frac{\mathbf{b}^{\mathrm{T}}\mathbf{x}}{c} , \tag{3.23}$$

which is the inverse depth of the point on the contour generator of quadric $\mathbf{Q}$.

The normal vector to a conic The normal vector $\mathbf{n}$ to a conic $\mathbf{C}$ at a point $\mathbf{x}$ can be computed in a straightforward manner. In homogeneous coordinates a line is represented by a 3–vector $\mathbf{g}$, and is defined as the set of points $\mathbf{x}$ satisfying

$$\mathbf{g}^{\mathrm{T}}\mathbf{x} = 0 . \tag{3.24}$$

By writing $\mathbf{g} = [\mathbf{n}, -d]^{\mathrm{T}}$, the line is defined in terms of its normal vector $\mathbf{n}$, and d, the distance to the origin.

The line $\mathbf{l}$ which is tangential to the conic at the point $\mathbf{x} \in \mathbf{C}$ is given by

$$\mathbf{l} = \mathbf{C}\mathbf{x} , \tag{3.25}$$

for a proof see [56]. For $\mathbf{l} = [l_1, l_2, l_3]^{\mathrm{T}}$ its normal in Cartesian coordinates is given as $\mathbf{n} = [l_1, l_2]^{\mathrm{T}}$ and is normalized to length one.

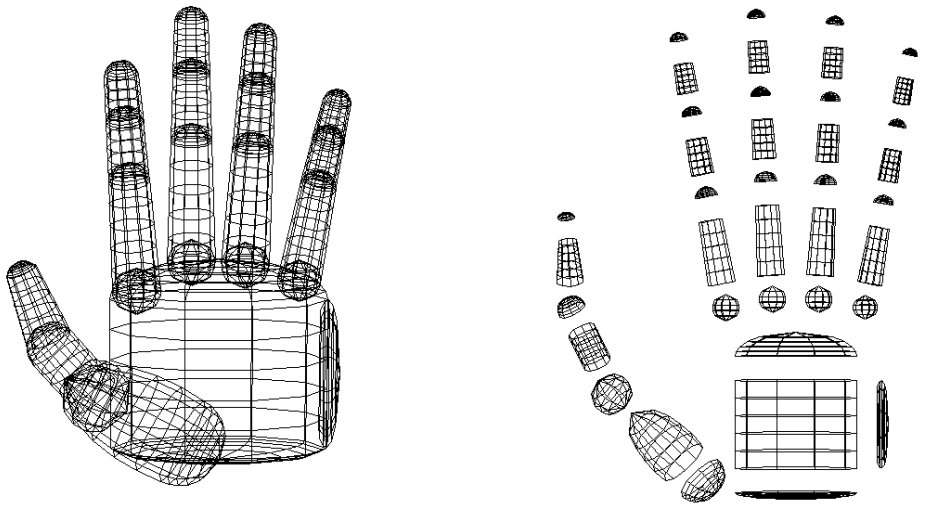
Projection into multiple cameras A world point $\mathbf{X}$ is mapped to an image point $\mathbf{x}$ by the 3×4 homogeneous camera projection matrix $\mathbf{P}$:

$$\mathbf{x} = \mathbf{P}\,\mathbf{X} = \mathbf{K}\,[\,\mathbf{I} \mid \mathbf{0}\,]\,\mathbf{X} , \tag{3.26}$$

where $\mathbf{K}$ is the camera calibration matrix and the camera is located at the origin of the world coordinate system, pointing in the direction of the positive z-axis. If there are multiple cameras, the first camera can be used to define a reference coordinate frame. All other camera positions and orientations are then given relative to this view, and can be transformed into the normalized camera view by a Euclidean transformation $\mathbf{H}$. For example, assume that $\mathbf{P}_1 = \mathbf{K}\,[\,\mathbf{I} \mid \mathbf{0}\,]$ is the normalized projective camera, and a second camera is defined as $\mathbf{P}_2 = \mathbf{P}_1\mathbf{H}$. Then projecting the world point $\mathbf{X}$ with camera $\mathbf{P}_2$ is the same as projecting the point $\mathbf{H}\mathbf{X}$ in the normalized view:

$$\mathbf{x} \;=\; \mathbf{P}_2\mathbf{X} \;=\; \mathbf{P}_1\mathbf{H}\,\mathbf{X} . \tag{3.27}$$

As seen previously, coordinate transformations for quadrics can be handled easily. It may be the case that a point is not visible in all views, thus occlusion handling must be done for each view separately.



(a) (b)

Figure 3.4: **The geometric hand model.** The 27 DOF hand model is constructed from 39 truncated quadrics. (a) Front view and (b) exploded view are shown.

3.2 Constructing a hand model

The hand model is built using a set of quadrics $\{\mathbf{Q}_i\}_{i=1}^{N_Q}$, approximately representing the anatomy of a real human hand, as shown in figure 3.4. A hierarchical model with 27 degrees of freedom (DOF) is used: 6 for the global hand position, 4 for the pose of each finger and 5 for the pose of the thumb. The DOF for each joint correspond to the DOF of a real hand. Starting from the palm and ending at the tips, the coordinate system of each quadric is defined relative to the previous one in the hierarchy.

The palm is modelled using a truncated cylinder, its top and bottom closed by half-ellipsoids. Each finger consists of three cone segments, one for each phalanx. They are connected by truncated spheres, representing the joints. Hemispheres are used for the tips of fingers and thumb. The shape parameters of each quadric are set by taking measurements from a real hand.

Joining quadrics – an example Figure 3.5 illustrates how truncated quadrics can be joined together in order to design parts of a hand, such as a finger. There may be several ways in which to define the model shape. However, it is desirable to make all variables dependent on a small number of values that can be easily obtained, such as the

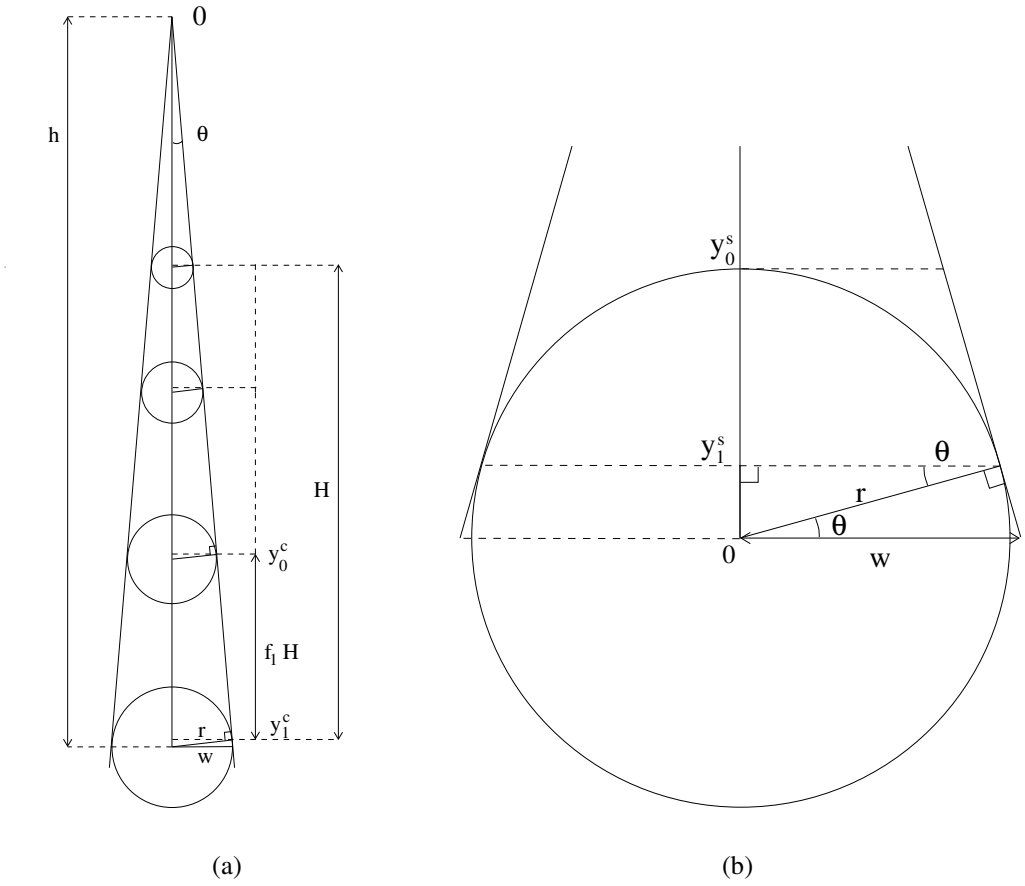


Figure 3.5: **Constructing a finger model from quadrics.** This figure shows "finger blueprints" in a side view. Cones are used for finger segments (a) and the joints (b) are modelled with spheres. Figure (b) is a close-up view of the bottom joint in (a).

height, width and depth of a finger. These values are then used to set the parameters of the quadrics. Each finger is built from cone segments and truncated spheres. The width parameter w of the cone in figure 3.5 (a) is uniquely determined by the height of the segment h and the radius r of the sphere, which models the bottom joint:

$$w = \frac{h\,r}{\sqrt{h^2 - r^2}}\ . \tag{3.28}$$

In order to create a continuous surface, the parameter y_1^s of the bottom clipping plane (see figure 3.5 b) for the sphere is found as

$$y_1^s = \frac{r^2}{h}\ , \tag{3.29}$$

and from this the clipping parameters for the cone are found as

$$y_1^c = h - y_1^s \quad \text{and} \quad y_0^c = y_1^c - f_1 H\ , \tag{3.30}$$

where H is the sum of the length of all three cone segments, and f_1 is the ration between the length of the bottom segment to the total length.

The projection of an ellipsoid is an ellipse, and the projection of a cone or cylinder is a pair of lines. The following section describes how to obtain the equations for these projections from a general conic matrix $\mathbf{C}$ in order to graph them in the image plane. The reader familiar with these details may skip this section and continue with section 3.4.

3.3 Drawing the contours

The idea when plotting a general conic is to factorize the conic matrix $\mathbf{C}$ in order to obtain its shape in terms of a diagonal matrix $\mathbf{D}$ and a transformation matrix $\mathbf{V}$. The matrix $\mathbf{C}$ is real and symmetric and therefore can be factorized using eigendecomposition:

$$\mathbf{C} = \mathbf{V}\ \mathbf{D}\ \mathbf{V}^{\mathrm{T}}\ , \tag{3.31}$$

where $\mathbf{V}$ is an orthogonal matrix, i.e. $\mathbf{V}\mathbf{V}^{\mathrm{T}} = \mathbf{I}$ and $\mathbf{D}$ is a diagonal matrix containing the eigenvalues of $\mathbf{C}$. If a point $\mathbf{x}$ lies on the conic represented by $\mathbf{D}$, the transformed point $\mathbf{V}\mathbf{x}$ lies on conic $\mathbf{C}$:

$$\mathbf{x}^{\mathrm{T}}\mathbf{D}\mathbf{x} = 0 \tag{3.32}$$

$$\Leftrightarrow \quad \mathbf{x}^{\mathrm{T}}(\mathbf{V}^{\mathrm{T}}\mathbf{V})\mathbf{D}(\mathbf{V}^{\mathrm{T}}\mathbf{V})\mathbf{x} = 0 \tag{3.33}$$

$$\Leftrightarrow \quad (\mathbf{V}\mathbf{x})^{\mathrm{T}}\mathbf{C}(\mathbf{V}\mathbf{x}) = 0\ . \tag{3.34}$$

The matrix $\mathbf{D}$ represents the conic in its normalized form, i.e. centred at the origin and aligned with the coordinate axes. A general conic $\mathbf{C}$ can therefore be plotted by finding a parameterization for the normalized conic and then map the points to $\mathbf{V}\mathbf{x}$.

- **Drawing ellipsoids**

  A normalized ellipse is plotted by using the parameterization $\mathbf{x}(t) = [a\cos t,\ b\ \sin t,\ 1]^{\mathrm{T}}$. From the equation $\mathbf{x}^{\mathrm{T}}\mathbf{D}\mathbf{x} = 0$ the radii a and b can be found from the elements in $\mathbf{D}$ as

  $$a = \sqrt{-\frac{d_{33}}{d_{11}}} \quad \text{and} \quad b = \sqrt{-\frac{d_{33}}{d_{22}}}\ . \tag{3.35}$$

  In order to check whether the ellipse points are between the clipping planes, one can plot only the points $\mathbf{X}$ on the contour generator for which $\mathbf{X}^{\mathrm{T}}\mathbf{\Pi}\mathbf{X} \geq 0$ holds. However, it is more efficient to explicitly find the end points of the arches to be drawn. For this the contour generator is intersected with the clipping planes $\boldsymbol{\pi}_0$ and

$\boldsymbol{\pi}_1$, and these points projected into the image. In order to compute the intersection, the following parameterization of the contour generator on $\mathbf{Q}$ is used

$$\mathbf{X}(t) \quad = \quad \begin{bmatrix} \mathbf{x}(t), \\ \zeta(t) \end{bmatrix} , \tag{3.36}$$

where $\zeta(t)$ is obtained from equation (3.23) as $\zeta(t) = \frac{-\mathbf{b}^{\mathrm{T}}\mathbf{x}(t)}{c}$. One of the clipping planes $\boldsymbol{\pi}$ is given by

$$\boldsymbol{\pi} = \begin{bmatrix} \mathbf{n}_\pi \\ -d_\pi \end{bmatrix} . \tag{3.37}$$

The points of intersection of the contour generator and this clipping plane are then given by the solutions of the equation

$$\mathbf{X}^{\mathrm{T}}(t) \quad \boldsymbol{\pi} \quad = \quad 0 \tag{3.38}$$

$$\Leftrightarrow \quad \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(t) & \frac{-\mathbf{x}^{\mathrm{T}}(t)\,\mathbf{b}}{c} \end{bmatrix} \begin{bmatrix} \mathbf{n}_\pi \\ -d_\pi \end{bmatrix} \quad = \quad 0 \tag{3.39}$$

$$\Leftrightarrow \quad \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(t) & \mathbf{k} \end{bmatrix} \quad = \quad 0 \tag{3.40}$$

$$\Leftrightarrow \quad a\,k_1\,\cos t \;+\; b\,k_2\,\sin t \;+\; k_3 \quad = \quad 0\,, \tag{3.41}$$

where the 3-vector $\mathbf{k}$ is defined as

$$\mathbf{k} = \mathbf{n}_\pi + \frac{\mathbf{b}\,d_\pi}{c} . \tag{3.42}$$

The solutions to (3.41) can be found using the sine addition formula.

- **Drawing cones and cylinders**

  The projection of cones and cylinders is a pair of lines $\mathbf{l}_1$ and $\mathbf{l}_2$. Once they have been found, they can be used to obtain a parameterization of the contour generator on $\mathbf{Q}$, see figure 3.6. This contour generator is then intersected with the clipping planes $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}_1$ to find the end points of each line.

  The line equations are obtained from the matrix $\mathbf{D}$. In the case of cones and cylinders, the matrix $\mathbf{D}$ has $\mathrm{rank}\,2$ and by column and row permutations it can be brought into the form

$$\mathbf{D} = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & -d_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} , \tag{3.43}$$

  where d_1 and d_2 are positive. The equation $\mathbf{x}^{\mathrm{T}}\mathbf{D}\mathbf{x} = 0$ can also be written as

$$(\sqrt{|d_1|}\,x_1 + \sqrt{|d_2|}\,x_2)\;\;(\sqrt{|d_1|}\,x_1 - \sqrt{|d_2|}\,x_2) = 0\,, \tag{3.44}$$

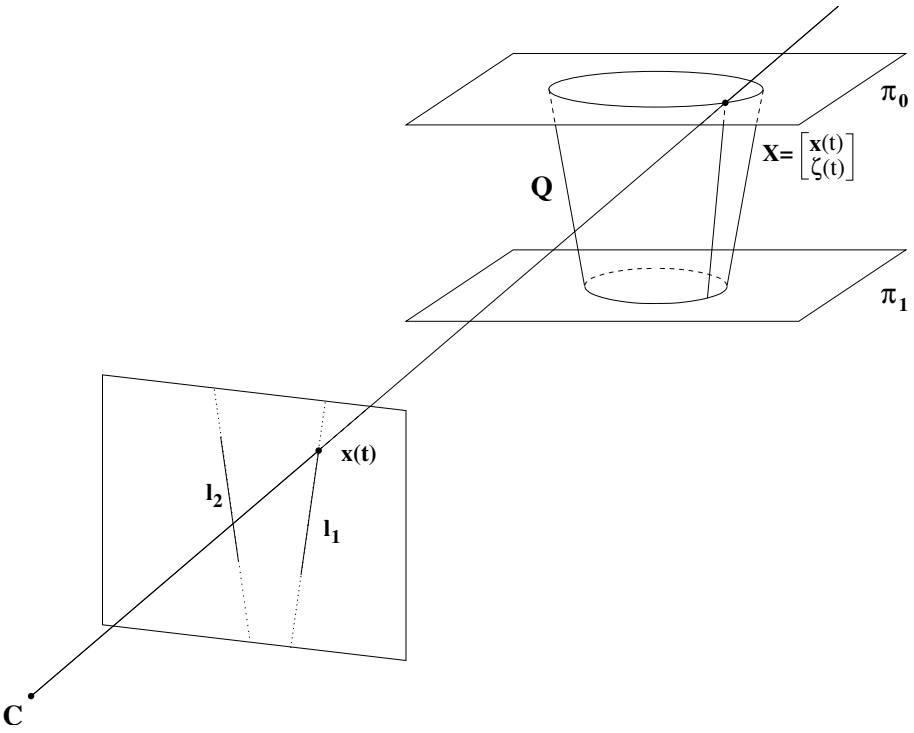


Figure 3.6: **Projecting cones and cylinders.** The projection of a cone (shown here) or cylinder is a pair of lines $\mathbf{l}_1$ and $\mathbf{l}_2$. In order to plot them, their end points are found as the projections of the intersecting points of the contour generators on the quadric $\mathbf{Q}$ and the clipping planes $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$.

which corresponds to the two lines $\widehat{\mathbf{l}}_1$ and $\widehat{\mathbf{l}}_2$, represented in homogeneous coordinates as

$$\widehat{\mathbf{l}}_1 = [\sqrt{|d_1|},\ \sqrt{|d_2|},\ 0]^{\mathrm{T}} \quad \text{and} \quad \widehat{\mathbf{l}}_2 = [\sqrt{|d_1|},\ -\sqrt{|d_2|},\ 0]^{\mathrm{T}}\ . \tag{3.45}$$

The two contour lines corresponding to $\mathbf{C}$ are then $\mathbf{l}_1 = \mathbf{V}\ \widehat{\mathbf{l}}_1$ and $\mathbf{l}_2 = \mathbf{V}\ \widehat{\mathbf{l}}_2$. It can be verified that if $\mathbf{x}$ is on line $\widehat{\mathbf{l}}_1$, then the transformed point $\mathbf{V}\mathbf{x}$ lies on $\mathbf{l}_1$ (analogously for line $\mathbf{l}_2$):

$$\widehat{\mathbf{l}}_1^{\mathrm{T}}\mathbf{x} = 0 \tag{3.46}$$

$$\Leftrightarrow \quad \widehat{\mathbf{l}}_1^{\mathrm{T}}\,(\mathbf{V}^{\mathrm{T}}\mathbf{V})\,\mathbf{x} = 0 \tag{3.47}$$

$$\Leftrightarrow \quad (\mathbf{V}\widehat{\mathbf{l}}_1)^{\mathrm{T}}(\mathbf{V}\mathbf{x}) = 0\ . \tag{3.48}$$

Using the notation for a line in the image plane $\mathbf{l} = [n_1, n_2, -d\,]^{\mathrm{T}}$, a parameterization is given by

$$\mathbf{x}(t) = \begin{bmatrix} -n_2 t + n_1 d \\ n_1 t + n_2 d \\ 1 \end{bmatrix}\ , \tag{3.49}$$

where $\mathbf{n} = [n_1, n_2]^{\mathrm{T}}$ is the unit normal to the line and d is the distance to the origin. This can be verified by showing that the line equation $\mathbf{x}^{\mathrm{T}}(t)\,\mathbf{l} = 0$ holds. The corresponding contour generator on $\mathbf{Q}$ is parameterized as in equation (3.36), and one of the clipping planes as in equation (3.37). The end point of the line in the image plane is the projection of the point in which the contour generator intersects one of the clipping planes as shown in figure 3.6. This means that the equation $\mathbf{X}^{\mathrm{T}}(t)\,\boldsymbol{\pi} = 0$ has to hold and the value of t at the intersection can be found by solving

$$\mathbf{X}^{\mathrm{T}}(t) \quad \boldsymbol{\pi} \quad = \quad 0 \tag{3.50}$$

$$\Leftrightarrow \quad \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(t) & \dfrac{-\mathbf{x}^{\mathrm{T}}(t)\,\mathbf{b}}{c} \end{bmatrix} \begin{bmatrix} \mathbf{n}_\pi \\ -d_\pi \end{bmatrix} \quad = \quad 0 \tag{3.51}$$

$$\Leftrightarrow \quad t \quad = \quad \frac{d(k_1 n_1 + k_2 n_2) + k_3}{k_1 n_2 - k_2 n_1}\,, \tag{3.52}$$

where the 3-vector $\mathbf{k}$ is defined as in (3.42). By solving this equation for the planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ both line end points are obtained. The end points for line $\mathbf{l}_2$ are found analogously.

3.4 Handling self-occlusion

The next step is the handling of self-occlusion, achieved by comparing the depths of points in 3D space. As in equation (3.23), the depth of a point $\mathbf{x}(\zeta_0)$ on the contour generator of a quadric $\mathbf{Q}$ is obtained by computing $\zeta_0 = -(\mathbf{b}^{\mathrm{T}}\mathbf{x})/c$. In order to check if $\mathbf{x}(\zeta_0)$ is visible, equation (3.19) is solved for each of the other quadrics $\mathbf{Q}_i$ of the hand model. In order to speed up the computation, it is first checked whether the 2D bounding boxes of the corresponding conic segments overlap in the image. If this is not the case, the calculation of the ray and conic intersection can be omitted. In the general case there are two solutions ζ_1^i and ζ_2^i, yielding the points where the ray intersects with quadric $\mathbf{Q}_i$. The point $\mathbf{x}(\zeta_0)$ is visible if it is closer to the camera than all other points, i.e. $\zeta_0 \geq \zeta_j^i \quad \forall i, j$, in which case the point $\mathbf{x}$ is drawn. Figure 3.7 shows examples of the projection of the hand model with occlusion handling.

The required time for projecting the model depends on the number of points plotted and the number of conic sections overlapping in the image plane. Using 500 contour points, it takes approximately 10 ms to graph one projection, where the model parameter update takes 1 ms, the generation of 2D image points takes 6 ms, and the occlusion handling 3 ms. The measurements were made on a 1.4 GHz machine with an Athlon 4 processor.

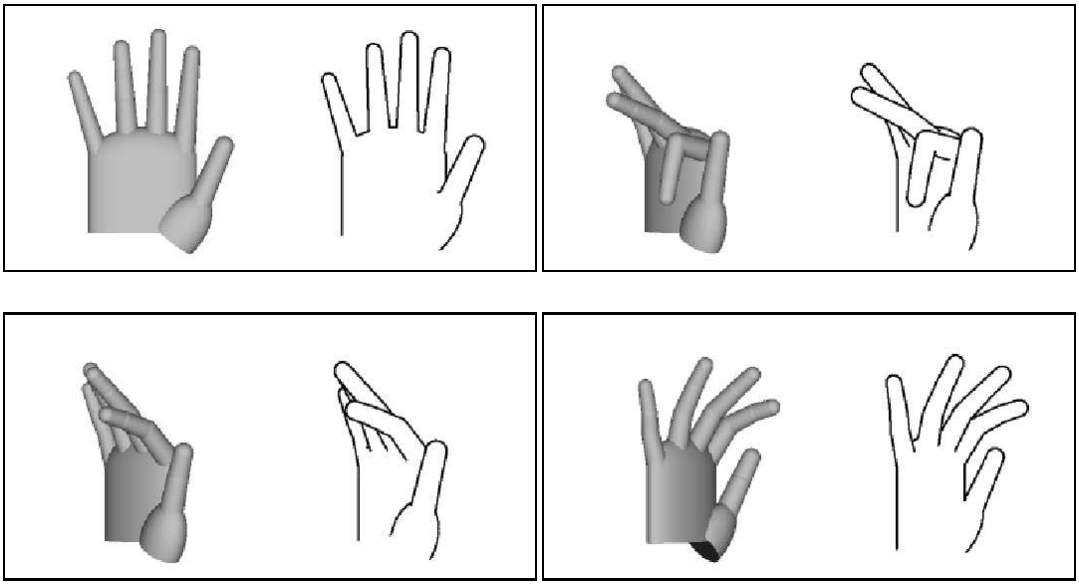


Figure 3.7: **Projected contours of the hand model.** This figure shows four examples of the 3D model (left) and the corresponding generated contour (right).

3.5 Summary

In this chapter it was shown how to construct a three-dimensional hand model from truncated quadrics. For geometric models that are used in tracking, there is typically a trade-off between accuracy and efficiency. More sophisticated models have been used to model the full human body, e.g. deformable super-quadrics [130] or smooth implicit surfaces [106]. This permits more accurate pose and shape recovery at the cost of a more complex estimation problem.

Various hand models have been used in computer graphics for animation purposes. Commercial software products, such as *VirtualHand* and *Poser*, include polygonal mesh models of a human hand, which can be texture mapped. Even though this software has been originally developed for visualization purposes, it can also be used for the task of pose estimation [4, 6, 117]. Albrecht *et al.* [2] recently presented an anatomically based hand model, which incorporates details such as muscle contraction and skin deformation. The hand model, which was described in this chapter is far less accurate in that it does not model local deformations. This implies that when the model and the hand are in the same pose, a residual error between model projection and hand image remains. However, it may be assumed that this error is not significant for the purpose of pose recovery, and may be tolerated given the efficient contour projection.

4 Model-Based Tracking Using an Unscented Kalman Filter

We love to expect, and when expectation is either disappointed or gratified, we want to be again expecting.

Samuel Johnson (1709–1784)

This chapter presents a method for 3D model-based hand tracking based on the *unscented Kalman filter* (UKF). The unscented Kalman filter is an extension of the Kalman filter to nonlinear systems. In the hand tracking application nonlinearity is introduced by the observation model, which involves projecting the 3D model into the image.

The next section gives a brief review of Bayesian state estimation and motivates the Kalman filter for linear systems from a probabilistic point of view. The unscented Kalman filter is then introduced and applied to 3D hand tracking. Experiments from single and dual camera views demonstrate the performance of this method.

4.1 Tracking as probabilistic inference

This section gives a brief introduction to recursive Bayesian estimation, which is the theoretical platform for the rest of this thesis. For a more thorough treatment of probability and estimation theory the reader is referred to a number of textbooks [71, 94, 103].

The internal state of the object at time t is represented as a set of model parameters, which are modelled by the random variable $\mathcal{X}_t$. The measurement obtained at time t are values of the random variable $\mathcal{Z}_t$. Tracking can be formulated as an inference problem: Given knowledge about the observations up to and including time t, $\mathbf{z}_{1:t} = \{\mathbf{z}_i\}_{i=1}^{t}$, the

aim is to find the distribution of the state $\mathcal{X}_t$:

$$p(\mathcal{X}_t|\mathbf{z}_{1:t}) \ . \tag{4.1}$$

where the notation $p(\mathbf{z}_t)$ is used as a shorthand for $p(\mathcal{Z}_t = \mathbf{z}_t)$. Two independence assumptions are typically made, which yield significant simplifications.

- The *first order Markov assumption* states that the state at time t only depends on the state at the previous time instant $t-1$:

$$p(\mathcal{X}_t|\mathcal{X}_{t-1}, \ldots, \mathcal{X}_1) \ = \ p(\mathcal{X}_t|\mathcal{X}_{t-1}) \ . \tag{4.2}$$

- The second assumption states that the measurement at time t is *conditionally independent* of all past measurements given $\mathcal{X}_t$:

$$p(\mathcal{Z}_t|\mathcal{X}_t, \mathbf{z}_{1:t-1}) \ = \ p(\mathcal{Z}_t|\mathcal{X}_t) \ . \tag{4.3}$$

State estimation is typically expressed as a time-ordered pair of recursive equations, namely a *prediction* step and a *update* step. For initialization it is assumed that $p(\mathcal{X}_1)$ is given, which is the state distribution in the absence of any measurement. When the observation $\mathbf{z}_1$ is obtained, the distribution is updated using Bayes rule:

$$p(\mathcal{X}_1 \mid \mathcal{Z}_1) \ = \ \frac{p(\mathbf{z}_1|\mathcal{X}_1) \ p(\mathcal{X}_1)}{\int p(\mathbf{z}_1|\mathcal{X}_1) \ p(\mathcal{X}_1) \ d\mathcal{X}_1} \ . \tag{4.4}$$

In the prediction step at time t, it is assumed that the estimate of $p(\mathcal{X}_{t-1}|\mathbf{z}_{1:t-1})$ from the previous time step is given. The state distribution given all previous measurements, $p(\mathcal{X}_t|\mathbf{z}_{1:t-1})$, is then computed as

$$p(\mathcal{X}_t|\mathbf{z}_{1:t-1}) \ = \ \int p(\mathcal{X}_t|\mathcal{X}_{t-1}) \ p(\mathcal{X}_{t-1}|\mathbf{z}_{1:t-1}) \ d\mathcal{X}_{t-1} \ . \tag{4.5}$$

This relation is also referred to as the *Chapman-Kolmogorov* equation [71]. After obtaining the observation at time t, the distribution is updated as follows

$$p(\mathcal{X}_t|\mathbf{z}_{1:t}) \ = \ c_t^{-1} \ p(\mathbf{z}_t|\mathcal{X}_t) \ p(\mathcal{X}_t|\mathbf{z}_{1:t-1}) \ , \tag{4.6}$$

where the partition function is given by

$$c_t \ = \ \int p(\mathbf{z}_t|\mathcal{X}_t) \ p(\mathcal{X}_t|\mathbf{z}_{1:t-1}) \ d\mathcal{X}_t \ . \tag{4.7}$$

Equation 4.6 can be interpreted as weighting a prior distribution $p(\mathcal{X}_t|\mathbf{z}_{1:t-1})$, computed in the prediction step, with the likelihood $p(\mathbf{z}_t|\mathcal{X}_t)$. The posterior distribution $p(\mathcal{X}_t|\mathbf{z}_{1:t})$ is obtained by normalization and is used in the next prediction step.

Algorithm 4.1 The Kalman filter equations

Dynamic model:

$$\mathbf{x}_t = \mathbf{\Phi}_{t-1}\mathbf{x}_{t-1} + \mathbf{v}_{t-1} \tag{4.10}$$
$$\mathbf{z}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{w}_t \tag{4.11}$$

$$\text{with} \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_{\phi_t}), \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_{h_t}) \ . \tag{4.12}$$

Initialization:

$$\bar{\mathbf{x}}_1^-, \Sigma_1^- \text{ are given.} \tag{4.13}$$

Prediction step:

$$\bar{\mathbf{x}}_t^- = \mathbf{\Phi}_t\bar{\mathbf{x}}_{t-1}^+ \tag{4.14}$$
$$\Sigma_t^- = \mathbf{\Phi}_t\Sigma_{t-1}^+\mathbf{\Phi}_t^{\mathrm{T}} + \Sigma_{\phi_t} \tag{4.15}$$

Update step:

$$\mathbf{K}_t = \Sigma_t^-\,\mathbf{H}_t^{\mathrm{T}}\left(\mathbf{H}_t\Sigma_t^-\mathbf{H}_t^{\mathrm{T}} + \Sigma_{h_t}\right)^{-1} \tag{4.16}$$
$$\bar{\mathbf{x}}_t^+ = \bar{\mathbf{x}}_t^- + \mathbf{K}_t\left(\mathbf{z}_t - \mathbf{H}_t\bar{\mathbf{x}}_t^-\right) \tag{4.17}$$
$$\Sigma_t^+ = \left(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t\right)\,\Sigma_t^- \tag{4.18}$$

Solving the filtering equations analytically involves solving several integrals, which are intractable in the general case. A prominent exception is the Gaussian probability distribution function. This is exploited by the Kalman filter [80], which is the optimal filter for systems with linear dynamics and observation function with additive white noise. In this case the distributions involved are Gaussian and they can be represented by their first and second order moments. The Kalman filter is optimal in the sense that is minimizes expected mean squared error of the state estimate [8]. Using the notation from [45, 50]

$$p(\mathcal{X}_t|\mathbf{z}_{1:t-1}) \sim \mathcal{N}(\bar{\mathbf{x}}_t^-, \Sigma_t^-) \tag{4.8}$$

and

$$p(\mathcal{X}_t|\mathbf{z}_{1:t}) \sim \mathcal{N}(\bar{\mathbf{x}}_t^+, \Sigma_t^+) \tag{4.9}$$

the Kalman filter equations are given in algorithm 4.1. The superscripts indicate whether values are computed before (-) or after (+) the new measurement $\mathbf{z}_t$ is obtained.

The filtering problem becomes nonlinear if either the *system function* f in

$$\mathbf{x}_t = f_{t-1}(\mathbf{x}_{t-1}) + \mathbf{v}_{t-1} \tag{4.19}$$

or the *observation function* h in

$$\mathbf{z}_t \;=\; h_t(\mathbf{x}_t) + \mathbf{w}_t \tag{4.20}$$

are nonlinear. The vectors $\mathbf{v}_{t-1}$ and $\mathbf{w}_t$ represent the system and observation noise, respectively. A common approach to dealing with nonlinearity is to linearize the equations and apply the standard Kalman filter. For example, the extended Kalman filter (EKF) [70] uses a first order Taylor approximation to the system and observation function, assuming that the second and higher order errors are negligible. The unscented Kalman filter (UKF) [75, 77, 85, 138, 144] is based on statistical linearization, and transforms approximations of the distributions through the system and observation functions. The UKF has shown better results than the EKF for a number of estimation problems [75, 139]. A short motivation and overview of the algorithm is given in the following section.

4.2 The unscented Kalman filter

The unscented Kalman filter (UKF) is an algorithm for recursive state estimation in nonlinear systems. As in the standard Kalman filter, the state distribution is represented by a Gaussian random variable. The UKF extends the Kalman filter to nonlinear systems by transforming approximations of the distributions through the nonlinear system and observation functions. This transformation is used to compute predictions for the state and observation variables in the standard Kalman filter. The distribution is represented by a set of deterministically chosen points, also called *sigma points* [139]. These points capture the mean and covariance of the random variable and are propagated through the nonlinear system. This transformation has also been termed *unscented transformation* [76, 145], and can be summarized as follows.

Let $\mathcal{X}$ be an n-dimensional random variable with mean $\widehat{\mathbf{x}}$ and covariance matrix Σ. First, a set of $2n+1$ weighted samples or *sigma points*, $\{(\mathbf{x}^i, w^i)\}_{i=0}^{2n}$, with the same mean and covariance as $\mathcal{X}$ are computed. The following scheme satisfies this requirement:

$$\begin{array}{lll}
\mathbf{x}^i = \widehat{\mathbf{x}}, & w^i = \kappa/(n+\kappa) & \text{for } i = 0 \\
\mathbf{x}^i = \widehat{\mathbf{x}} - \left(\sqrt{(n+\kappa)\Sigma}\right)_i, & w^i = 1/(2(n+\kappa)) & \text{for } i = 1,\ldots,n \\
\mathbf{x}^i = \widehat{\mathbf{x}} + \left(\sqrt{(n+\kappa)\Sigma}\right)_i, & w^i = 1/(2(n+\kappa)) & \text{for } i = n+1,\ldots,2n
\end{array}, \tag{4.21}$$

where $\kappa \in \mathbb{R}$ is a scaling factor and $(\sqrt{(n+\kappa)\Sigma})_i$ denotes the ith column of the matrix $\sqrt{(n+\kappa)\Sigma}$. The square root $\sqrt{\mathbf{M}}$ of a positive definite matrix $\mathbf{M}$ is defined such that $\sqrt{\mathbf{M}}\sqrt{\mathbf{M}}^{\mathrm{T}} = \mathbf{M}$ and it can be computed using Cholesky decomposition [77]. These

sigma points have the same mean and covariance as the random variable $\mathcal{X}$ [75]. Each point $\mathbf{x}^i$ is then transformed by the nonlinear function f

$$\mathbf{y}^i = f(\mathbf{x}^i) \qquad , \; i = 0, \ldots, 2n \tag{4.22}$$

and the mean and covariance of the transformed random variable are estimated as

$$\begin{aligned} \widehat{\mathbf{y}} &= \sum_{i=0}^{2n} w^i \mathbf{y}^i \; , \\ \Sigma_y &= \sum_{i=0}^{2n} w^i (\mathbf{y}^i - \widehat{\mathbf{y}})(\mathbf{y}^i - \widehat{\mathbf{y}})^{\mathrm{T}} \; . \end{aligned}$$

It can be shown that these estimates are accurate to the second order of the Taylor expansion [75]. A generalization to the unscented transform has been introduced recently, which addresses the question of how to set the scaling parameter κ and introduces different weights for computing the mean and covariance [78, 139]. The unscented Kalman filter is an application of the unscented transformation to recursive estimation. The transformation is used to compute the predicted state and covariance matrix, as well as the expected observation. In order to include noise terms, the state random variable is augmented by the noise vectors, ensuring that the system noise on the mean and the covariance are introduced with the same accuracy as the uncertainty in the state. An overview of the UKF algorithm is given in algorithm 4.2.

The UKF has shown better performance than the EKF in a number of estimation tasks [75, 139]. In addition the UKF has the advantage that no Jacobian matrices need to be computed and that it is applicable also to non-differentiable functions. Recently, it has been pointed out that the UKF can also be derived using *statistical linear regression*, where the *expected* linearization error is minimized [50, 85, 139], which is in contrast to using the first order Taylor approximation at a single point as in the EKF.

By using only the first two moments, the UKF still assumes a unimodal state distribution. This is inadequate when the distributions are multi-modal and a different representation is needed in this case, for example the sample-based representation used in particle filtering (CONDENSATION) [52, 66]. Both methods can be thought of as representing distribution functions using weighted particle sets. The key differences are that this set is chosen deterministically in the UKF and the number of points required is fixed at $2n + 1$, where n is the state space dimension. Particle filters typically use many more particles, which are obtained using a sampling method. However, the concepts of the unscented Kalman filter and particle filters can be used in a complementary way, for example the unscented transform has been applied to generating proposal distributions within a particle filter framework [138].

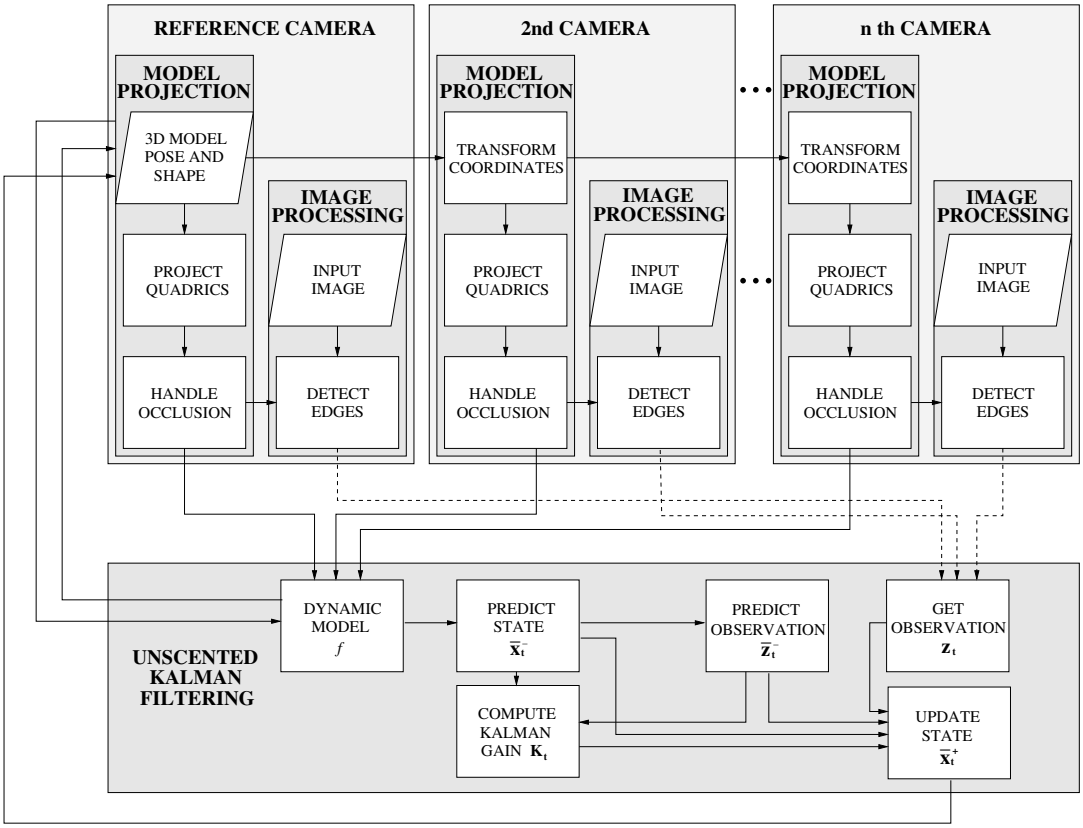


Figure 4.1: **Flowchart of the UKF tracker.** *This diagram shows the process flow for the tracker during a single time step. The model is projected into one or more camera views and is used to find local edges. The UKF computes the predicted state $\bar{\mathbf{x}}_t^-$, as well as the predicted observation $\bar{\mathbf{z}}_t^-$ in order to compute the Kalman gain matrix $\mathbf{K}_t$. These terms are then used to update the estimate of the state $\bar{\mathbf{x}}_t^+$ given the new observation $\mathbf{z}_t$.*

4.3 Object tracking using the UKF algorithm

This section outlines the application of the unscented Kalman filter to model-based tracking. A flowchart of the complete tracking system is given in figure 4.1.

Two dynamic models are used in the experiments, a constant velocity and a constant acceleration model. The system noise is modelled as additive Gaussian noise. Assume that the parameters, which are to be estimated are written as an n-vector $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^{\mathrm{T}}$. A constant velocity model assumes that $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \delta_t \dot{\boldsymbol{\theta}}_{t-1}$, where δ_t is the difference between $t-1$ and t. By stacking the parameter vector and the velocity vector into a single state vector $\mathbf{x}_t = [\boldsymbol{\theta}^{\mathrm{T}}, \dot{\boldsymbol{\theta}}^{\mathrm{T}}]^{\mathrm{T}}$, the dynamic equation can be conveniently written as

$$\mathbf{x}_t = \boldsymbol{\Phi}\mathbf{x}_{t-1} + \mathbf{v}_{t-1} \,, \tag{4.36}$$

where

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{I} & \delta_t \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} . \tag{4.37}$$

Similarly the constant acceleration model can be written as a linear system by augmenting the state vector by the acceleration vector such that the state vector is given as $\mathbf{x} = [\boldsymbol{\theta}^{\mathrm{T}}, \dot{\boldsymbol{\theta}}^{\mathrm{T}}, \ddot{\boldsymbol{\theta}}^{\mathrm{T}}]^{\mathrm{T}}$. In this case the system matrix is given by

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{I} & \delta_t \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \delta_t \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{4.38}$$

All of the above dynamic models are described by linear transformations. Nonlinearity is introduced by the observation function. The observation vector $\mathbf{z}$ is obtained by projecting the model into each camera view and finding local features along vectors normal to the contour. Two different types of features are used:

- The input sequences in the first set of experiments consist of greyscale images. In these experiments the features are edges in the neighbourhood of the projected contour. Let $\{s_i^v\}_{i=1}^{n_s^v}$ be a set of contour points in the image of camera v. For each point s_i^v the normal vector $\mathbf{n}_i^v$ is obtained as explained in section 3.1 and along each normal vector the strongest local edge is detected [15, 27, 55]. For this the intensity values along the normal vector are convolved with a Gaussian kernel and the largest absolute value is labelled as edge location $\mathbf{e}(s_i^v)$.

- In a second set of experiments, colour sequences are used and the extracted features are local skin colour edges [90]. A colour histogram in RGB space is used to classify pixels into skin and non-skin pixels. For this set of experiments measurements are only taken from points on the silhouette, and the point of transition from skin to non-skin colour is used as the observation $\mathbf{e}(s_i^v)$.

The observation vector $\mathbf{z}$ is constructed by stacking the inner products

$$\mathbf{e}(s_i^v)^{\mathrm{T}} \mathbf{n}_i^v \qquad i = 1, \ldots, n_s^v \,, \quad v = 1, \ldots, n_v \tag{4.39}$$

into a single vector. The *sigma points* correspond to points in the state space, symmetrically distributed around the current state estimate. After transforming them using the dynamic model, the moments of the predicted state, $\bar{x}_t^-$ and Σ_t^- are computed. Each sigma point corresponds to a parameter vector, which is used to project the model in a particular pose in order to obtain $\tilde{\mathbf{z}}^i$. The predicted observation vector $\bar{\mathbf{z}}_t^-$ at time t in the UKF algorithm is obtained by projecting the hand model corresponding to the state

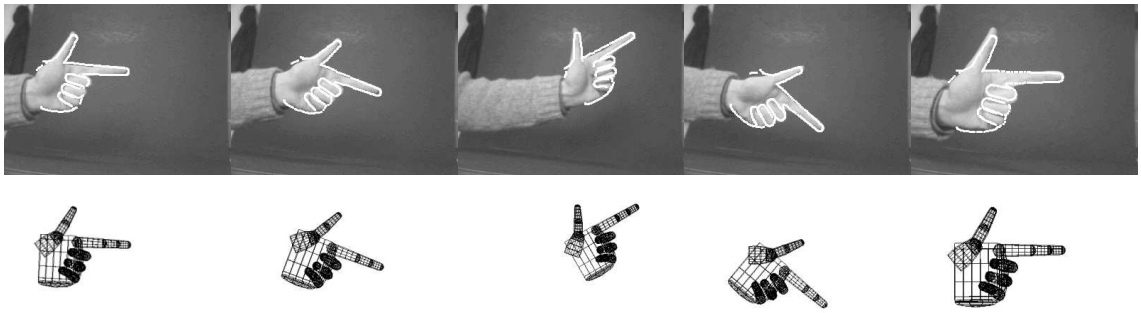


Figure 4.2: **Tracking a pointing hand using local edge features.** *This figure shows frames from a sequence (of 210 frames) while the motion of an open hand. The motion tracked has 4 DOF, including x, y, and z-translation as well as rotation around the z-axis. The observed features are local intensity edges and a constant velocity model is used.*

vectors $\tilde{\mathbf{x}}^i$ into each image and averaging these observation vectors. Let $\{\tilde{s}_i^v\}_{i=1}^{n_s^v}$ denote the points of this predicted observation in each view v, then the entries in the innovation vector $(\mathbf{z} - \bar{\mathbf{z}}^-)$ have the form

$$\left(\mathbf{e}(\tilde{s}_i^v) - \tilde{s}_i^v\right)^{\mathrm{T}} \mathbf{n}_i^v \ . \tag{4.40}$$

A component of the innovation is the distance between the average contour point $\tilde{s}_i^v$ and the corresponding feature point $\mathbf{e}(\tilde{s}_i^v)$. Thus the innovation can be interpreted as the error in pixels between the projection of the hand model in each image and the edges in that image. In order to be scale-invariant it is normalized by the number of points on the contour.

There are different options of determining the set of contour points, which are used to obtain the observations. One possibility is to keep the number of points per quadric fixed. Alternatively, the distance between the points can be kept constant. This is the approach taken in the experiments, as this has the effect that the number of points changes with the size of the hand in the image. If the hand is closer to the camera, more points are used in the observation than when it is further away. This complies with the observation that there is a lot of edge data when the hand is close, which can be used for a more accurate estimate of the hand pose. Furthermore, different weight can be given to the edge measurement at different parts of the hand. In the experiments the sample rate at the finger tips is increased in order to obtain a more accurate model fit at these locations.

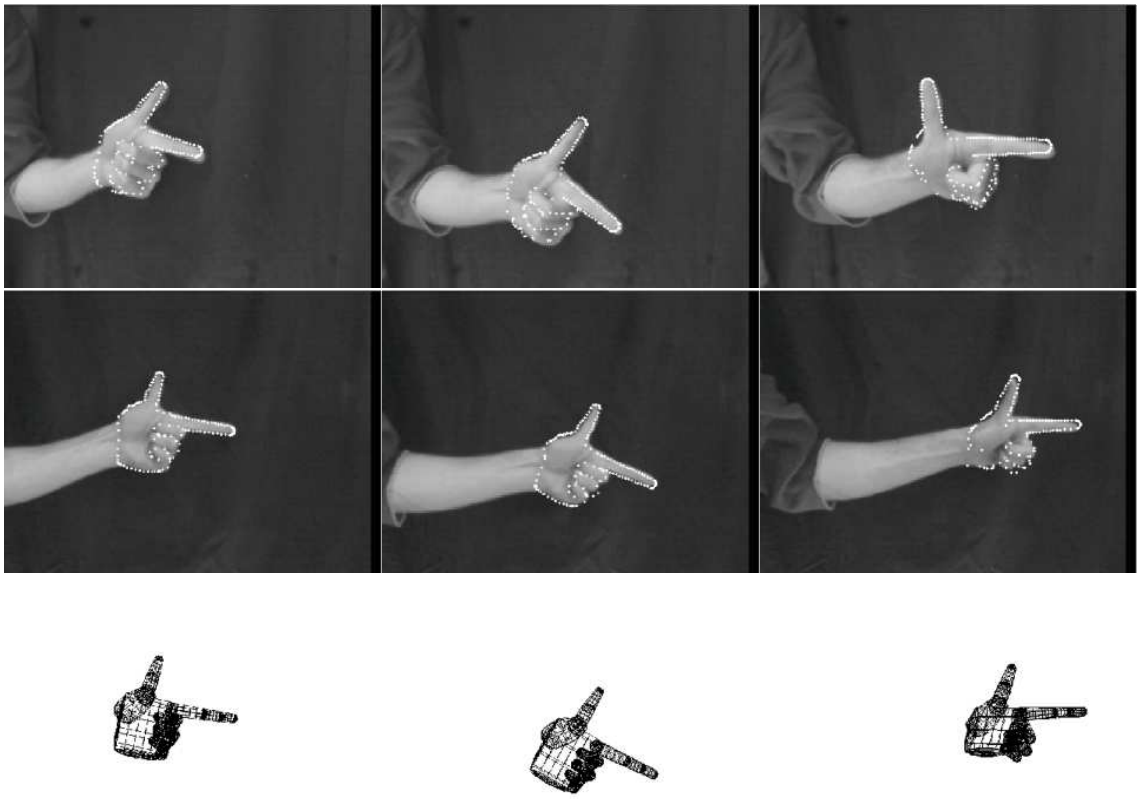


Figure 4.3: **Tracking a pointing hand using two calibrated cameras.** *This figure shows successful tracking of 6 DOF motion of a pointing hand. Local edge features are used as observations.* **Top** *and* **middle row:** *Model projected on images from camera 1 and 2, respectively.* **Bottom row:** *corresponding 3D pose as seen in view 2.*

4.4 Experimental results

The proposed tracking method was tested in a number of experiments. The hand model parameters are manually adusted to match the pose of the hand in the first frame of each sequence.

In the first set of experiments the input images are greyscale images, taken in front of dark background. The first test sequence demonstrates single view tracking of a pointing hand over a sequence of 210 frames. The tracked hand motion has four degrees of freedom: translation in x, y and z-directions and rotation about the z-axis. The dynamics of the hand is modelled using a first order process, i.e. using position and velocity. The hand motion is tracked successfully over the whole sequence, and typical frames shown with the model contours superimposed are shown in figure 4.2. For a single camera the

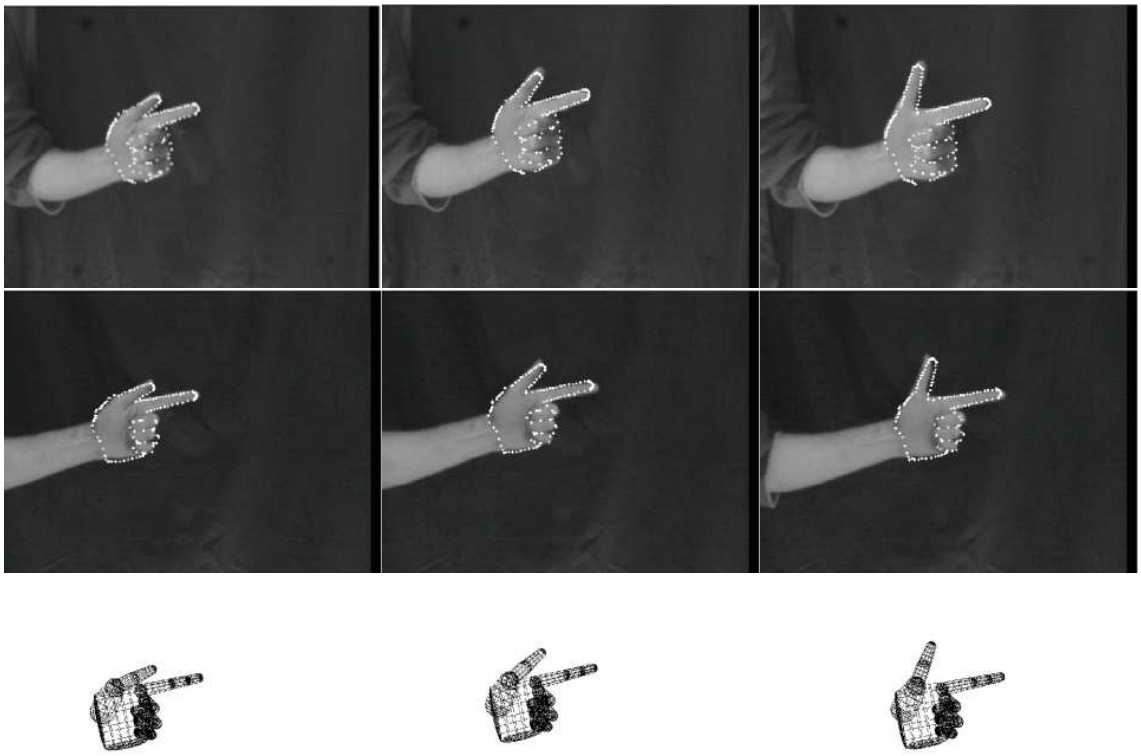


Figure 4.4: **Tracking a pointing hand with thumb extension.** *This figure shows successful tracking of 7 DOF motion of a pointing hand using local intensity edges as observations.* **Top** *and* **middle row:** *Model projected on images from camera 1 and 2, respectively.* **Bottom row:** *corresponding 3D pose as seen in view 2.*

tracker operates at a rate of 3 frames per second on a Celeron 433MHz machine. The computational complexity grows linearly with the number of cameras.

To demonstrate the use of multiple views two stereo sequences of eighty 360×288 greyscale images of a hand in front of a dark background were acquired. The cameras were calibrated beforehand using a calibration grid, and the Euclidean transformation between the two camera views was determined in order to project the model into both views. Six degrees of freedom were given to the hand motion, allowing for full rigid body motion. The dynamics of the hand and thumb were modelled using a second order process, i.e. using position, velocity and acceleration. The quality of the estimation of the 3D position and orientation of the model is demonstrated in figure 4.3. The bottom row of the figure shows the corresponding pose of the 3D model. In a second experiment using two camera views, an additional degree of freedom was given to thumb motion, allowing

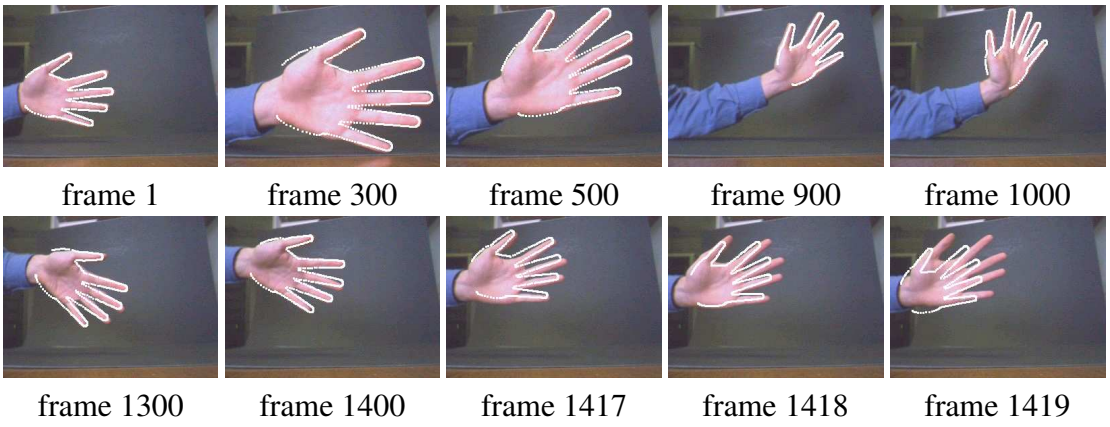


Figure 4.5: **Tracking an open hand using local edge features.** *This figure shows frames from a sequence (of 1810 frames) while the motion of an open hand. The motion tracked has 4 DOF, including x, y, and z-translation as well as rotation around the z-axis. Local edge features are used as observed features and constant velocity is assumed for the dynamic model. Track is lost in the last frames, where the hand undergoes a fast rotation and suddenly stops, whereas the model continues to move.*

for the modelling of arbitrary rigid body motion plus the flexion of the thumb, simulating a "point and click" interface. Figure 4.4 shows typical frames from the tracked sequence and demonstrates correct estimation of the thumb motion.

In order to test the robustness of intensity edge features, a sequence of 1810 colour frames of size 360×240 was captured. In the first experiment, only local intensity edges are used without using any colour information. The hand is tracked accurately for 1400 frames, see figure 4.5. At this point in the sequence the hand rotates around the z-axis and comes to a sudden halt. This leads to loss of track, because the dynamic model predicts continued rotation, leading to incorrect feature association. It can be observed that when the finger tips are slightly misaligned, no edges can be found along the normal at these contour points. However, the measurements at the tips are important, as they are strong cues for motion components in certain directions. In the second experiment the same input sequence was successfully tracked using skin-colour edge features as observations. For this an RGB skin-colour histogram was estimated from the first 10 frames of the sequence. Using skin-colour information helps to avoid incorrect feature association. For this sequence the hand position error was measured against manually labelled ground truth. The RMS error for the skin-colour features was 2.0 pixels, compared to a 3.1 pixel error for intensity edges measured on the first 1400 frames, which were successfully tracked in both cases. Figure 4.6 shows the error performance over the complete sequence.

In another experiment hand motion of 6 DOF is tracked from a single view using

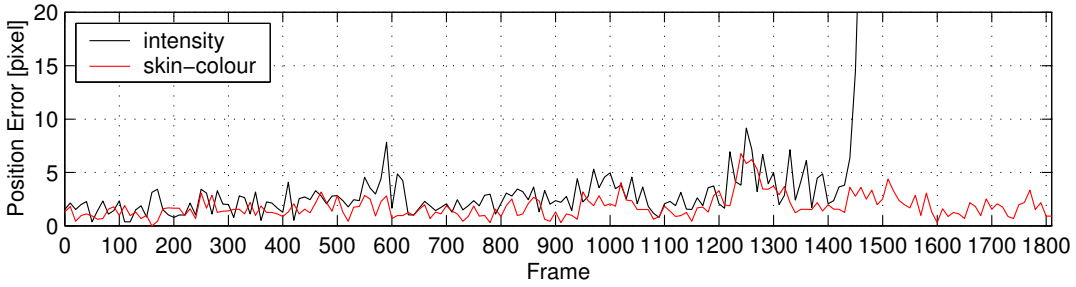


Figure 4.6: **Error comparison for different features.** *This figure shows the error performance of the UKF filter using local intensity edges (black) or skin-colour edges (red) on the sequence in figure 4.5. Errors were measured against manually labelled ground truth. Skin-colour edges improve tracking performance by more robust feature association.*

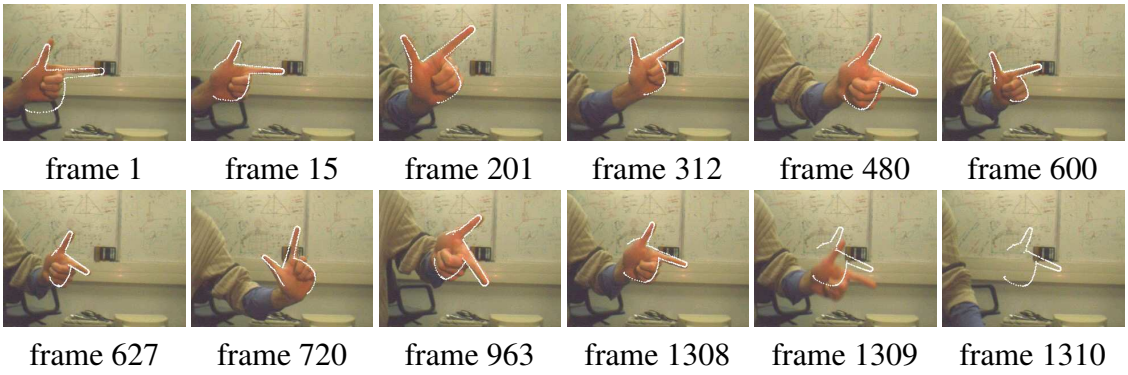


Figure 4.7: **Tracking a pointing hand using skin colour edges.** *This figure shows frames from a sequence tracking 6 DOF using a single camera. The hand motion is tracked accurately over 1400 frames, until the hand is lowered with high velocity, shown in the last three images, which are consecutive frames in the sequence.*

skin-colour edges, as shown in figure 4.7. The hand motion is tracked accurately over 1400 frames, until the hand is lowered with high velocity, shown in the last three images. The constant velocity model is unable to predict this motion and the tracker needs to be re-initialized before tracking can resume.

The experiments have demonstrated successful tracking of smooth hand motion with limited articulation, namely thumb articulation in a “point and click” sequence. Nevertheless, the tracker has a number of draw-backs, limiting its practical use.

4.5 Limitations of the UKF tracker

This section briefly summarizes limitations of the approach presented in this chapter.

Initialization and recovery The tracker uses incremental estimation and therefore requires manual initialization of the pose parameters in the first frame. In the experiments this is solved by roughly aligning the model and starting the tracker from this initial pose. However, in an interactive application, possibly with fast hand motion or with the hand leaving the camera view, it is not desirable to hold the hand in a particular pose each time track has been lost.

Ambiguous poses When using a single view, there are several poses where a subset of state parameters become unobservable, e.g. due to self-occlusion [15, 39]. In such cases the underlying distributions become multimodal and a tracker, which can handle multi-modality has to be used. Since the UKF assumes unimodal distributions, it cannot handle such cases. The use of multiple cameras can help to resolve some of the ambiguities.

Feature robustness For the greyscale sequences local edges are used, which could only be reliably detected in front of neutral background. As soon as background clutter is introduced, tracking becomes unstable. Skin colour edges have shown to be more reliable. However, only points on the silhouette can be used to obtain these measurements, and therefore information from the internal region of the hand is not included. Furthermore, skin colour classification is noisy in general, and the features disappear completely when a skin-coloured background is introduced.

4.6 Summary

In this chapter a model-based hand tracking system based on the unscented Kalman filter was presented. The observed features are intensity edges or skin-colour edges, found by local search along the contour normals. Skin-colour edges have been shown to make tracking more robust by improving feature correspondence, however, they rely on successful skin-colour segmentation. Chapter 5 examines how both edge and colour features can be integrated to yield a better observation model. The dynamic models used are constant velocity and constant acceleration models. These are reasonable models when the tracked motion is smooth, but they fail to capture very fast or abrupt hand motion. The system scales from single to multiple views and experiments using single and dual camera views were shown. The main limitations of the approach are that it requires manual initialization and does not have a recovery strategy when track is lost. Also, the assumption

of the UKF of unimodality does not hold in several cases, as there can be ambiguity when tracking a hand in a single view.

Algorithm 4.2 Unscented Kalman filter (UKF)

Dynamic model:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{v}_{t-1}, t-1) \tag{4.23}$$

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{w}_t, t) \tag{4.24}$$

Initialization:

$$\bar{\mathbf{x}}_1^-, \Sigma_1^- \text{ are given.} \tag{4.25}$$

Prediction step:

(a) Compute *sigma points* $\{(\mathbf{x}_{t-1}^i, w^i)\}_{i=0}^{2n}$ as in (4.21) from the estimate at time $t-1$.

(b) Transform the points $\mathbf{x}_{t-1}^i$ and compute moments of the predicted state variable:

$$\tilde{\mathbf{x}}_t^i = f(\mathbf{x}_{t-1}^i, \mathbf{v}_{t-1}, t-1) \tag{4.26}$$

$$\bar{\mathbf{x}}_t^- = \sum_i w^i \, \tilde{\mathbf{x}}_t^i \tag{4.27}$$

$$\Sigma_t^- = \sum_i w^i \, (\tilde{\mathbf{x}}_t^i - \bar{\mathbf{x}}_t^-)(\tilde{\mathbf{x}}_t^i - \bar{\mathbf{x}}_t^-)^{\mathrm{T}} \tag{4.28}$$

(c) Compute moments of the predicted observation variable:

$$\tilde{\mathbf{z}}_t^i = h(\tilde{\mathbf{x}}_t^i, \mathbf{w}_t, t) \tag{4.29}$$

$$\bar{\mathbf{z}}_t^- = \sum_i w^i \, \tilde{\mathbf{z}}_t^i \tag{4.30}$$

$$\Sigma_{zz,t}^- = \sum_i w^i \, (\tilde{\mathbf{z}}_t^i - \bar{\mathbf{z}}_t^-)(\tilde{\mathbf{z}}_t^i - \bar{\mathbf{z}}_t^-)^{\mathrm{T}} \tag{4.31}$$

$$\Sigma_{xz,t}^- = \sum_i w^i \, (\tilde{\mathbf{x}}_t^i - \bar{\mathbf{x}}_t^-)(\tilde{\mathbf{z}}_t^i - \bar{\mathbf{z}}_t^-)^{\mathrm{T}} \tag{4.32}$$

Update step:

$$\mathbf{K}_t = \Sigma_{xz,t}^- (\Sigma_{zz,t}^-)^{-1} \tag{4.33}$$

$$\bar{\mathbf{x}}_t^+ = \bar{\mathbf{x}}_t^- + \mathbf{K}_t \left(\mathbf{z}_t - \bar{\mathbf{z}}_t^-\right) \tag{4.34}$$

$$\Sigma_t^+ = \Sigma_t^- - \mathbf{K}_t \, \Sigma_{zz,t}^- \, \mathbf{K}_t^{\mathrm{T}} \tag{4.35}$$

5 Similarity Measures for Template-Based Shape Matching

I found I could say things with colour and shapes that I couldn't say any other way – things I had no words for.

Georgia O'Keeffe (1887–1986)

The task of robust tracking demands a robust observation model. In the previous chapter intensity and colour edges have been used. Either of these feature types is not reliable enough in the general case of a cluttered background, which may also contain skin-colour. It is therefore necessary to examine in more detail how a hand shape can be detected in an arbitrary image. In this chapter the detection task is formulated as a template matching problem and it is examined how shape and colour information can be used to solve it. In terms of image regions, edge and skin colour features can be seen as complementary, as they describe the boundary and the internal area of the hand. A hand in front of a complex, but non-skin coloured background may be recognized by its colour, whereas a hand in a greyscale image may still be recognized by its shape. The two feature types are treated separately in the following sections, and are then combined into a single cost function in section 5.4.

Shape carries significant visual information about an object. In the absence of other cues, people can usually still recognize objects by their shape, for example in line drawings. A common way to extract shape information from an image is by edge detection [25]. The first section therefore examines methods for robust shape matching based on distance transforms and compares them to a number of methods which could be employed.

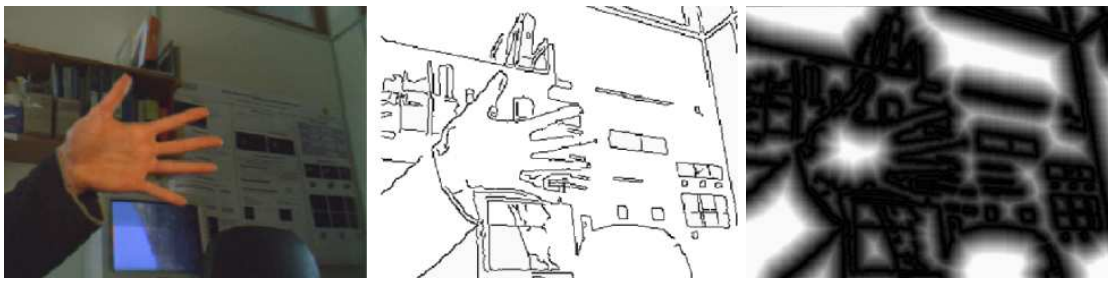
5.1 Shape matching using distance transforms

Image edges provide a strong cue for the human visual system which is often sufficient for scene interpretation [81, 91]. They are typically represented by a set of feature points, such as Canny edges. A common approach, when trying to locate a particular object in a scene based on its shape, is to use a prototype object shape and search for it in the image.

There are a number of shape matching algorithms. Some of them explicitly establish point correspondences between two shapes and subsequently find a transformation that aligns the two shapes. These two steps can be iterated, and this is the principle of algorithms such as *iterated closest points* (ICP) [13, 26] or *shape context matching* [10]. In order to converge, these methods require good initial alignment, particularly if the scene contains a cluttered background.

In this work a different approach is taken, which is searching the transformation space explicitly using *chamfer* or *Hausdorff matching*, which are closely related techniques. Chamfer matching is an efficient way for matching shapes, in which the model template is correlated with a distance transformed edge image. The chamfer distance function was first proposed by Barrow *et al.* [9] and improved versions of it have been used for object recognition and contour alignment. For example, Borgefors [16] introduced hierarchical chamfer matching, using a coarse-to-fine search in the transformation parameter space to locate an object. Olson and Huttenlocher [101] use Hausdorff matching to recognize three dimensional objects from different views using a template hierarchy. Gavrila [47] uses chamfer matching to detect pedestrian shapes in real time. The chamfer distance function is also used by Toyama and Blake [135] as a robust cost function within a probabilistic tracking framework.

To formalize the idea of chamfer matching, the shape of an object is represented by a set of points $\mathcal{A} = \{\mathbf{a}_i\}_{i=1}^{N_a}$. In our case, this is a set of points on the projected model contour. The image edge map is represented as a set of feature points $\mathcal{B} = \{\mathbf{b}_i\}_{i=1}^{N_b}$. In order to be tolerant to small shape variations, any similarity function between two shapes should vary smoothly when the point locations change by small amounts. This requirement is fulfilled by chamfer distance functions, which are based on a *distance transform* (DT) of the edge map. This transformation takes a binary feature image as input and assigns each location the distance to its nearest feature, see figure 5.1. If $\mathcal{B}$ is the set of edge points in the image, the DT value at location u contains the value $\min_{b \in \mathcal{B}} || u - b ||$. A number of cost functions can be defined using the distance transform [16], one choice being the average of the squared distances between each point of $\mathcal{A}$ and its closest point



(a) (b) (c)

Figure 5.1: **Computing the distance transform.** *This figure shows* **(a)** *an input image, its* **(b)** *edge map and* **(c)** *the thresholded distance transformed edge map, which contains at each pixel the distance to the nearest edge point, here represented by greyscale values.*

in $\mathcal{B}$:

$$d_{cham}(\mathcal{A},\mathcal{B}) \;=\; \frac{1}{N_a}\sum_{a\in\mathcal{A}}\min_{b\in\mathcal{B}}\,||\,a-b\,||^2. \tag{5.1}$$

The distance between a template $\mathcal{A}$ and an edge map $\mathcal{B}$ can then be computed by adding the squared DT values at the template point coordinates and dividing by the number of points N_a. The DT can be computed efficiently using a two-pass algorithm over the image [16], thus the operation is linear in the number of pixels. It only needs to be computed once for each input image and can be used to match more than one template. Therefore large computational savings can be achieved compared to methods that explicitly search for feature correspondence between each template and the feature image, for example by searching for edges along the normals to the contour. The DT can also be viewed as a smoothing operation in the feature space. Without smoothing, the correlation of a template $\mathcal{A}$ with an edge map $\mathcal{B}$ leads to a sharply peaked cost function, which is not robust towards small perturbations of the contour. In contrast to this, the value of the function $d_{cham}(\mathcal{A},\mathcal{B})$ in equation (5.1) varies smoothly when the template $\mathcal{A}$ is translated away from the global minimum, thereby smoothing the cost surface, as illustrated in figure 5.2 (c). The cost function can be made more robust by thresholding the DT image by an upper bound. This reduces the effect of missing edges and is therefore more tolerant towards partial occlusion. In this case the matching cost becomes

$$d_{cham}(\mathcal{A},\mathcal{B},\tau) = \frac{1}{N_a}\sum_{a\in\mathcal{A}}\min\left(\min_{b\in\mathcal{B}}||a-b||^2,\tau\right), \tag{5.2}$$

where τ is the threshold value. Note that the chamfer cost function is not a metric, as it is

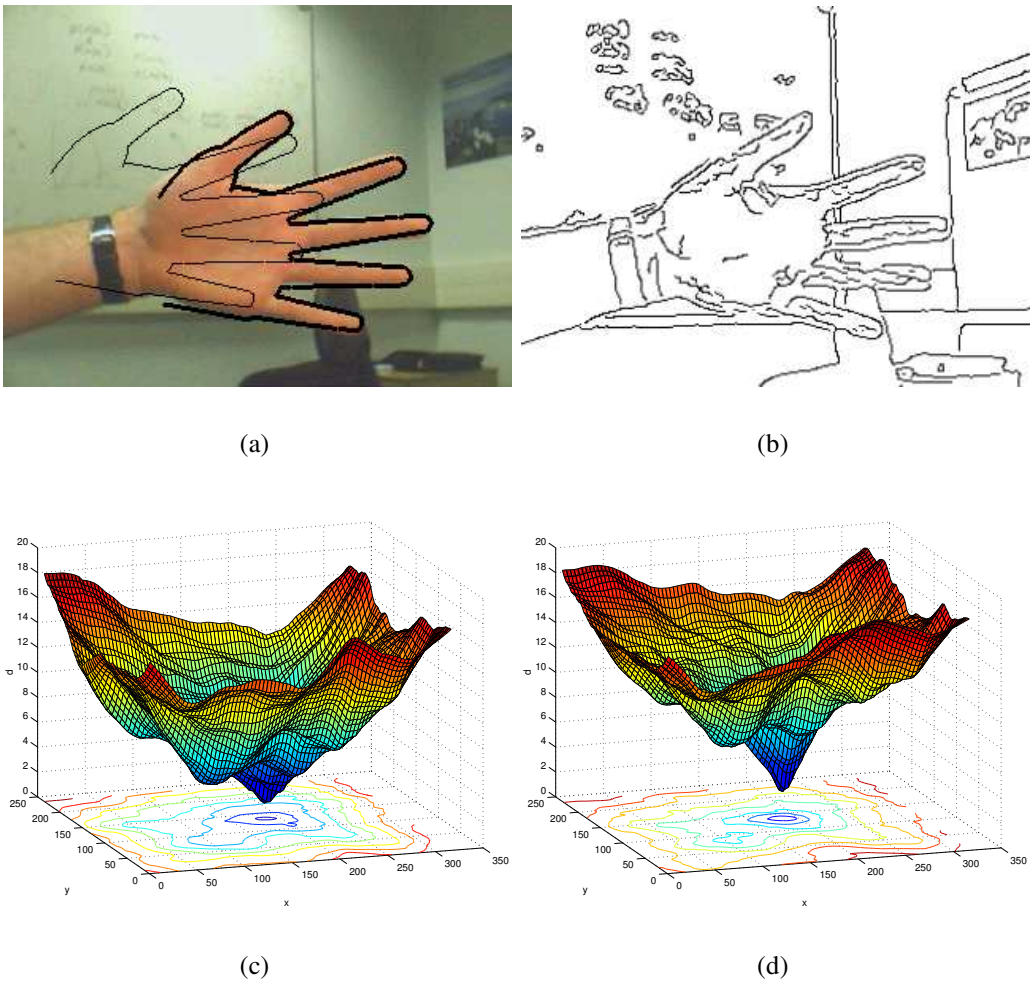


Figure 5.2: **Chamfer cost function in translation space.** *This figure shows* **(a)** *an input image with two templates superimposed, corresponding to the global cost minimum and a local, non-global minimum.* **(b)** *the edge map of the input image.* **(c)** *shows the surface of the chamfer cost using non-oriented edges generated by translating the correct template over the image, and* **(d)** *shows the surface of the chamfer cost using six edge orientations. The position of the global minimum is the same in both cases, but the shape of the cost surface differs. A threshold value of* $\tau = 20$ *is used in this example.*

not a symmetric function. It can be made symmetric, for example by defining a distance as the sum $d_{cham}(\mathcal{A}, \mathcal{B}) + d_{cham}(\mathcal{B}, \mathcal{A})$. However, in the case of matching a template to an image, the term $d_{cham}(\mathcal{B}, \mathcal{A})$ indicates how well edges in the background match to the model. Since we explicitly want to handle images with cluttered background, this term is not computed.

When matching to images with many background edges, the DT image contains low

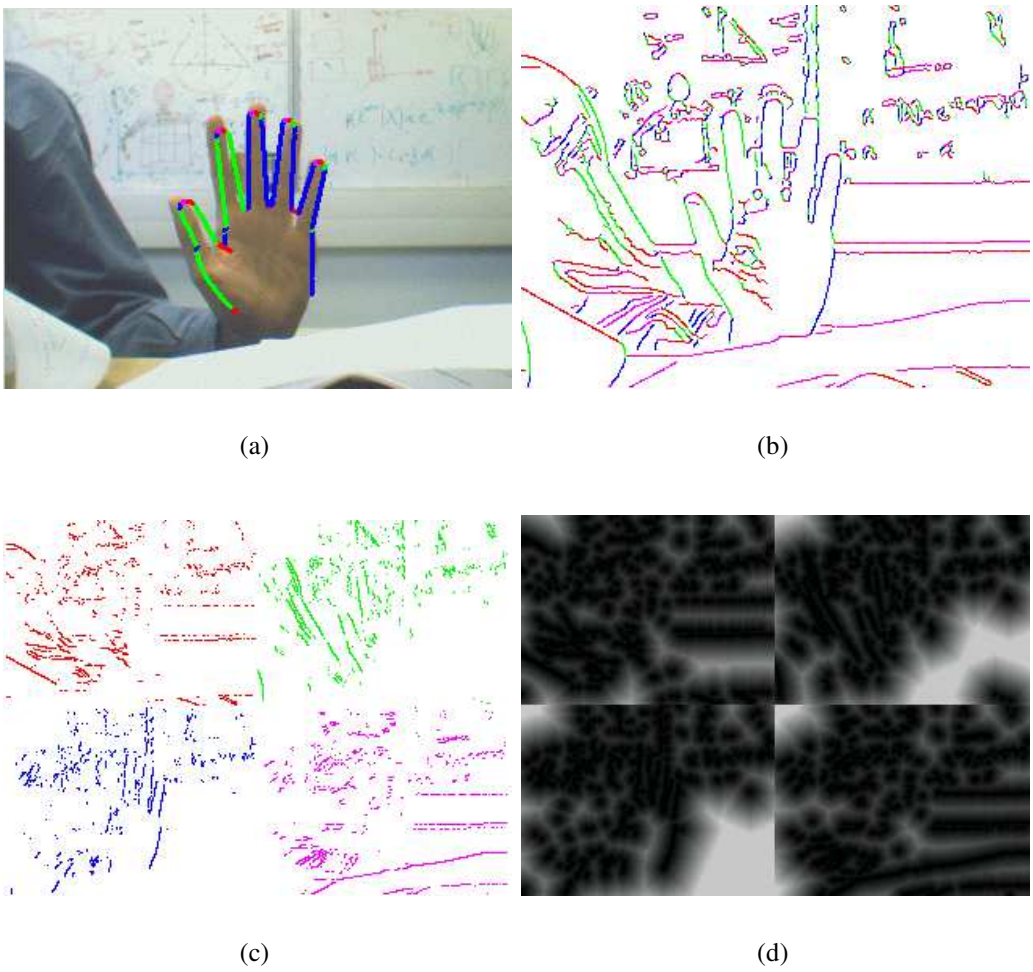
values on average and therefore a single DT image is not sufficient to discriminate between different templates. One way to handle this situation is to make use of the gradient direction as well. The idea is to use a distance measure, which not only considers distance in translation space, but also in gradient orientation space. This increases the discriminatory power of the matching function, as demonstrated by Olson and Huttenlocher for the Hausdorff distance [101]. Each feature point (x, y) is augmented with the gradient direction γ to obtain a vector $\mathbf{p} = (x, y, \gamma)^T$. Given a suitable scaling factor between distances in translation and orientation space, the distance transform can be computed with the analogous two-pass algorithm in three dimensions. However, this is significantly more expensive than in the 2D case. Initial experiments computing a 3D distance transform required approximately 4 seconds per frame, whereas a 2D distance transform can be computed in 1.9 ms (for an image size of 320×240 pixels, times measured on 2.4 GHz Pentium IV machine). In practice the range of orientation is therefore divided into N_γ discrete intervals, with a typical value of $N_\gamma = 6$, which is used in the experiments in this chapter. The edge image is decomposed into N_γ *orientation channels* according to edge orientation, where edges at the boundary of two intervals are assigned to both corresponding channels. The DT is computed separately for each channel and the matching costs for each are then summed up:

$$d_{cham}(\mathcal{A}, \mathcal{B}, \tau) \;=\; \frac{1}{N_a} \sum_{i=1}^{N_\gamma} \sum_{a \in \mathcal{A}^i} \min \left(\min_{b \in \mathcal{B}^i} ||a - b||^2, \tau \right) \tag{5.3}$$

where $\mathcal{A}^i$ and $\mathcal{B}^i$ are the feature points in orientation channel i, see figure 5.3. The sign of the gradient orientation encodes the information whether the transition is from light to dark or vice versa. Since no particular brightness intensity is assumed for the background, this information is discarded, and only the orientation interval $\left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$ is considered. Figure 5.2 (d) shows the cost surface when using eight orientation channels, showing that difference between the value of the global minimum and any other local minimum on the surface is increased.

As pointed out above, chamfer matching avoids the explicit computation of point correspondences. However, the correspondences can easily be obtained during the DT image computation by storing for each value in the DT image the corresponding location. These can be used to incorporate higher order constraints within the cost function, such as continuity or curvature between corresponding points. Such constraints have been successfully used in other shape matching approaches [23, 82, 131].

The chamfer distance function is not invariant towards transformation of the template points, such as translation, rotation or scale. Therefore, each of these cases needs to be



(a) (b)

(c) (d)

Figure 5.3: **Chamfer matching using multiple orientations.** *This figure shows* **(a)** *an input image with the correctly aligned template superimposed and* **(b)** *the edge map of the input image. The colours correspond to different edge orientations.* **(c)** *The edge image is decomposed into a number of channels corresponding to these orientations, and the distance transform is computed separately, shown in* **(d)**. *Here four discrete orientations are used for illustration, and each channel is of the same size as the input image.*

handled by searching over the parameter space. In order to match a large number of templates efficiently, hierarchical search methods have been suggested [16, 47].

5.1.1 The Hausdorff distance

This section briefly describes the *Hausdorff distance function* for shape matching, which is included for comparison with other methods in the following sections. It is closely related to chamfer matching and has been used in a number of applications [17, 64, 101].

Instead of computing an average distance between point sets, Hausdorff matching seeks to minimize the largest distance between the point sets. The Hausdorff distance function between a point set $\mathcal{A}$ and a set of feature points $\mathcal{B}$ is defined as the maximum distance from every point in $\mathcal{A}$ to its nearest point in $\mathcal{B}$. Because this distance can be dominated by an outlier, it is common to use the *partial Hausdorff distance* [64] by taking the k-th ranked distance rather than the maximum:

$$d_{hd}(\mathcal{A}, \mathcal{B}, k) \;=\; \underset{a \in \mathcal{A}}{k\text{th}} \; \min_{b \in \mathcal{B}} \, || \, a - b \, ||, \tag{5.4}$$

where kth denotes the k-th ranked value. For example for $k = |\mathcal{A}|/2$, this is the median distance from points in $\mathcal{A}$ to their closest point in $\mathcal{B}$. Equivalently, instead of fixing k, one may fix a maximum distance value δ and determine the fraction of points with distance less than this value. This is the *Hausdorff fraction*:

$$f_{hd} \;=\; \frac{|\mathcal{A}^\delta|}{|\mathcal{A}|}, \qquad \text{where} \quad \mathcal{A}^\delta = \{a \in \mathcal{A} \;:\; \min_{b \in \mathcal{B}} || \, a - b \, || < \delta\}. \tag{5.5}$$

An efficient way to compute this value is to dilate the edge image by δ and correlate it with the template points in $\mathcal{A}$. The gradient direction can be used in a way analogous to chamfer matching [101].

5.2 Shape matching as linear classification

Both chamfer and Hausdorff matching can be viewed as special cases of linear classifiers [43]. Let $\mathbf{B}$ be the feature map of an image region of size $n \times m$, for example a binary edge image. The template $\mathbf{A}$ is of the same size and can be thought of as a prototype shape. The problem of recognition is to decide whether or not $\mathbf{B}$ is an instance of $\mathbf{A}$. By writing the entries of the matrices $\mathbf{A}$ and $\mathbf{B}$ into vectors $\mathbf{a}$ and $\mathbf{b}$, each of size nm, this task can be formulated as a classification problem with a linear discriminant function $\mathbf{a}^{\mathrm{T}}\mathbf{b} = c$, with constant $c \in \mathbb{R}$. This generalization also permits giving different weights for different parts of the shape, and allows negative coefficients of $\mathbf{a}$, potentially increasing the cost of cluttered image areas. Felzenszwalb [43] has shown that a single template $\mathbf{A}$ and a dilated edge map $\mathbf{B}$ are sufficient to detect a variety of shapes of a walking person. The following section looks at shape matching from a classification point of view. Each template corresponds to a classifier, which is applied to a subregion of an image. The target class includes a number of similar hand poses. For chapter 6 it will be important that the classifier detects hands in certain poses, which correspond to the hand model parameters being within a compact region in the state space. In other words, we

seek to construct classifiers that are "tuned" to certain regions in parameter space. Ideally one such classifier, when applied to a subwindow, should detect a hand in these poses and reject all other image subwindows, i.e. ones with the hand being in a different pose or background regions. Consistent with pattern classification terminology, these will be referred to as negative examples. Perfect class separability is not to be expected. However, by setting the detection threshold low in order to achieve a very high true positive rate, we still hope to reject a significant number of negative example images. The motivation behind this is that such classifiers could be used in a cascaded structure (see section 2.3.2). Thus many negative examples could be rejected early at low computational cost, and more computation could be spent on subwindows that are more difficult to classify. With this in mind the next section examines a number of different linear classifiers based on edge features.

5.2.1 Comparison of classifiers

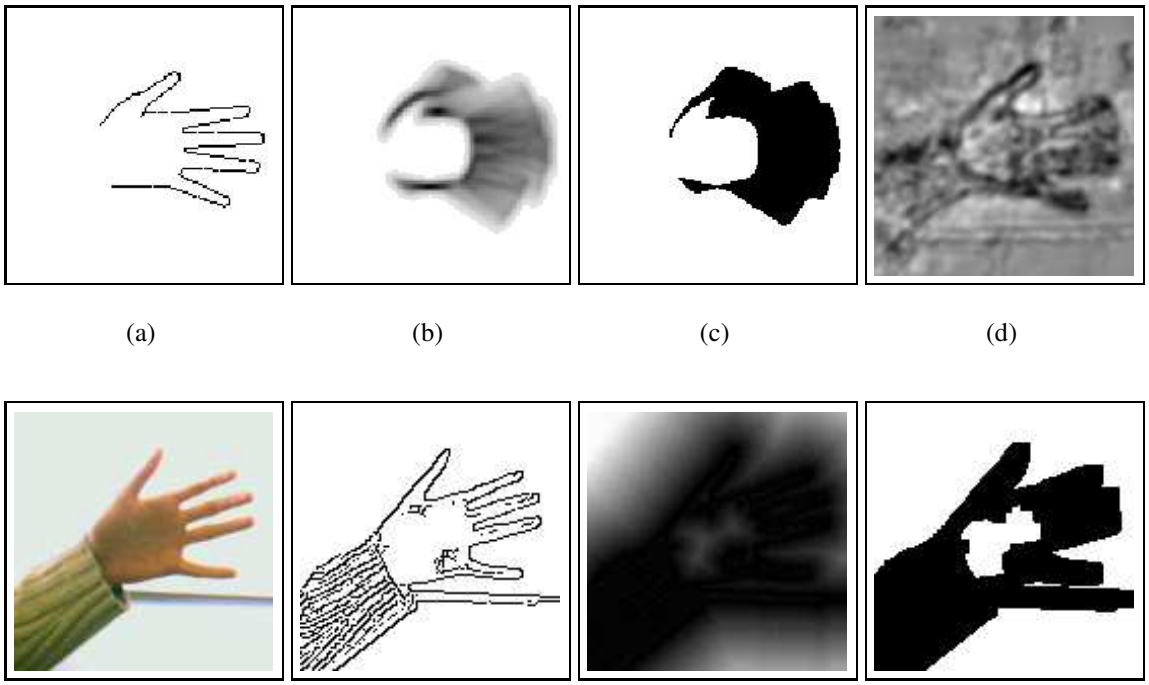
This section compares a number of classifiers in terms of their performance and efficiency. The following classifiers are evaluated (illustrated in figure 5.4, top row):

- **Centre template [Figure 5.4 (a)].** This classifier corresponds to using a template $\mathbf{A}$, generated at the centre of a region in parameter space. Two possibilities for the feature matrix $\mathbf{B}$ are compared. One is the distance transformed edge image in order to compute the truncated chamfer distance as in section 5.1. For comparison, the Hausdorff fraction is computed as in section 5.1.1 using the dilated edge image. The parameters for both methods are set by testing the classification performance on a test set of 5000 images. Values for the chamfer threshold τ from 2 to 120 were tested, and $\tau = 50$ was chosen, but little variation in classification performance was observed for values larger than 20. For the dilation parameter δ values from 1 to 11 were compared, with $\delta = 3$ showing the best performance.

- **Marginalized template [Figure 5.4 (b), (c)].** In order to construct a classifier which is sensitive to a particular region in parameter space, the template $\mathbf{A}$ is constructed by densely sampling the values in this region, and generating the contours for each state. The resulting model projections are then pixel-wise added and smoothed. Different versions of the matrix $\mathbf{A}$ are compared:

  (a) the pixel-wise average of model projections, see figure 5.4 (b),

  (b) the pixel-wise average, additionally setting the background weights uniformly to a negative value such that the sum of coefficients is zero, and



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

(e) (f) (g) (h)

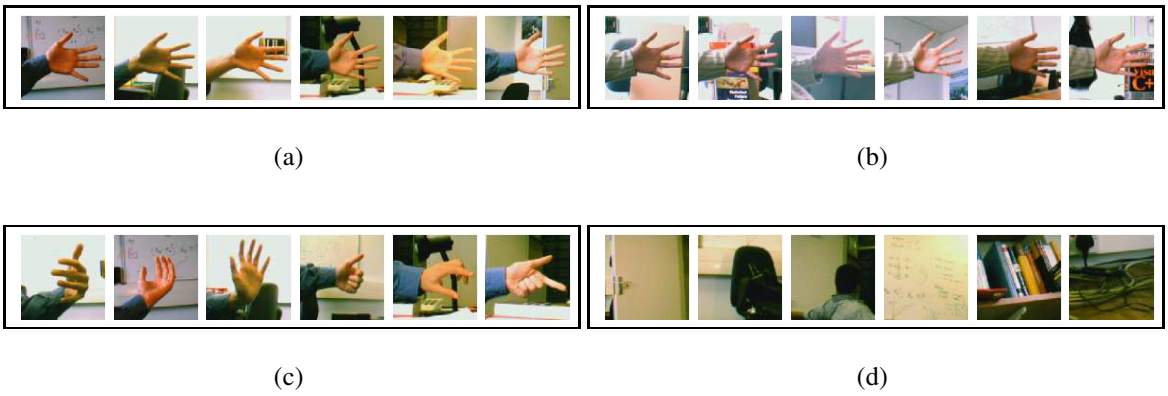
Figure 5.4: **Templates and feature maps used for edge-based classification. Top row:** *Templates* **A** *used for classifying edge features.* **(a)** *single template,* **(b)** *marginalized template created by averaging over a region in parameter space,* **(c)** *binary marginalized template,* **(d)** *template learnt from real image data.* **Bottom row:** *Extracted features from an image.* **(e)** *input image,* **(f)** *edge map,* **(g)** *distance transformed edge image,* **(h)** *dilated edge image.*

(c) the union of all projections, resulting in a binary template, see figure 5.4 (c).

- **Linear classifier learnt from image data [Figure 5.4 (d)].** The template **A** is obtained by learning a classifier as described by Felzenszwalb [43]. A labelled training set consisting of 1,000 positive examples and 4,000 negative examples of which 1,000 contain the hand in a different pose and 3,000 images contain background regions (see figure 5.5) is used to train a linear classifier by minimizing the Perceptron cost function [42, 97].

Two sets of templates are generated, one with orientation information, and one without it. For oriented edges the angle space is subdivided into six discrete intervals, resulting in a template for each orientation channel.

In order to compare the performance of different classifiers, a labelled training set of hand images is collected. The positive class is defined as all images, in which the hand



(a) (b)

(c) (d)

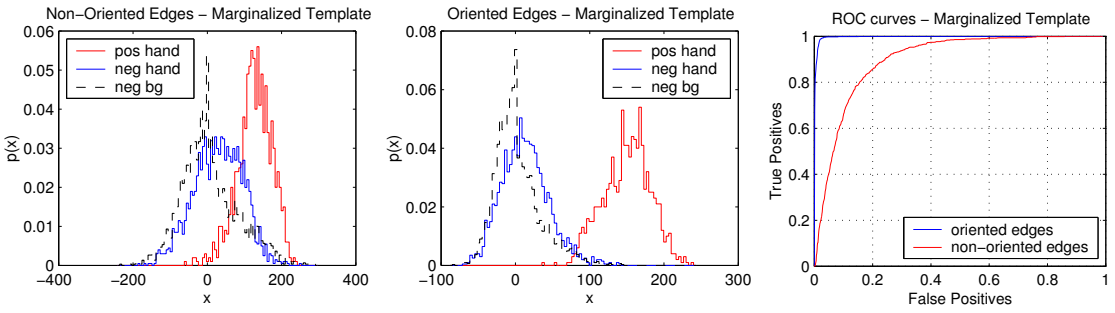
Figure 5.5: **Examples of training and test images.** *The following are example image regions used for training and testing a linear classifier:* **(a)** *positive training examples,* **(b)** *test images,* **(c)** *negative training examples containing a hand in different poses,* **(d)** *negative examples containing office scenes as background.*

pose is within a certain region in parameter space. For the following experiments this region is chosen to be a rotation of 30 degrees parallel to the image plane together with a 3 pixel translation in x- and y-direction. Negative examples are background images as well as images of hands in configurations outside of this parameter region. These are images of a hand in various poses, recorded independently of all three positive image sets, see figure 5.5 (c). The evaluation of classifiers is done in three independent experiments for three different hand poses, an open hand, a pointing hand and a closed hand. The results which were observed consistently in all experiments are reported, even though generalization to other poses is to be examined further. Each of the three test data sets contains 5000 images, of which 1000 are true positive examples. The number of data was chosen such that the performance on a test set of 5000 images did not significantly improve by adding more training data, however choosing the right size of the training data set is an open issue.

5.2.2 Experimental results

The following observations were made in the experiments:

- In all cases the use of edge orientation resulted in better classification performance. Including the gradient direction is particularly useful when discriminating between positive examples and negative examples of hand images. This is illustrated in figure 5.6 (a) and (b), which show the class distributions for non-oriented edges and oriented edges in the case of marginalized templates with non-negative weights.



(a) (b) (c)

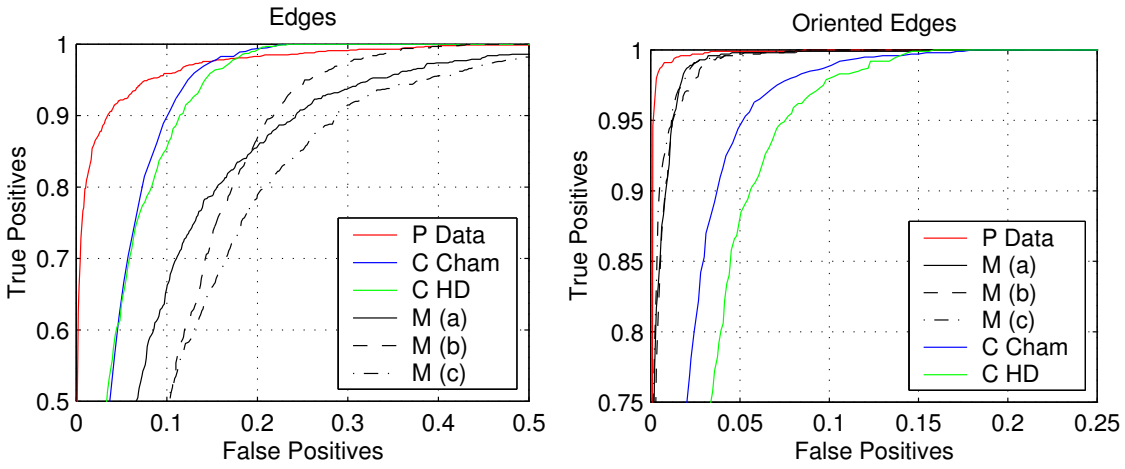
Figure 5.6: **Including edge orientation improves classification performance.** *This example shows the classification results on a test set using a marginalized template.* **(a)** *histogram of classifier output using edges without orientation information: hand in correct pose (red), hand in incorrect pose (blue) and background regions (black).* **(b)** *histogram of classifier output using edges with orientation information. The classes are clearly better separated,* **(c)** *the corresponding ROC curves.*

The corresponding ROC curves are shown in 5.6 (c), demonstrating the benefit of using oriented edges. These improvements of the ROC curves are observed in different amounts for all classifiers and are shown in figure 5.7 (a) and (b).

- In all experiments the best classification results were obtained by the classifier trained on real data. The ROC curves for a particular hand pose (open hand parallel to image plane) are shown in figure 5.7. At a detection rate of 0.99 the false positive rate was below 0.05 in all experiments.

- Marginalized templates showed good results, also yielding low false positive rates at high detection rates. Templates using pixel-wise averaging and negative weights for background edges were found to perform best when comparing the three versions of marginalized templates. For this template the false positive rates were below 0.11 at a detection rate of 0.99.

- Using the centre template with chamfer or Hausdorff matching showed slightly lower classification performance than the other methods, but in all cases the false positive rate was still below 0.21 for detection rates of 0.99. Chamfer matching gave better results than Hausdorff matching, as can be seen in the ROC curve in figure 5.7 (b). It should be noted that this result is in contrast to the observations made by Huttenlocher in [61], where Hausdorff matching is shown to outperform chamfer matching in a Monte-Carlo simulation study. However, this may be explained by



(a) (b)

Figure 5.7: **ROC curves for edge-based classifiers.** *This figure shows the ROC curve for each of the classifiers.* **(a)** *edge features alone, and* **(b)** *oriented edges. Note the difference in scale of the axes. The classifier trained on real image data (P Data) performs best, the marginalized templates (M (a), M(b), M(c)) all show similar results, and chamfer matching (C Cham) is slightly better than Hausdorff matching (C HD) in this experiment. When used within a cascade structure, the performance at high detection rates is important.*

differences in the implementation details. Whereas the L_1 norm and a threshold value of $\tau = 2$ is used in [61], here the L_2 norm and a threshold value of $\tau = 50$ is used. Values of τ in the range of 2–10 were tested in initial experiments and were found to yield worse results. Also the experimental settings in [61] are different, where background edges and occlusions are generated synthetically and the rotation around the z-axis is limited to a range of 20 degrees.

The execution times for different choices of templates $\mathbf{A}$ were compared in order to assess the computational efficiency. Computing the scalar product of two vectors of size 128×128 is relatively expensive. One possibility to reduce the computational cost is to use an approximation to this classifier, as suggested by Huttenlocher *et al.* [63] for Hausdorff matching, by projecting the training edge images into an eigenspace of lower dimension. However, this will come at the cost of a likely reduction in classification performance. If the matrix $\mathbf{A}$ has many zero valued entries, the computational time can be reduced by avoiding the corresponding multiplications. For chamfer and Hausdorff matching, the template only contains the points of a single model projection. The number of points in the marginalized template depends on the size of the parameter space region it represents. In

Classification Method	Number of Points	Execution Time	fp at $tp = 0.99$
Chamfer	400	13 ms	0.10
Hausdorff	400	13 ms	0.12
Marginalized Template	5,800	186 ms	0.02
Binary Marginalized Template	5,800	136 ms	0.02
Trained Classifier Template	16,384	524 ms	0.01

Table 5.1: **Computation times for template correlation.** *The execution times for computing the dot product of 10,000 image patches of size* 128×128*, where only the non-zero coefficients are correlated for efficiency, measured on a 2.4 GHz PC with Pentium IV. The last column shows the false positive (*fp*) rates for each classifier at a fixed true positive (*tp*) rate of 0.99.*

the experiments it contained approximately 14 times as many non-zero points as a single model template. When using a binary template the dot product computation simplifies to additions of coefficients. If both vectors are in binary form, a further speed-up can be achieved by using AND operations. For this the larger vectors can be split up into arrays of 32-bit size, i.e. an integer variable, which are then correlated using an AND operation. The correlation score is then obtained by counting the number of bits set to one. A further speed-up can be obtained by retrieving the number of bits by indexing into a look-up table, however, this has yet not been implemented.

The execution times for correlating 10,000 templates are shown in table 5.1. The time for computing a distance transform or dilation, which needs to be only computed once for each frame when chamfer or Hausdorff matching is used, is less than 2 ms and is therefore negligible when matching a large number of templates.

5.2.3 Discussion

The classifiers can be obtained from a geometric 3D model or from training images. Different classifiers were compared according to their performance and execution time. The results suggest that using a classifier trained on real data is the best option. This is not surprising, as the templates created using the model only approximate the hand appearance, and do not attempt to model arm contours or background edges, for example. However, a drawback of using this classifier is that a large number of template points is needed to evaluate the classifier. Another factor, which makes the use of a model attractive is that training a classifier becomes impractical when the number of poses and views is very large, and thus a larger training set is required. The templates for chamfer or Hausdorff matching, as well as the marginalized templates have the advantage of being easy to gen-

erate and being labelled with 3D pose parameters.

There clearly seems to be a trade-off between computation time and classification performance for the classifiers, as shown in table 5.1. When used in a cascaded structure, the detection rate of a classifier needs to be very high, so as not to miss any true positives. The false positive rate for each classifier at a fixed detection rate of 0.99, is given in the last column of table 5.1. Chamfer and Hausdorff matching, while having a larger false positive rate, are about 10-14 times faster to evaluate than marginalized templates and about 40 times faster than the trained classifier.

One issue that has been left aside for the moment are large scale changes of the template; The chamfer cost of a small template with relatively few edges on a random edge image will be lower than that of a large template [96]. This effect becomes significant when searching over a larger range of scales, and it is treated in more detail in chapter 6.

In the next section the attention is shifted from contour to colour information. In particular it is examined how the fact that the hand interior is skin-coloured can be used in template matching.

5.3 Modelling skin colour

Skin colour in images has a characteristic distribution and this has been exploited by systems for tracking or detecting hands or faces [14, 67, 88, 119]. However, the way in that colour information has been used differs widely, in particular with regard to the choice of a colour space and the representation of the distribution.

Jones and Rehg [74] carried out a thorough study of skin colour models. A data set of 13,640 images was used to obtain distributions of skin colour and non-skin colour values. A Bayesian classifier, treating each pixel independently, was defined using different representations of the distribution. It was shown that a RGB-histogram representation, quantized to 32^3 bins leads to better generalization performance than a histogram with 64^3 or 256^3 bins and it also outperforms a classifier based on a Gaussian mixture model.

Many authors choose to work in colour spaces where the brightness value is separated from the hue, such as the HSV and YUV colour spaces [14, 137]. The intensity value is then either weighted less relative to the hue or discarded completely. Another option is to work in the intensity normalized (r, g)-space, where $r = R/(R + G + B)$ and $g = G/(R + G + B)$. Several studies have indicated that skin tones differ mainly in their brightness values but that their chrominance is relatively constrained among different people [14, 93, 154]. Yang *et al.* [154] suggest modelling the distribution as a Gaussian in (r, g)-space, an advantage of this parametric representation being that it is easy to update

when necessary. Adapting colour models to changes in illumination can yield significant improvements over static models, particularly over very long image sequences [93, 124].

In this work two representations for the colour distributions were used: a Gaussian distribution in normalized (r, g)-space and an RGB-histogram using 32^3 bins. The normalized (r, g)-space introduces invariance to brightness changes, however, it does not compensate for changes in illumination colour. The distribution parameters were estimated from skin colours in the first frame of a sequence, obtained by manual segmentation and no adaptation was performed. The RGB model was constructed by acquiring a skin colour data set of 700,000 RGB vectors taken from one person under different lighting conditions over the time period of one day. Skin areas which were brightness-saturated or very dark due to shadows were not included in the skin colour histogram. In practice, it was found that this model was able to detect skin-colour in a wide range of illumination conditions while rejecting most of the background area in office scenes.

A cost function based on colour values can be constructed in a number of ways. One option is to colour segment the image and subsequently match a silhouette template to this binary feature map. Here a probabilistic approach is taken by deriving a likelihood function $p(\mathbf{z}^{col} \mid \mathbf{x})$ based on the image observation $\mathbf{z}^{col}$, the colour vectors in either (r, g)- or RGB space in the whole image. Given a state vector $\mathbf{x}$, corresponding to a particular hand pose, the pixels in the image are partitioned into a set of locations within the hand silhouette $\{k : k \in S(\mathbf{x})\}$ and within the background region $\{k : k \in \bar{S}(\mathbf{x})\}$ outside the silhouette. If pixel-wise independence is assumed, the likelihood function for the whole image can be factored as

$$p(\mathbf{z}^{col}|\mathbf{x}) = \prod_{k \in S(\mathbf{X})} p^s(I(k)) \prod_{k \in \bar{S}(\mathbf{X})} p^{bg}(I(k)) \tag{5.6}$$

$$= \prod_{k \in S(\mathbf{X})} \frac{p^s(I(k))}{p^{bg}(I(k))} \prod_{k \in S(\mathbf{X}) \cup \bar{S}(\mathbf{X})} p^{bg}(I(k)) \ , \tag{5.7}$$

where $I(k)$ is the colour vector at location k in the image, and p^s and p^{bg} are the skin colour and background colour distributions, respectively. Note that the second term on the right hand side of equation (5.7) is independent of the state vector $\mathbf{x}$. When taking the log-likelihood, this becomes a sum

$$\log p(\mathbf{z}^{col}|\mathbf{x}) = \sum_{k \in S(\mathbf{X})} \log p^s(I(k)) + \sum_{k \in \bar{S}(\mathbf{X})} \log p^{bg}(I(k)) \tag{5.8}$$

$$= \sum_{k \in S(\mathbf{X})} \big(\log p^s(I(k)) - \log p^{bg}(I(k)) \big) + \sum_{k \in S(\mathbf{X}) \cup \bar{S}(\mathbf{X})} \log p^{bg}(I(k)). \tag{5.9}$$

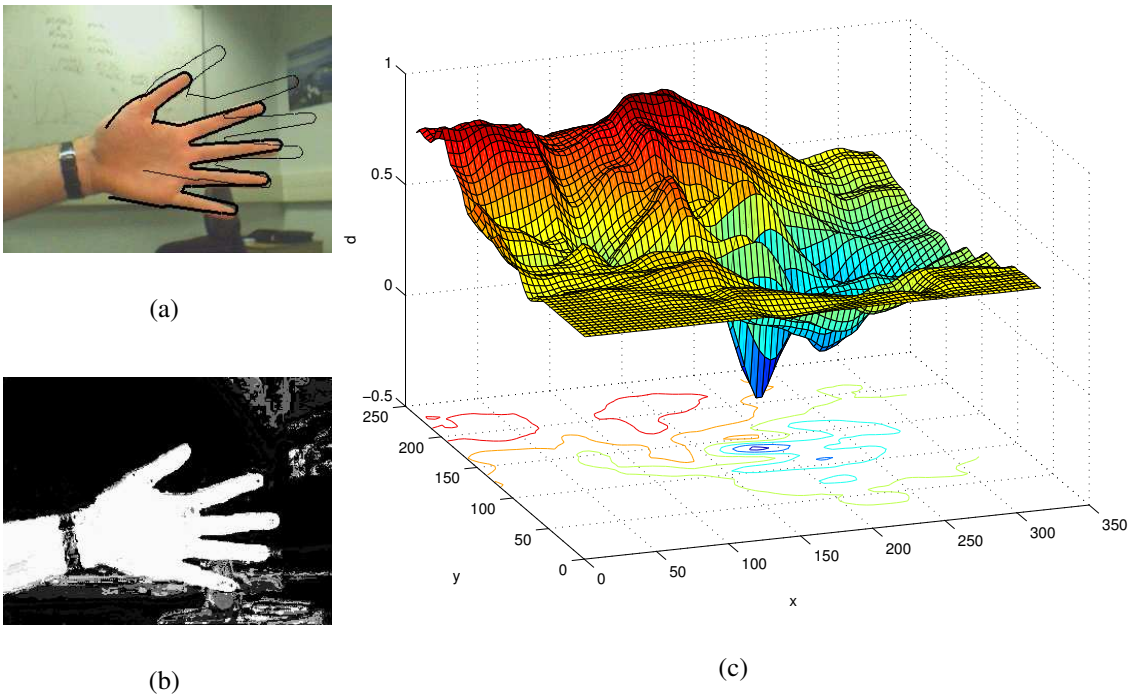


Figure 5.8: **Colour likelihood in translation space.** *This figure shows* **(a)** *an input image with two templates superimposed, corresponding to the global optimum and a non-global local minimum,* **(b)** *the log-likelihood image encoded as a greyscale image and* **(c)** *the surface of the colour cost (negative log-likelihood) generated by translating the correct template over the image.*

Figure 5.8 (c) shows the cost surface in terms of negative log-likelihood, when translating a template over the same input image as in figure 5.2. Because the cost is computed by integrating over an area, the cost varies continuously with small changes in translation, so that in contrast to edge features, it is not necessary to smooth the features.

The independence assumption in equation (5.6) is clearly a simplification; the colour values of neighbouring pixels are not independent. Jedynak *et al.* [72] suggest using a first order model by considering a pixel neighbourhood, and show that this slightly improves skin colour classification, however at a tenfold increase in computational cost. A different way to circumvent the problem is to apply filters with local support at points on a regular grid on the image. The filter outputs are then approximately independent [69, 129]. However the pixel-wise independence assumption gives reasonable approximations and leads to a form of the likelihood function that is efficient to evaluate.

5.3.1 Comparison of classifiers

Experiments analogous to the ones for shape classifiers in section 5.2.1 were carried out. In this section, however, the linear classifiers are based on the silhouette area in order to evaluate the colour likelihood of an image subwindow. Using subwindows instead of the complete image limits the effect of background pixels that are skin coloured. They also make it straightforward to detect the presence of more than one hand in the scene. As in section 5.2.1, one such classifier has a number of hand shapes as the target class, corresponding to a range within the parameter space.

Given an input image region, define the feature matrix $\mathbf{B}^s$ as the log-likelihood map of skin colour and $\mathbf{B}^{bg}$ as the log-likelihood map of background colour. Skin colour is represented as a Gaussian in (r, g)-space and the background distribution is modelled as a uniform distribution. A silhouette template $\mathbf{A}$ is defined as containing +1 at locations within the hand silhouette and 0 otherwise. Writing these matrices as vectors, analogous to section 5.2, the log-likelihood in equation (5.8) can be written as:

$$\log p(\mathbf{z}^{col}|\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\mathbf{b}^s + (\mathbf{1}-\mathbf{a})^{\mathrm{T}}\mathbf{b}^{bg} \quad (5.10)$$

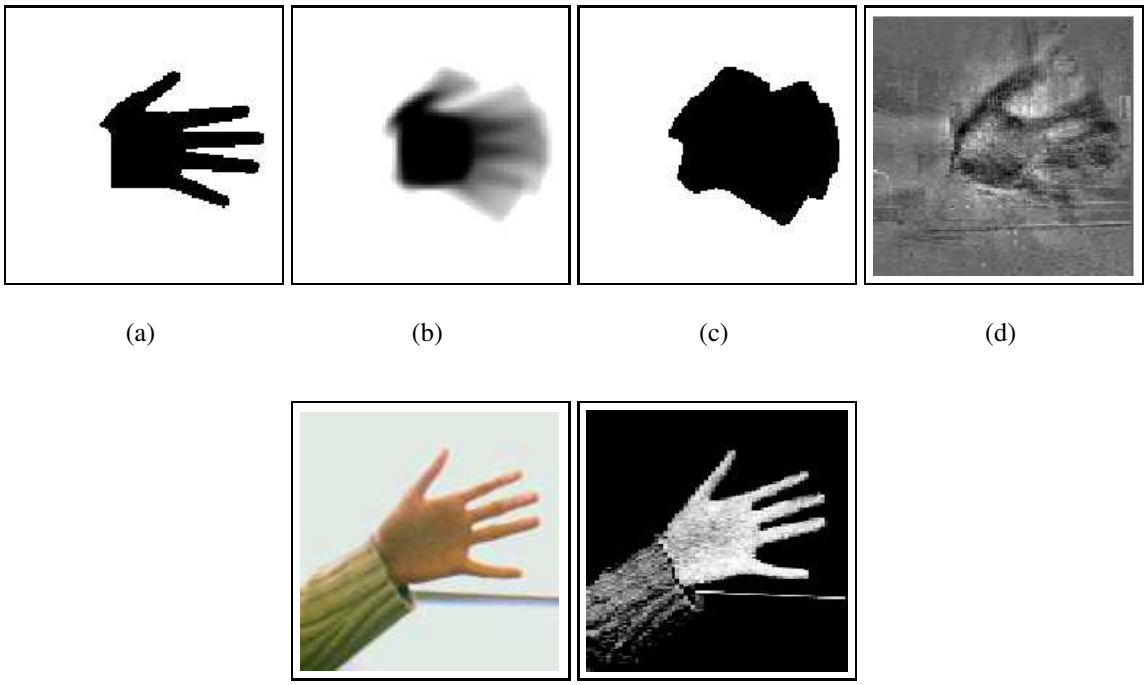
$$= \mathbf{a}^{\mathrm{T}}(\mathbf{b}^s - \mathbf{b}^{bg}) + \mathbf{1}^{\mathrm{T}}\mathbf{b}^{bg} \ , \quad (5.11)$$

where $\mathbf{1} = [1, \ldots, 1]^{\mathrm{T}}$ is the one-vector and the entries in $\mathbf{b}^{bg}$ are constant when using a uniform background model. Thus, correlating the matrix $\mathbf{B} = \mathbf{B}^s - \mathbf{B}^{bg}$ with a silhouette template $\mathbf{A}$ corresponds to evaluating the log-likelihood of an input region up to an additive constant. Note that when the distributions are fixed, the log-likelihood values for each colour vector can be pre-computed and stored in a look-up table beforehand. A number of different classifiers were compared (illustrated in figure 5.9, top row).

- **Centre template [Figure 5.9 (a)].** This classifier corresponds to using a silhouette template $\mathbf{A}$, generated at the centre of a region in parameter space. The values of $\mathbf{A}$ within the silhouette area are set to one, the values in the background to zero. This corresponds to evaluating the log-likelihood of the centre template according to equation (5.11).

- **Marginalized template [Figure 5.9 (b), (c)].** This template is generated by densely sampling the values in a region of parameter space and generating the silhouettes for each state. The resulting model projections are then pixel-wise added and different versions of matrices $\mathbf{A}$ are compared:

  (a) the pixel-wise average of model projections, see figure 5.9 (b),



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

(e) (f)

Figure 5.9: **Templates and feature maps used for colour-based classification. Top row:** *Templates* **A** *used for classifying colour features.* **(a)** *single template,* **(b)** *marginalized template created by averaging over region in parameter space,* **(c)** *template learnt from real image data.* **Bottom row:** *Extracted features from an image.* **(d)** *input image,* **(e)** *log-likelihood image of skin colour, represented as greyscale image.*

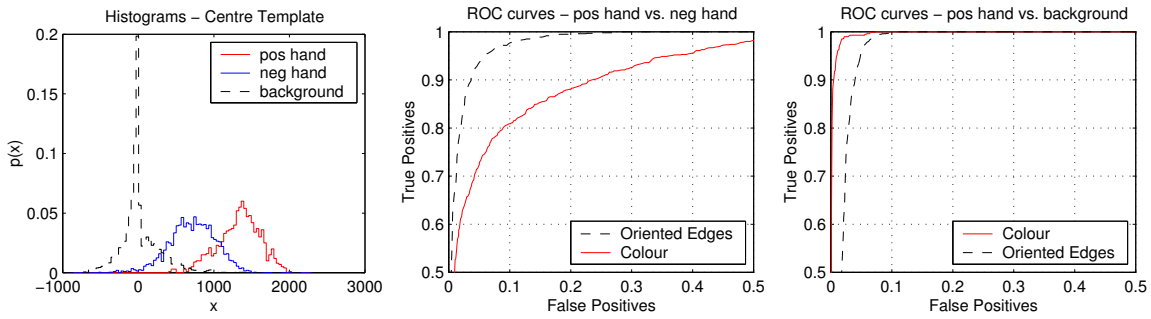
(b) the union of all projections, resulting in a binary template, see figure 5.9 (c).

- **Linear classifier learnt from image data [Figure 5.9 (d)].** The template **A** is obtained by learning a classifier from training data. The same training set as in section 5.2.1 (see figure 5.5) is used to train a linear classifier using the skin colour log-likelihood map as training data.

5.3.2 Experimental results

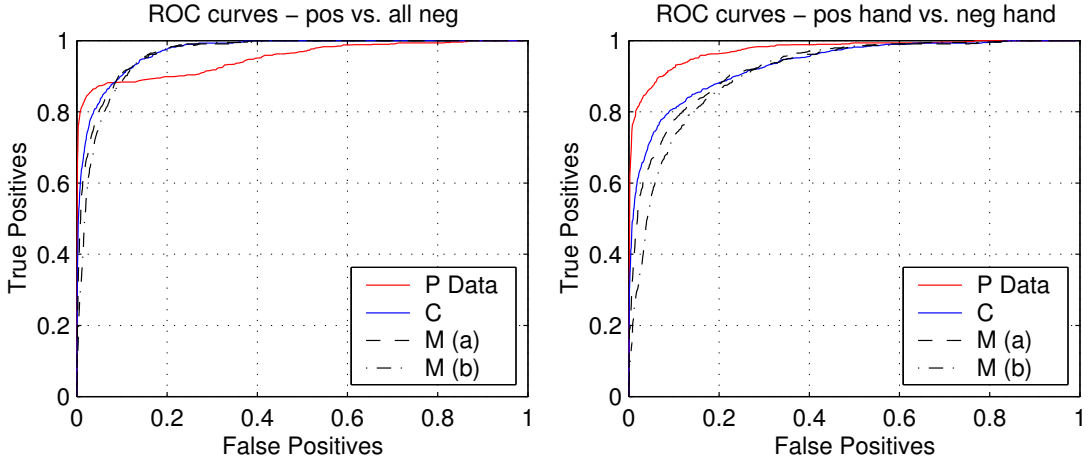
The same test data set as in section 5.2.1 is used throughout the experiments, and the following observations were made:

- For the images in the test set colour information helps to discriminate between positive examples of hands and background regions. However, there is significant overlap between the positive and negative class examples which contain a hand.



(a) (b) (c)

Figure 5.10: **Colour information helps to exclude background regions.** *This example shows the classification results on a test set using a centre template.* **(a)** *histogram of classifier output using a skin colour log-likelihood hand in correct pose (red), hand in incorrect pose (blue) and background regions (black).* **(b)** *ROC curve showing that oriented edges are better than colour features for discriminating between the hand in the correct pose and the hand in incorrect pose.* **(c)** *ROC curve showing that colour features are better than oriented edges for discriminating between the hand in the correct pose and background regions.*



(a) (b)

Figure 5.11: **ROC curves for colour-based classifiers.** *This figure shows the ROC curves for each of the classifiers.* **(a)** *The classifiers based on the centre (C) and marginalized templates (M (a), M(b)) have similar performance, whereas the trained classifier (P Data) does not perform as well in regions of high detection rates.* **(a)** *The ROC curves only for classifying positive examples and negative example images containing a hand, where the trained classifier (P Data) performs better.*

This is demonstrated in figure 5.10 (a), which shows a histogram of the three class distributions for the case of the marginalized template. Oriented edges are better features to discriminate between the hand in different poses (figure 5.10 (b)), whereas colour features are slightly better at discriminating between the positive class and background regions (figure 5.10 (c)).

- Both centre template and marginalized templates show better classification performance than the trained classifier, in particular in the high detection range, see figure 5.11 (a). At detection rates of 0.99 the false positive rate for the centre template is 0.24, wheras it is 0.64 for the trained classifier. One explanation for this is the difference in lighting conditions under which the training and test sets were acquired, leading to template coefficients for background regions, which do not generalize well. Working in the normalized (r, g)-space only gives limited variance towards illumination changes. However, the trained classifier shows better performance at separating positive examples from negative example images containing hands, see figure 5.11 (b). At a detection rate of 0.99, the false positive rate is 0.41 compared to 0.56 for the other two classifiers.

- There is not much difference between the performance of marginalized templates and the centre template. The reason for this could be that the area difference between them is relatively small compared to the corresponding types of edge templates.

The methods were also compared in terms of computation time. The time measured for computing 10,000 correlations of templates of size 128×128 is 524 ms (see table 5.1). However, for binary templates the evaluation can be performed efficiently by pre-computing a sum table, $\mathbf{B}^{sum}$, which has the same size as the image and contains the cumulative sums along the x-direction:

$$\mathbf{B}^{sum}(x, y) = \sum_{i=1}^{x} \left(\log p^s(I(i, y)) - \log p^{bg}(I(i, y)) \right) \quad , \tag{5.12}$$

where in this equation the image I is indexed by its x and y-coordinates. This array only needs to be computed once and is then used to compute sums over areas by adding and subtracting values at points on the silhouette contour, see figure 5.12. This is a convenient technique to convert area integrals into contour integrals. As such it is related to the *summed area tables* of Crow [33], the *integral images* of Viola and Jones [141], and the integration method based on Green's theorem of Jermyn and Ishikawa [73]. Compared to

(a) (b)

Figure 5.12: **Efficient evaluation of colour likelihoods. (a)** *The skin colour log-likelihood image encoded as a greyscale image. Higher intensity corresponds to higher likelihood of skin vs. non-skin colour.* **(b)** *The sum table contains the cumulative sum of values in image (a) along the x-direction. The sum of values in within an area can be efficiently computed by adding and subtracting values at silhouette points only. The greyscale intensities are scaled to the range [0,255] in both images.*

integral images, the sum over non-rectangular regions can be computed more efficiently, and in contrast to the technique of Jermyn and Ishikawa the contour normals are not needed. In contrast to these methods, however, the model points need to be connected pixels, e.g. obtained by line-scanning a silhouette image. The computation time for evaluating 10,000 templates is reduced from 524 ms to 13 ms for a silhouette template of 400 points. As the computation of the sum table $\mathbf{B}^{sum}$ is only computed once, this is negligible when matching a large number of templates.

5.4 Combining edge and colour information

In the previous sections similarity measures based on edge and colour information have been derived. When combining the two features, they can be stacked into a single 2D observation vector $\mathbf{z} = (\mathbf{z}^{edge}, \mathbf{z}^{col})^{\mathrm{T}}$. In a first approach, the edge and colour cost terms are computed for a number of test images. Figure 5.13 shows a graph of the distributions of the cost vectors for 1000 positive and 1000 negative example image subregions. Positive examples correspond to a hand being in a pose corresponding to the parameter range represented by the classifier. Negative examples correspond to the hand in different poses and background regions. The plot shows that for the test images that the class overlap is not large and a linear discriminant is used to separate the two classes, yielding

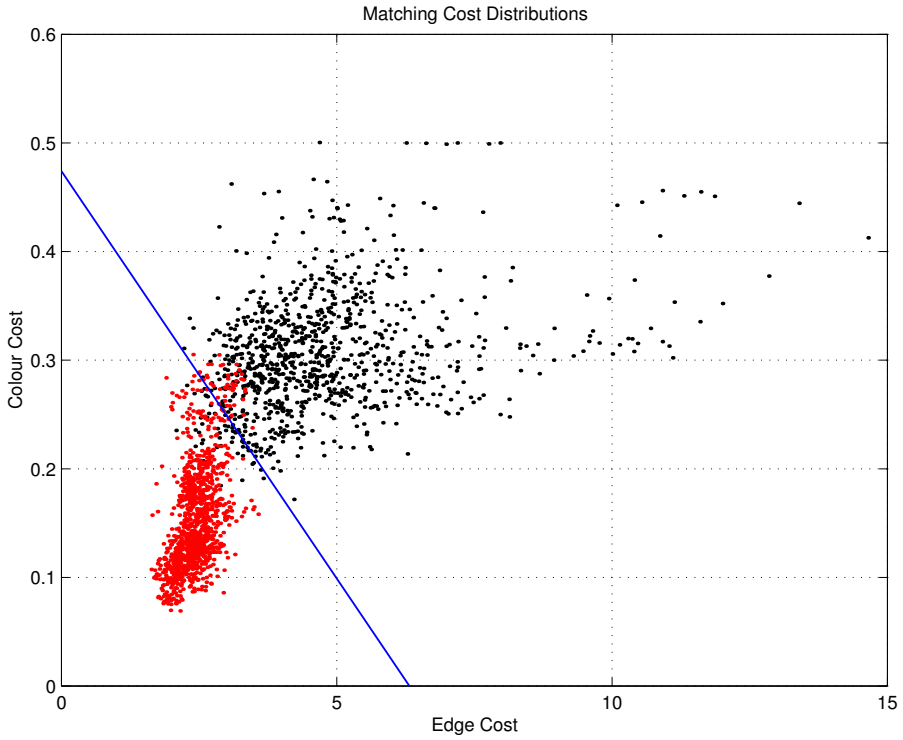


Figure 5.13: **Determining the weighting factor in the cost function.** *The distribution of edge and colour cost values for a number of positive (lower left) and negative training examples (upper right) is shown, and the linear classifier found using a maximum margin linear classifier. The weighting factor is set to the negative inverse of the slope of this line.*

a total misclassification rate of 4.8% on this data set. Using a linear discriminant corresponds to simply using a weighted sum of the edge and colour cost terms [14], where the weighting factor is derived from the data directly in order to yield optimal classification performance on a test set. Experiments for three different classifiers, corresponding to the hand in different poses, but at the same scale, show that this weighting factor varies little for different templates. A method to combine the two features within a probabilistic framework is presented in chapter 6.

5.4.1 Experimental results

In order to test the integration of shape and colour information a set of 500 templates was generated, corresponding to 100 discrete orientations and five different scales. The templates are matched to an image by translating it over the image at a 6-pixel resolution in x and y-direction. The weighted cost function was used to detect a hand in the following scenarios:

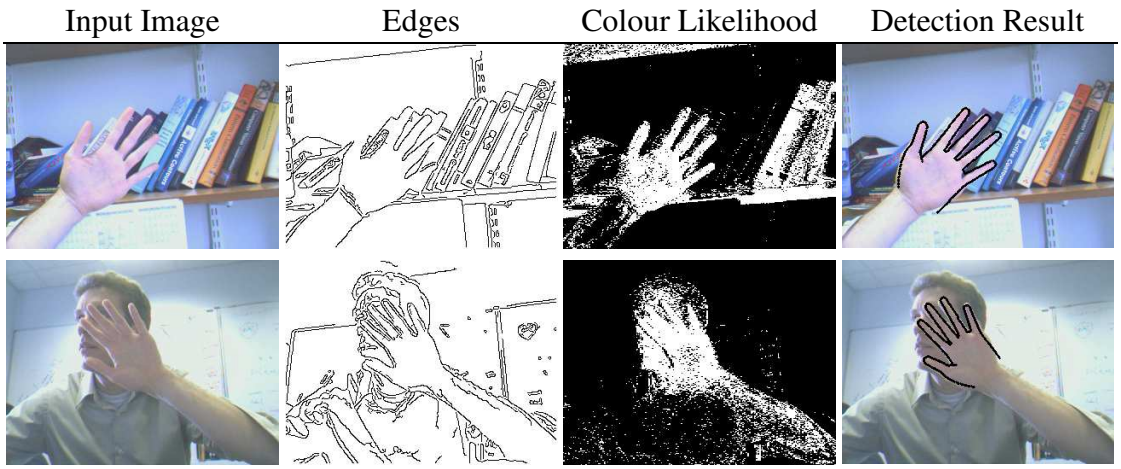


Figure 5.14: **Detection with integrated edge and colour features.** *This illustrative example shows how a cost function that uses edge and colour features improves detection. For each input image the best match is shown in the last column.* **(Top row)** *Hand in front of cluttered background, and* **(bottom row)** *hand in front of face.*

- A hand in front of cluttered background, which contains little skin colour: In this case the hand is difficult to detect using edge information alone. The colour likelihood, however allows for correct detection (See top row of figure 5.14).

- A hand in front of a face: In this example there is not enough skin colour information to detect the hand, however the hand edges are still visible and are used as features to correctly locate the hand (See bottom row of figure 5.14). The method of extracting skin colour edges (section 4.3) clearly fails in this example.

This illustrative example shows that using a robust cost function can improve the performance of hand detection or tracking algorithms that use intensity or skin colour edges alone.

In the second experiment, an input sequence of 640 frames was recorded. The pose is an open hand, parallel to the image plane, and it moved with 4 DOF; translation in x, y, and z-direction, as well as rotation around the z-axis. The task is made challenging by introducing a cluttered background with skin-coloured objects. The hand motion is fast, and during the sequence the hand is partially and fully occluded, as well as out of the camera view. The same set of 500 templates is used to search for the best match over the image. No dynamic model is used in this sequence, because the hand leaves and re-enters the camera view several times.

Figure 5.15 shows typical results for a number of frames of this sequence and figure 5.16 shows the position error measured against manually labelled ground truth. The RMS error over the complete sequence for the frames in which the hand was detected, was 3.7 pixels. The main reason for the magnitude of this error is the coarse search at 6-pixel resolution in translation space. Assuming a uniform distribution on the hand location in the image, the expected RMS error is larger than 1.5 pixels. The UKF algorithm from chapter 4 was also run on this input sequence, but it was not able to track the hand for more than 20 frames. The main reason is that neither intensity nor colour edge features alone are robust enough for this experiment. Another reason for the loss of track is that the motion in this sequence is fast and abrupt, so that neither a constant velocity nor a constant acceleration model is accurate.

5.5 Summary

When tracking hands in an unconstrained environment, it is important to design robust cost functions. Integrating edge and colour information leads to robust similarity measures, as the features complement each other. For example, colour is very useful when edge information is unreliable due to fast hand motion. On the other hand, edge information allows accurate matching when the hand is in front of skin coloured background.

In this chapter a number of different similarity measures for template matching with shape variation were considered. If a training set of shapes is available, a linear classifier can be trained which has high discriminative power. It can give different weights to different parts of the shape and is able to penalize background edges. However, it is also expensive to evaluate. Marginalized templates have been introduced as a method to use an object model to construct a classifier, which is specifically "tuned" to a specific region of parameter space. They can be efficiently evaluated, for example by approximating them with a binary valued template. Finally, templates generated from a single contour ("centre templates" in the experiments), can be used to classify shapes. However it is necessary to pre-process the edge image first, e.g. by a dilation operation or a distance transform, in order to be tolerant to shape variation. The main advantage of this approach is that it is very efficient because the pre-processing step is only required once per image and matching a single template is fast.

The colour cost term is based on independent pixel likelihoods and can be evaluated using silhouette area templates. In most cases skin colour has a characteristic distribution, which is distinct from most background regions. This chapter made a number of simplifying assumptions in deriving the colour likelihood, including the independence of pixel

likelihoods, a uniform background colour model, as well as limited illumination changes. It was found that a representation using a quantized RGB histogram obtained from a images taken under a variety of illumination conditions was still able to discriminate between skin-colour and most of the background regions. Finally, by using pre-computed sum tables the evaluation cost of the colour likelihood term is proportional to the length of the silhouette contour instead of its area.

The detection results in this chapter were demonstrated for a single hand pose only and it remains to be shown that the method works for arbitrary hand poses. In the following chapter many ideas that have been developed up to here will be coming together to build a hand tracking system. The main idea is to use the robust cost function developed in this chapter, where it was used for detection at each frame and apply it within an efficient temporal filtering framework.

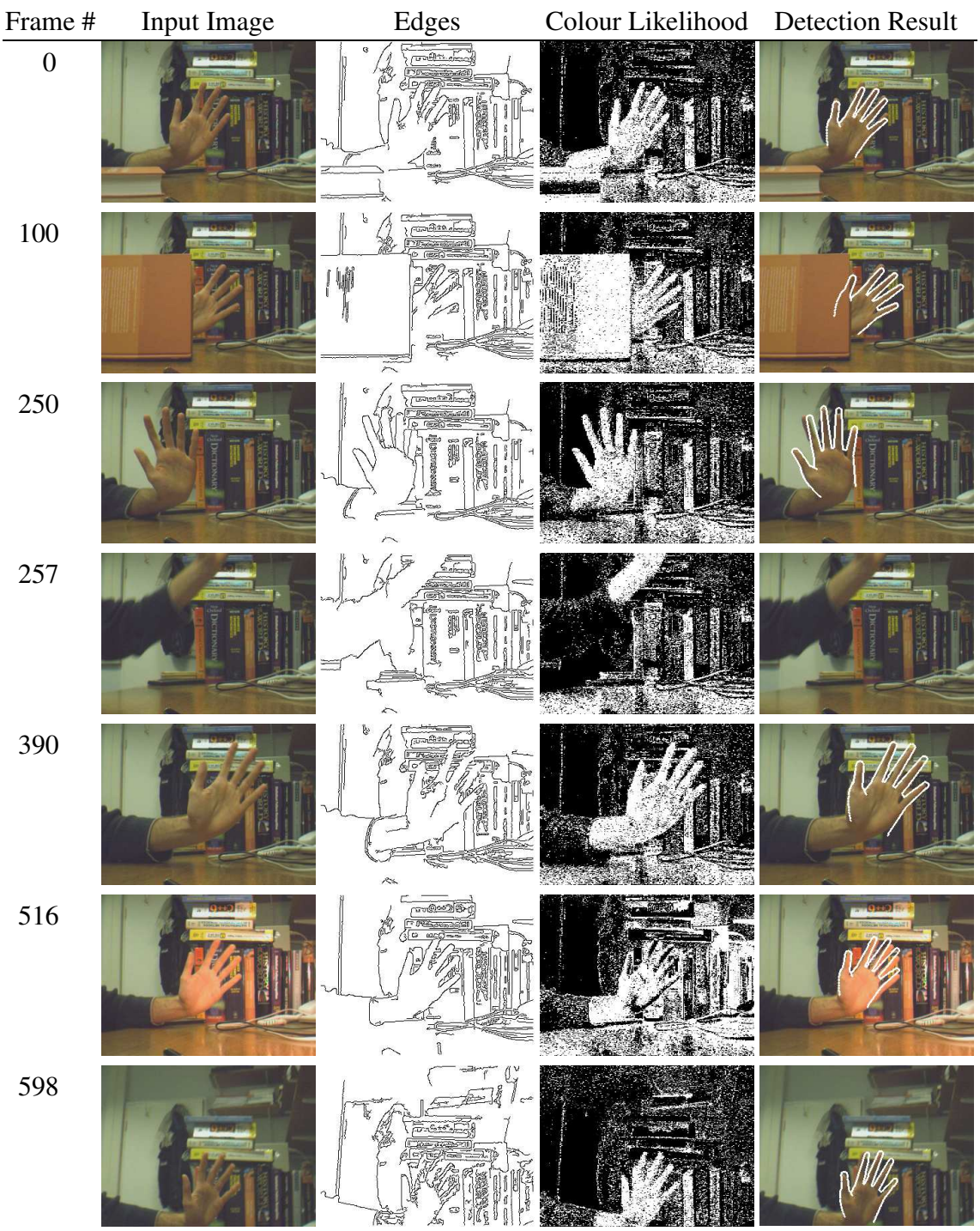


Figure 5.15: **Detection results using edge and colour information.** *This figure shows successful detection of an open hand moving with 4 DOF. The first two columns show the frame number and the input frame, the next two columns show the Canny edge map and the skin colour likelihood. The last column shows the best match superimposed, if the likelihood function is above a constant threshold. The sequence is challenging because the background contains skin-coloured objects* **(frame 0)** *and motion is fast, leading to motion blur and missed edges. The detection handles some partial occlusion* **(frame 100)***, recovers from loss of track* **(frames 257, 390)***, can deal with lighting changes* **(frame 516)** *and unsteady camera motion* **(frame 598)**.

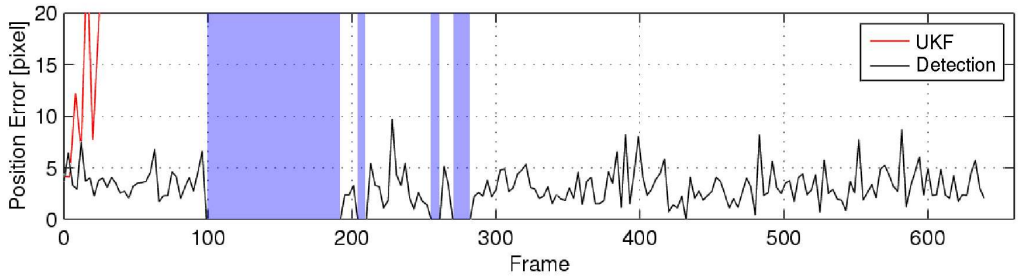


Figure 5.16: **Error comparison between UKF tracking and detection.** *This figure shows the error performance of the UKF tracker and detection on the image sequence in figure 5.15. The hand position error was measured against manually labelled ground truth. The shaded areas (blue) indicate intervals in which the hand is either fully occluded or out of camera view. The detection algortihm successfully finds the hand in the whole sequence, whereas the UKF tracker using skin-colour edges is only able to track the hand for a few frames. The reasons for the loss of track is that the hand motion is fast between two frames and that skin-colour edges cannot be reliably found in this input sequence.*

6 Hand Tracking Using A Hierarchical Filter

He who plants a tree plants a hope.

Lucy Larcom (1824–1893)

The task of robust tracking demands a strategy for detecting the object in the first frame, as well as finding the object when track is lost. This is particularly important for HCI applications, where hand motion can be fast and it is difficult to accurately predict the motion.

A second challenge is that there can be a lot of ambiguity due to self-occlusion when using a single input image. Several different hand poses can give rise to the same appearance, for example when some fingers are occluded by other fingers. In such cases dynamic information can help to resolve the ambiguity.

This chapter addresses both issues and proposes an algorithm for Bayesian tracking, which is based on a multi-resolution partitioning of the state space. It is motivated by methods introduced in the context of hierarchical detection, which are briefly outlined in the next section.

6.1 Tree-based detection

Methods for detecting objects are becoming increasingly efficient. Examples are real-time face detection or pedestrian detection [47, 86, 141]. However, applying these techniques to hand detection from a single image is difficult because of the large variation in shape and appearance of a hand in different poses. In this case detection and pose estimation are tightly coupled. One approach to solving this problem is to use a large number of

shape templates and find the best match in the image. In *exemplar-based* methods, such templates are obtained directly from the training sets. This has the advantage that no parametric model needs to be constructed [18, 47, 135]. For example, Gavrila uses approximately 4,500 shape templates to detect pedestrians in images [47]. To avoid exhaustive search, a template hierarchy is formed by bottom-up clustering based on the chamfer distance (see section 5.1). A number of similar shape templates are represented by a cluster prototype. This prototype is first compared to the input image, and only if the error is below a threshold value, the templates within the cluster are compared to the image. The use of a template hierarchy is reported to result in a speed-up of three orders of magnitude compared to exhaustive matching [47].

If a parametric object model is available, another option to build such a tree of templates is by partitioning the state space. Let this tree have L levels, each level l defines a partition $\mathcal{P}_l$ of the state space into N_l distinct sets $l = 1, \ldots, L$, such that $\mathcal{P}_l = \{\mathcal{S}^{il}\}_{i=1}^{N_l}$. The leaves of the tree define the finest partition of the state space $\mathcal{P}_L = \{\mathcal{S}^{iL}\}_{i=1}^{N_L}$. The use of a parametric model also allows to combine a template hierarchy created off-line with an on-line optimization process. Once the leaf level is reached, the template match can be refined by continuous optimization of the model parameters. In [131] we used this method to adapt the shape of a generic hand model to an individual user in the first frame.

A draw-back of a single-frame detector, such as the one presented in [47], is the difficulty to incorporate temporal constraints. Toyama and Blake [135] proposed a probabilistic framework for exemplar-based matching, which allows the integration of a dynamic model. Temporal information is useful, firstly to resolve ambiguous situations, and secondly, to smooth the motion. However, in [135] no template hierarchy is formed. The following section introduces an algorithm which combines the efficiency of hierarchical methods with Bayesian filtering.

6.2 Tree-based filtering

Tracking has been formulated as a Bayesian inference problem in section 4.1, leading to recursive prediction and update equations:

$$p(\mathcal{X}_t|\mathbf{z}_{1:t-1}) = \int p(\mathcal{X}_t|\mathcal{X}_{t-1})\; p(\mathcal{X}_{t-1}|\mathbf{z}_{1:t-1})\; d\mathcal{X}_{t-1} \tag{6.1}$$

$$p(\mathcal{X}_t|\mathbf{z}_{1:t}) = c_t^{-1}\; p(\mathbf{z}_t|\mathcal{X}_t)\; p(\mathcal{X}_t|\mathbf{z}_{1:t-1}) \tag{6.2}$$

$$c_t = \int p(\mathbf{z}_t|\mathcal{X}_t)\; p(\mathcal{X}_t|\mathbf{z}_{1:t-1})\; d\mathcal{X}_t \;\; . \tag{6.3}$$

In the general case it is not possible to obtain analytic solutions for these equations, but there exist a number of approximation methods which can be used to obtain a numerical solution [12, 40, 66, 126].

An important issue in each approach is how to represent the prior and posterior distributions in the filtering equations. One suggestion, introduced by Bucy and Senne [24], is to use a point-mass representation on a uniform grid. The grid is defined by a discrete set of points in state space, and is used to approximate the integrals in the filtering equations by replacing continuous integrals with Riemann sums over finite regions. The distributions are approximated as piecewise constant over these regions. The underlying assumption of this method is that the posterior distribution is band-limited, so that there are no singularities or large oscillations between the points on the grid. Typically grid-based filters have been applied using an evenly spaced grid and the evaluation is thus exponentially expensive as the dimension of the state space increases [12, 24]. Bucy and Senne suggest modelling each mode of the distribution by a separate adapting grid, and they devise a scheme for creating and deleting local grids on-line. A different approach is taken by Bergman [12], who uses a fixed grid, but avoids the evaluation at grid points where the probability mass is below a threshold value. The grid resolution is adapted in order to keep the number of computations within pre-defined bounds.

The aim in this chapter is to design an algorithm that can take advantage of the efficiency of tree-based search to efficiently compute an approximation to the optimal Bayesian solution using a grid-based filter.

In the following, assume that the values of the state vectors $\mathbf{x}$ are within a compact region $\mathcal{R}$ of the state space. In the case of hand tracking, this corresponds to the fact that the parameter values are bounded, the boundary values being defined by the valid range of motion. Define a multi-resolution partition of the region $\mathcal{R}$ as described in section 6.1 by dividing the region $\mathcal{R}$ at each tree-level l into N_l distinct sets $\{\mathcal{S}^{il}\}_{i=1}^{N_l}$, which are non-overlapping and mutually exclusive, that is

$$\bigcup_{i=1}^{N_l} \mathcal{S}^{il} = \mathcal{R} \quad \text{for } l = 1, \ldots, L \ . \tag{6.4}$$

A graphical depiction is shown in figure 6.1. The posterior distribution is represented as piecewise constant over these sets, the distribution at the leaf level being the representation at the highest resolution. Even though the complexity of this method will grow exponentially in the state space dimension, the idea is to use the process of hierarchical detection to rapidly identify regions in state space with low probability mass.

To formalize the discrete filtering problem, define Θ_t^{il} as the parameter values in state

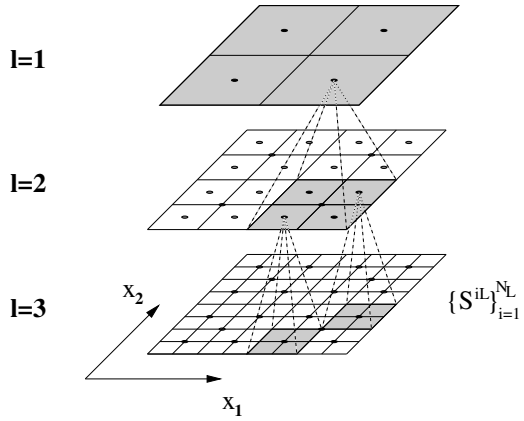


Figure 6.1: **Hierarchical partitioning of the state space.** *The state space is partitioned using a multi-resolution grid. The regions $\{\mathcal{S}^{iL}\}_{i=1}^{N_L}$ at the leaf level define the finest partition, over which the filtering distributions are approximated as piecewise constant. The number of regions is exponential in the state dimension. However, if large regions of parameter space have negligible probability mass, these can be identified early, achieving reduction in computational complexity.*

space integrated over the region $\mathcal{S}^{il}$ at time t, i.e.

$$p(\Theta_t^{il}) = \int_{\mathcal{X}_t \in \mathcal{S}^{il}} p(\mathcal{X}_t)\, d\mathcal{X}_t. \tag{6.5}$$

The initial prior distribution for the discrete states $p(\Theta_1^{iL}|\mathbf{z}_1)$ can be obtained by integration from $p(\mathcal{X}_1|\mathbf{z}_1)$ as

$$p(\Theta_1^{iL}|\mathbf{z}_1) = \int_{\mathcal{X}_1 \in \mathcal{S}^{iL}} p(\mathcal{X}_1|\mathbf{z}_1) d\mathcal{X}_1. \tag{6.6}$$

Next the discrete recursive relations are obtained from the continuous case by integrating over regions. Given the distribution over the leaves of the tree, $p(\Theta_{t-1}^{iL}|\mathbf{z}_{1:t-1})$, at the previous time step $t-1$, the prediction equation now becomes a transition between discrete regions in state space:

$$p(\Theta_t^{jl}|\mathbf{z}_{1:t-1}) = \sum_{i=1}^{N_L} p(\Theta_t^{jl}|\Theta_{t-1}^{iL})\; p(\Theta_{t-1}^{iL}|\mathbf{z}_{1:t-1}). \tag{6.7}$$

Given the state transition distribution $p(\mathcal{X}_t|\mathcal{X}_{t-1})$ the transition probability between regions is given by

$$p(\Theta_t^{jl}|\Theta_{t-1}^{iL}) = \int_{\mathcal{X}_t \in \mathcal{S}^{jl}} \int_{\mathcal{X}_{t-1} \in \mathcal{S}^{iL}} p(\mathcal{X}_t|\mathcal{X}_{t-1})\; d\mathcal{X}_t\, d\mathcal{X}_{t-1}\;. \tag{6.8}$$

Thus the transition distribution is approximated by transition probabilities between discrete regions in state space, see figure 6.2 (a).

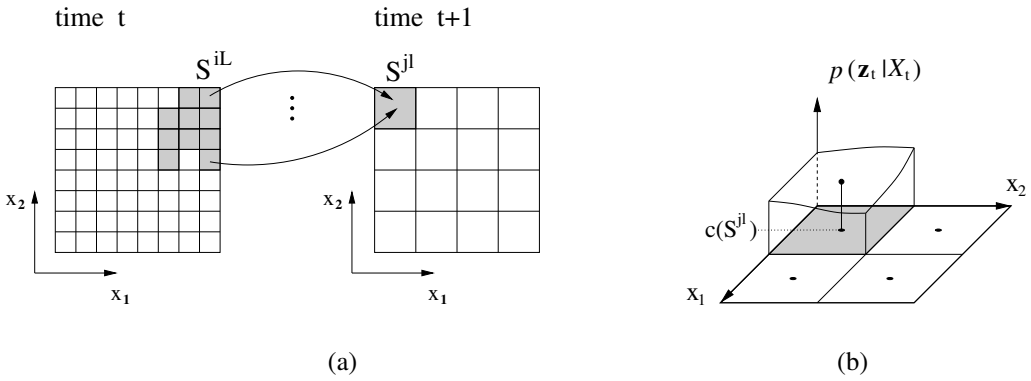


Figure 6.2: **Discretizing the filtering equations. (a)** *The transition distributions are approximated by transition probabilities between discrete regions in state space, which can be modelled by a Markov transition matrix.* **(b)** *The likelihood function is evaluated at the centre $c(\mathcal{S}^{jl})$ of each region, assuming that the function is locally smooth.*

In order to evaluate the distribution $p(\Theta_t^{jl}|\mathbf{z}_{1:t})$, the likelihood $p(\mathbf{z}_t|\Theta_t^{jl})$ needs to be evaluated for a region in parameter space. This is computed by evaluating the likelihood function at a single point, taken to be the centre of the region $c(\mathcal{S}^{jl})$.

This approximation assumes that the likelihood function in the region $\mathcal{S}^{jl}$ can be represented by the value at the centre location $c(\mathcal{S}^{jl})$ (see figure 6.2 (b)):

$$p(\mathbf{z}_t|\Theta_t^{jl}) = \int_{\mathcal{X}_t \in \mathcal{S}^{jl}} p(\mathbf{z}_t|\mathcal{X}_t)\, d\mathcal{X}_t \tag{6.9}$$

$$\approx \underbrace{\int_{\mathcal{X}_t \in \mathcal{S}^{jl}} 1\, d\mathcal{X}_t}_{vol(\mathcal{S}^{jl})} \quad p(\mathbf{z}_t|c(\mathcal{S}^{jl})) \ . \tag{6.10}$$

This is similar to the idea of using a cluster prototype to represent similar shapes. In equation (6.9) the likelihood function has to take the shape variation within the region $\mathcal{S}^{jl}$ into account. Having laid out Bayesian filtering over discrete states, the question arises how to combine the theory with the efficient tree-based algorithm previously described. The idea is to approximate the posterior distribution by evaluating the filter equations at each level of the tree. In a breadth-first traversal regions with low probability mass are identified and not further investigated at the next level of the tree. Regions with high posterior are explored further in the next level (Figure 6.3). It is to be expected that the higher levels of the tree will not yield accurate approximations to the posterior, but are used to discard inadequate hypotheses, for which the posterior of the set is below a threshold value. The following scheme is used to set the thresholds: For each level the

maximum and minimum of the posterior values

$$\hat{p}_t^{l,max} = \max_{j=1,\ldots,N^l} p(\Theta_t^{jl}|\mathbf{z}_{1:t}) \quad \text{and} \quad \hat{p}_t^{l,min} = \min_{j=1,\ldots,N^l} p(\Theta_t^{jl}|\mathbf{z}_{1:t}) \tag{6.11}$$

is used to compute the threshold as

$$\tau_t^l = \hat{p}_t^{l,min} + c_\tau \ (\hat{p}_t^{l,max} - \hat{p}_t^{l,min}) \ , \quad \text{where } c_\tau \in [0,1]. \tag{6.12}$$

The search proceeds only in the subtrees of nodes with a posterior value larger than τ_t^l. After each time step t the posterior distribution is represented by the piecewise constant distribution over the regions at the leaf level.

In order to determine whether the hand is still in the scene, it is checked if the variance of the normalized posterior values at the leaf level is below a certain threshold value τ_{detect}.

$$\frac{1}{N_L - 1} \sum_{j=1}^{N_L} \left(p(\Theta_t^{jL}|\mathbf{z}_{1:t}) - \hat{p}_t^{L,mean} \right)^2 < \tau_{detect} \tag{6.13}$$

where

$$\hat{p}_t^{L,mean} = \frac{1}{N_L} \sum_{j=1}^{N_L} p(\Theta_t^{jL}|\mathbf{z}_{1:t}) \ . \tag{6.14}$$

When a hand is in the scene, this leads to a distribution where there is at least one strong peak, whereas in background scenes the values do not vary as much. In this case no dynamic information is used and only the likelihood values are computed, as in the initialization step. An overview of the algorithm is given in Algorithm 6.1.

6.2.1 The relation to particle filtering

This section comments on similarities and differences between this method and Monte-Carlo based approaches for evaluating the Bayesian filtering equations; in particular particle filters. These are based on sequential importance sampling with resampling [40, 52, 66]. Particle filters have been applied in a number of fields, including computer vision, robotics and signal processing [15, 40, 66, 133]. One of their main attractions is their convergence property: The convergence rate of the approximation error is independent of the dimension of the state space. This is in contrast to grid-based methods, such as the one suggested here. However, a number of issues are important when applying particle filters. Their performance depends largely on the *proposal distribution* from which particles are sampled in each time step [40]. It is essential to sample particles in regions in state space, which have high likelihood values. If additional knowledge is available in form of an *importance distribution*, e.g. a different sensor input or knowledge about

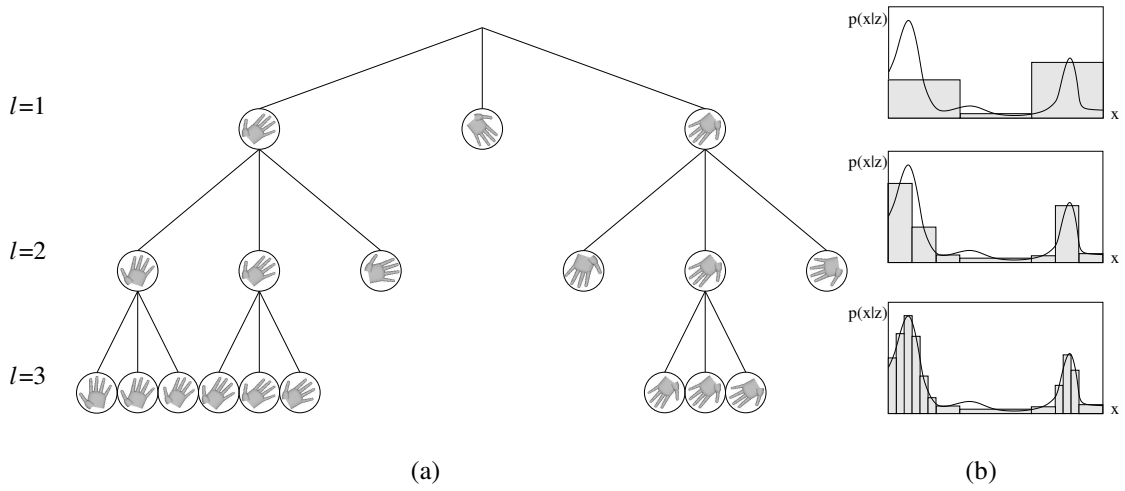


Figure 6.3: **Tree-based estimation of the posterior density.** *(a) Associated with the nodes at each level is a non-overlapping set in the state space, defining a partition of the state space. The posterior for each node is evaluated using the center of each set, depicted by a hand rotated by a specific angle. Sub-trees of nodes with low posterior are not further evaluated. (b) Corresponding posterior density (continuous) and the piecewise constant approximation using tree-based estimation. The modes of the distribution are approximated with higher precision at each level.*

the valid posterior distribution, this can be used in order to obtain a significantly better estimate of the posterior with the same number of samples [67, 111, 153].

In contrast to particle filters, the tree-based filter is a deterministic filter, which was motivated by a number of problem-specific reasons. First of all, the facts that hand motion can be fast and that the hand can enter and leave the camera view in HCI applications call for a method that provides automatic initialization. It is difficult to define a dynamic model that can accurately predict the hand motion at the given temporal resolution of 30 frames per second, and thus it is important to maintain large support regions of the distributions. The second important consideration is the fact that the time required for projecting the model is approximately three orders of magnitude higher than evaluating the likelihood function. Thus the sampling step in a particle filter is relatively expensive and this makes the off-line generation of templates attractive.

The ideas of tree-based filtering and particle filtering can also be combined by using the posterior distribution estimated with the tree-based filter as an importance distribution for a particle filter. Finally, it can be noted that hierarchical approaches have also successfully applied in particle filters directly [38, 140].

6.3 Edge and colour likelihoods

This section introduces the likelihood function which is used within the algorithm. The likelihood $p(\mathbf{z}|\mathbf{x})$ relates observations $\mathbf{z}$ in the image to the unknown state $\mathbf{x}$. The observations are based on the edge map $\mathbf{z}^{edge}$ of the image, as well as pixel colour values $\mathbf{z}^{col}$. These features have proved useful for detecting and tracking hands in previous work [88, 90, 155] and the previous chapter has examined a number of cost functions based on them. In the following sections the joint likelihood of $\mathbf{z} = (\mathbf{z}^{edge}, \mathbf{z}^{col})^{\mathrm{T}}$ is approximated as

$$p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}^{edge}, \mathbf{z}^{col} \,|\, \mathbf{x}) \tag{6.23}$$

$$= p(\mathbf{z}^{edge} \,|\, \mathbf{x})\; p(\mathbf{z}^{col} \,|\, \mathbf{x}) \;\;, \tag{6.24}$$

thus treating the observations independently. We aim to derive likelihood terms for each of the observations, $\mathbf{z}^{edge}$ and $\mathbf{z}^{col}$, in the following paragraphs.

- **Edge likelihood:** In chapter 5 a number of similarity measures have been compared, all of which can be used to define the observation model. Here we choose chamfer matching, introduced in section 5.1, which has been demonstrated to be computationally most efficient when matching a large number of templates. However, deriving a likelihood based on the chamfer distance function is a non-trivial problem. Toyama and Blake [135] suggest a likelihood model, where an exemplar shape template is the centre of a mixture distribution, each component being a *metric exponential* distribution [135]. Given the shape template $\mathcal{P}$ and the observed edge image $\mathbf{z}^{edge}$, the likelihood has the form

$$p(\mathbf{z}^{edge}|\mathcal{P}) = \frac{1}{Z_{\mathcal{P}}} \exp\left(-\lambda_{\mathcal{P}}\; d_{cham}(\mathcal{P}, \mathbf{z}^{edge})\right) \;\;, \tag{6.25}$$

  where $Z_{\mathcal{P}}$ is the partition function. Based on the assumption that the shape $\mathcal{P}$ can be represented as a linear combination of d independent curve basis functions, the chamfer distance is shown to have a $\sigma^2\chi^2_d$ distribution, where σ is a distance constant [135]. These parameters also define $\lambda = 1/2\sigma^2$ and the partition function $Z \propto \sigma^d$ for each template (the subscript $\mathcal{P}$ has been omitted). The likelihood is approximately Gaussian and the parameters σ and d are learnt from a validation set of edge images for each exemplar cluster separately. In our case the templates $\mathcal{P}$ are generated by the model, and are therefore dependent on the values of the state vector $\mathbf{x}$. Analogous to Toyama and Blake, the template at each node is treated as

the centre of an exponential distribution of the form in equation (6.25), and the parameters λ and σ are learnt by computing the moments within a cluster of templates as suggested in [135].

A draw-back of this method of constructing the liklihood is that the distribution of the chamfer cost on random background edges is not taken into account. Different subsets of edge points are used to compute the chamfer distance for different templates. This directly influences the distribution of the feature values, for example to compute the chamfer distance of a template with very few points, only a small number of edge points is used, and this template will have a low chamfer cost near any edge, also on background edges. A different method to define likelihoods is based on the *PDF projection theorem* [7, 96], which states how the distribution on features can be converted to a distribution on the images. However, the application of the PDF projection theorem to deriving likelihood functions is left to future work.

- **Colour likelihood:** The colour cost term has already been defined as a likelihood function $p(\mathbf{z}^{col}|\mathbf{x})$ in chapter 5. It assumes pixelwise independence of the colour values, so that it can be factored into a product of skin-colour likelihoods for pixels within the silhouette $\{k : k \in S(\mathbf{x})\}$ and background colour likelihoods for pixels outside of the silhouette $\{k : k \in \bar{S}(\mathbf{x})\}$:

$$p(\mathbf{z}^{col}|\mathbf{x}) \;=\; \prod_{k \in S(\mathbf{X})} p^{s}\left(I(k)\right) \prod_{k \in \bar{S}(\mathbf{X})} p^{bg}\left(I(k)\right) \tag{6.26}$$

where $I(k)$ is the colour vector at location k in the image. p^s and p^{bg} are the skin colour and background colour distributions, respectively. Skin colour is modelled with a Gaussian distribution in (r, g)-space, for background pixels a uniform distribution is assumed. This is an approximation only, and if a distribution of background colour can be obtained from a static scene, this should be used instead.

6.4 Modelling hand dynamics

The dynamics of global hand motion and finger articulation are modelled in different ways. A simple first order process is used to model the global dynamics

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \;\sim\; \mathcal{N}(\mathbf{x}_{t-1}, \Sigma) \;\; . \tag{6.27}$$

This model makes only weak assumptions by not taking velocity into account and there are many other possible choices for the dynamic model, such as the constant velocity or

constant acceleration model used in chapter 4, or second order models, which have been used for hand tracking [15, 90].

The dynamics for articulated motion are also modelled as a first order process, however, they are learnt from training data obtained with a data glove (see section 6.6). Since discrete regions in state space are considered, the process can be described by a Markov transition matrix $\mathbf{M}^{LL} \in [0,1]^{N_L \times N_L}$, which contains the transition probabilities between the regions $\{S^{jL}\}_{j=1}^{N_L}$ at the leaf-level. In order to evaluate the transitions at different tree levels, a transition matrix $\mathbf{M}^{Ll} \in [0,1]^{N_L \times N_l}$ for each level l of the tree is required, where each matrix contains the values:

$$\mathbf{M}_{ij}^{Ll} = p(\Theta_t^{jl} | \Theta_{t-1}^{iL}) \quad \text{for} \quad i = 1, \ldots, N_L \ \text{ and } \ j = 1, \ldots, N_l \ . \tag{6.28}$$

In practice, these matrices are sparse and the non-zero values are stored in a look-up table.

6.5 Tracking rigid body motion

In the following experiments the hierarchical filter is used to track rigid hand motion in cluttered scenes using a single camera. The results reveal the ability to tolerate self-occlusion during out-of-image-plane rotations. The 3D rotations are limited to a hemisphere and for each sequence a three-level tree is built, which has the following resolutions at the leaf level: 15 degrees in two 3D rotations, 10 degrees in image rotation and 5 different scales, resulting in a total of $13 \times 13 \times 19 \times 5 = 16,055$ templates. The resolution of the translation parameters is 20 pixels at the first level, 5 pixels on the second level, and 2 pixels at the leaf level. Note that for each of the sequence a different tree is constructed, using the corresponding hand configuration.

Figures 6.4 shows the tracking results on an input sequence of a pointing hand during translation and rotation. During the rotation there is self-occlusion as the index finger moves in front of the palm. For each frame the maximum a-posteriori (MAP) solution is shown. An error plot for this sequence is shown in figure 6.5 and in this sequence the error is measured as the localization error of the index fingertip. The RMS error over the whole sequence is 8.0 pixels.

Figure 6.6 shows frames from the sequence of an open hand performing out of image plane rotation. This sequence contains 160 frames and shows a hand first rotating approximately 180 degrees and returning to its initial pose (figure 6.6, two top rows). This is followed by a 90 degree rotation (figure 6.6, two bottom rows), and returning to its initial position. This motion is difficult to track as there is little information available when the palm surface is normal to the image plane.

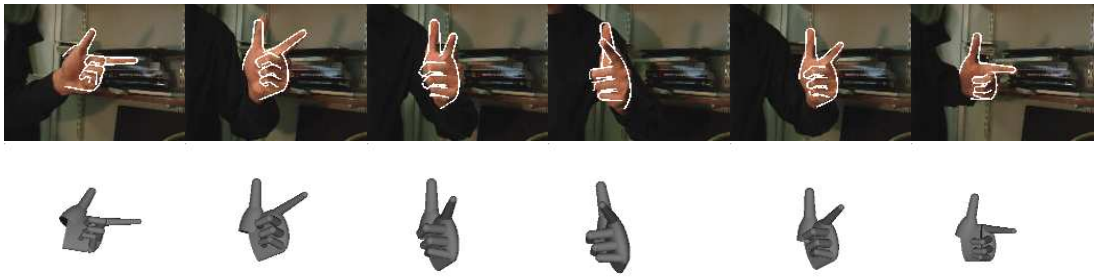
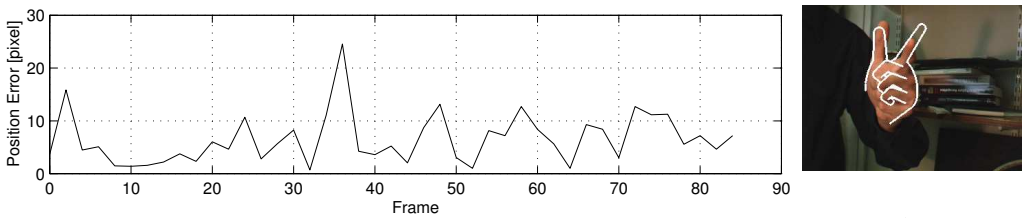


Figure 6.4: **Tracking a pointing hand**. *The images are shown with projected contours superimposed* **(top row)** *and corresponding 3D model* **(bottom row)**, *which are estimated using the tree-based filter. The hand is translating and rotating.*



(a)

(b)

Figure 6.5: **Error performance.** **(a)** *This figure shows the error performance on the image sequence in figure 6.4. Errors are the fingertip localization error of the index finger and were measured against manually labelled ground truth. It can be seen that the model is not exactly aligned in every frame, however, this does not effect subsequent frames. The RMS error over the whole sequence is 8.0 pixels* **(b)** *Frame number 36 with largest error (24 pixels) at the index finger tip; the rest of the MAP model template is still well aligned.*

Figure 6.7 shows example images from a sequence of 509 frames of a pointing hand. The tracker handles fast motion and is able to recover after the hand has left the camera view and re-enters the scene.

The computation takes approximately two seconds per frame on a 1GHz Pentium IV, corresponding to a speed-up of two orders of magnitude over exhaustive detection. Note that in all cases the hand model was automatically initialized in the first frame of the sequence.

6.5.1 Initialization

In a number of experiments the ability of the algorithm to solve the initialization problem was tested. In the first frame of each sequence the posterior term $p(\Theta_1^{j1}|\mathbf{z}_1)$ for each region is proportional to the likelihood value $p(\mathbf{z}_1|\Theta_1^{j1})$ for the first observation $\mathbf{z}_1$. A

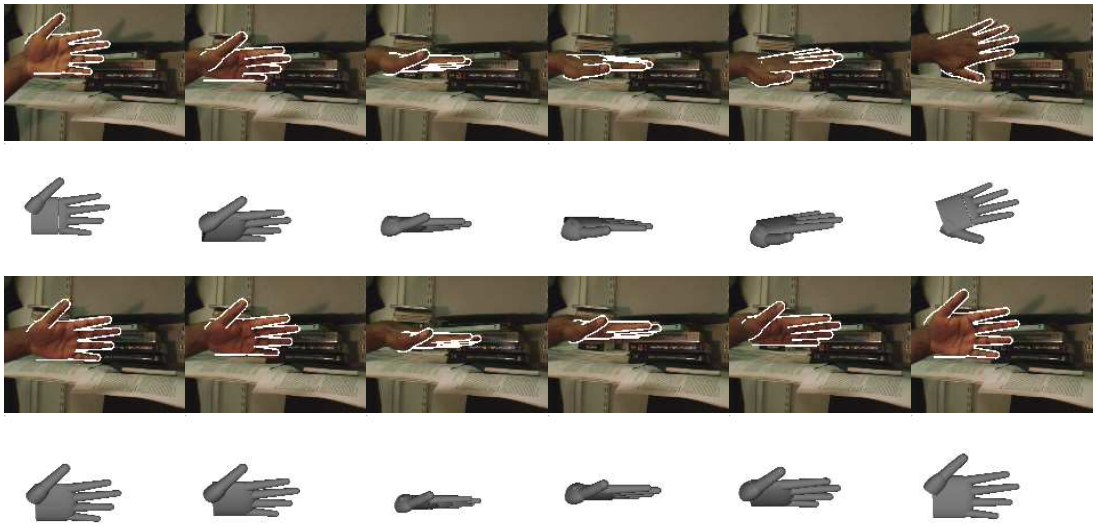


Figure 6.6: **Tracking out-of-image-plane rotation.** *In this sequence the hand undergoes rotation and translation. The frames showing the hand with self-occlusion do not provide much data, and template matching becomes unreliable. By including prior information, these situations can be resolved. The projected contours are superimposed on the images, and the corresponding 3D model is shown below each frame.*

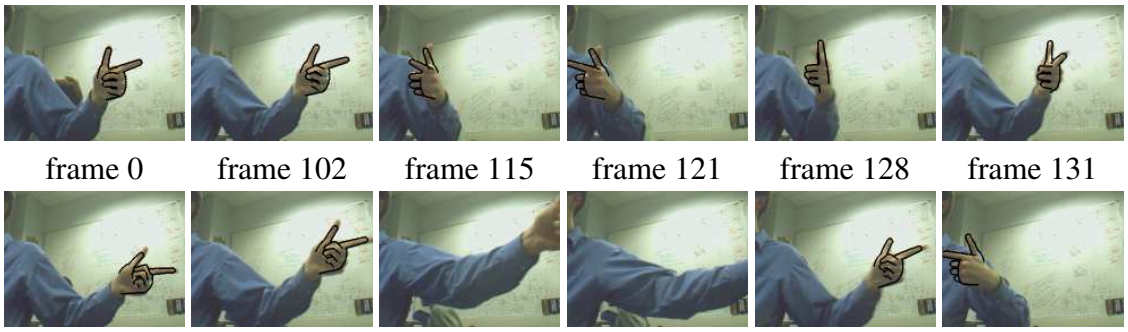


Figure 6.7: **Fast motion and recovery.** *This figure shows frames from a sequence tracking 6 DOF.* **(Top row)** *The hand is tracked during fast motion and* **(bottom row)** *the tracker is able to successfully recover after the hand has left the camera view.*

tree with four levels and 8,748 templates of a pointing hand at the leaf level was generated, corresponding to a search over 972 discrete angles and nine scales, and a search over translation space at single pixel resolution. As before, the motion is restricted to a hemisphere. The search process at different levels of the tree is illustrated in figure 6.8. The templates at the higher levels correspond to larger regions of parameter space, and are thus less discriminative. The image regions of the face and the second hand, for which

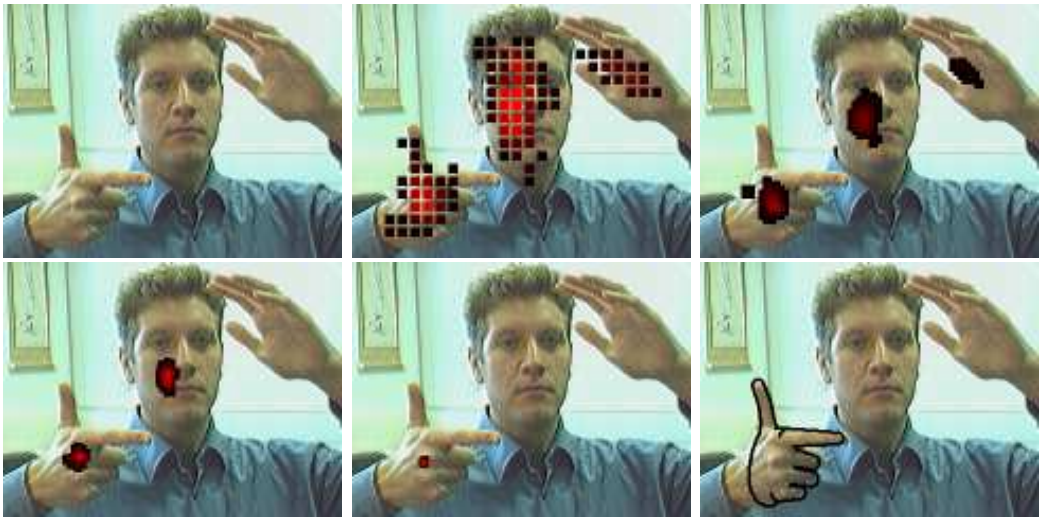


Figure 6.8: **Automatic initialization. Top Left**: *Input image,* **Next**: *Images with detection results super-imposed. Each square represents an image location which contains at least one node with a likelihood estimate above the threshold value. The intensity indicates the number of matches, with high intensity corresponding to more matches. Ambiguity is introduced by the face and the second hand in the image (for which there is no template in the tree). As the search proceeds through levels 1–4, regions are progressively eliminated, resulting in only few final matches.* **Bottom right**: *The template corresponding to the maximum likelihood.*

there is no template, have relatively low cost as they contain skin colour pixels and edges. As the search proceeds, these regions are progressively eliminated, resulting in only few final matches.

Figure 6.9 gives a more detailed view by showing examples of templates at different tree levels which are above (accepted) and below (rejected) the threshold (equation (6.12)) at level l for a different input image. It can be seen that the templates which are above the threshold at the first level do not present very accurate matches, however a large number of templates can be rejected. At lower tree levels the accuracy of the match increases.

6.6 Dimensionality reduction for articulated motion

The 3D hand model presented in chapter 3 allows 21 degrees of freedom to articulated motion, corresponding to the degrees of freedom of the joints in a human hand [2]. Even though there are 21 parameters in our model that are used to describe articulated motion, this motion is highly constrained. Each joint can only move within certain limits. Thus the articulation parameters are expected to lie within a compact region in the 21 dimensional

Figure 6.9: **Search results at different levels of the tree.** *This figure shows templates above and below the threshold values τ_t^l at levels 1 to 3 of the tree, ranked according to their posterior values. As the search is refined at each level, the difference between accepted and rejected templates decreases.*

angle space. Additionally the motion of different joints is correlated, for example most people find it difficult to bend the little finger while keeping the ring finger extended.

Wu *et al.* [153] use principal component analysis (PCA) to project their 20 dimensional joint angle measurements into a 7D space, while capturing 95 percent of the variation within their data set. Zhou and Huang [155] introduce an *eigendynamics* model. These eigendynamics are defined as the motion of a single finger, modeled by a 2D linear dynamic system in a lower dimensional eigenspace, reducing the dimension of the search space from 20 to 10. In both cases, valid finger configurations define a compact region within the subspace and the goal is to sample from this region. One approach, adopted by Wu *et al.*, is to define 28 basis configurations in the subspace, corresponding to each

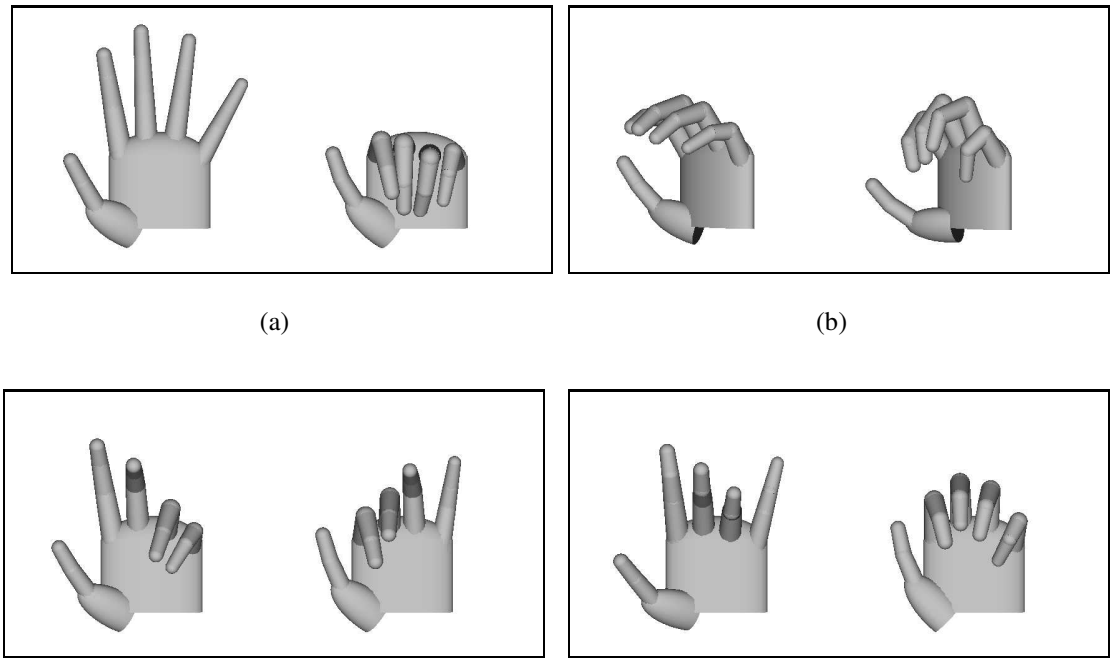
finger being either fully extended or fully bent (not all $2^5 = 32$ combinations of fingers extended/bent are used). It is claimed that natural hand articulation lies largely on linear 1D manifolds spanned by any two such basis configurations, and sampling is implemented as a random walk along these 1D manifolds. Another approach [155] is to train a linear dynamic system on the region while decoupling the dynamics for each finger. In this case one motion sample corresponds to five random walks along 2D curves, each random walk corresponding to the motion of one particular finger.

We have collected 15 sets of joint angles (sizes of data sets: 3,000 to 264,000) using a data glove. The input data was captured from three different subjects who were carrying out a number of different hand gestures. It was found in all cases that 95 percent of the variance was captured by the first eight principal components, in 10 cases within the first seven, which confirms the results reported in [153]. However, except for very controlled opening and closing of fingers, we did not find the one-dimensional linear manifold structure reported in [153]. For example, figure 6.11 shows trajectories between a number of basis configurations used in [153] projected onto the first three eigenvectors. The hand moved between the four basis configurations in random order for the duration of one minute. It can be seen that even in this controlled experiment the trajectories between the four configurations do not seem to be adequately represented by 1D manifolds.

6.6.1 Constructing a tree for articulated motion

Having shown that articulated motion can be approximated in a lower dimensional eigenspace, this section introduces two methods to build the template hierarchy from data sets collected with a data glove. The discrete sets $\{\mathcal{S}^{il}\}_{i=1}^{N_l},\ l = 1, \ldots, L$, can be obtained in the following ways:

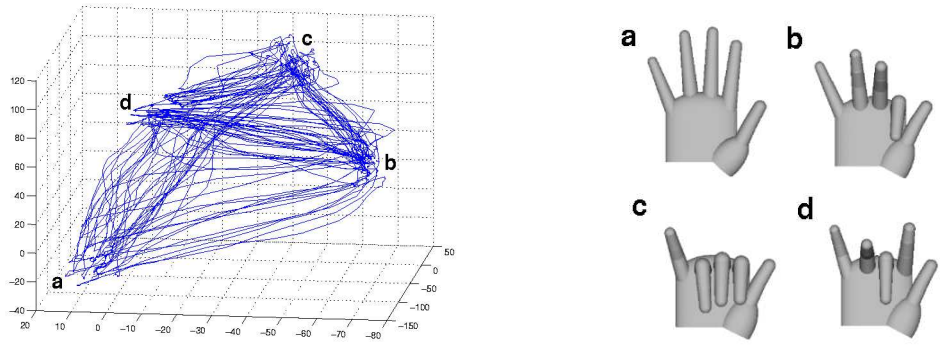
- **Partitioning the eigenspace:** One option to subdivide the space is to use a regular grid. Defining partition boundaries in the original 21 dimensional space is difficult, because the number of partitions increases exponentially with the number of divisions along each axis. Therefore the data can first be projected onto the first k principal components ($k < 21$), and the partitioning is done in the transformed parameter space. The centres of these hyper-cubes are then used as nodes in the tree on one level, where only partitions, which contain data points need to be considered, see figure 6.12 (b). Multiple resolutions are obtained by subdividing each partition.



(a) (b)

(c) (d)

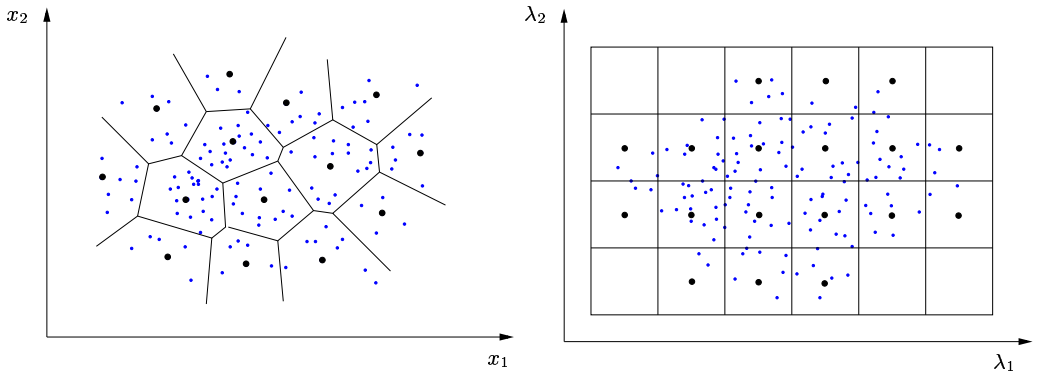
Figure 6.10: **Variation along principal components**. *This figures shows the variation in hand pose when moving away from the mean pose into the direction of the* **(a)** *first,* **(b)** *second,* **(c)** *third, and* **(d)** *fourth principal component. The input data set contained 50,000 joint angle vectors, obtained from a data glove.*



(a) (b)

Figure 6.11: **Paths in the eigenspace**. *This figure shows* **(a)** *a trajectory of projected hand state (joint angle) vectors onto the first three principal components as the hand pose changes between the four poses shown in* **(b)**.

- **Clustering in the original state space:** Since the joint angle data lies within a compact region in state space, the data points can simply be clustered using a hi-



(a) (b)

Figure 6.12: **Partitioning the state space.** *This figure illustrates two methods of partitioning the state space:* **(a)** *by clustering the data points in the original space, and* **(b)** *by first projecting the data points onto the first k principal components and then using a regular grid to define a partition in the transformed space. The figure only shows one level of a hierarchical partition, corresponding to one level in the tree.*

erarchical k-means algorithm with an L_2 distance measure. This algorithm is also known as the *generalized Lloyd algorithm* in the vector quantisation literature [51], where the goal is to find a number of prototype vectors which are used to encode vectors with minimal distortion. The cluster centres obtained in each level of the hierarchy are used as nodes in one level of the tree. A partition of the space is given by the Voronoi diagram defined by these nodes, see figure 6.12 (a).

Both techniques are used to build a tree for tracking finger articulation. In the first sequence the subject alternates between four different gestures (see figure 6.13). For learning the transition distributions, a data set of size 7200 was captured while performing the four gestures a number of times in random order.

In the first experiment the tree is built by hierarchical k-means clustering of the whole training set. The tree has a depth of three, where the first level contains 300 templates together with a partitioning of the translation space at a 20×20 pixel resolution. The second and third level each contain 7200 templates (i.e. the whole data set) and a translation search at 5×5 and 2×2 pixel resolution, respectively.

In the second experiment the tree is built by partitioning a lower dimensional eigenspace. Applying PCA to the data set shows that more than 96 percent of the variance is captured within the first four principal components, thus we partition the four dimensional eigenspace. In this experiment, the first level of the tree contains 360 templates, the sec-

ond and third level contain 9163 templates. The grid resolutions in translation space are the same as in the first experiment. In both cases transition matrices $\mathbf{M}^{Ll}$ are obtained by histogramming the data.

The tracking results for the tree constructed by clustering are shown in figure 6.13. The results for the tree built by partitioning the eigenspace are qualitatively the same. The method of constructing the tree by partitioning the eigenspace has a number of advantages: Firstly, a parametric representation is obtained, which allows for interpolation between hand states, and further optimisation once the leaf level has been reached. Secondly, for large data sets the computational cost of PCA is significantly lower than the cost of clustering.

Figure 6.14 shows the results of tracking global hand motion together with finger articulation. The opening and closing of the hand is captured by the first two eigenvectors, thus only two articulation parameters are estimated. For this sequence the range of global hand motion is restricted to a smaller region, but it still has 6 DOF. In total 35,000 templates are used at the leaf level. The resolution of the translation parameters is 20 pixels at the first level, 5 pixels on the second level, and 2 pixels at the leaf level. The out-of-image-plane rotation and the finger articulation are tracked successfully in this sequence.

6.7 Summary

This chapter has introduced a hierarchical filtering algorithm for three-dimensional hand tracking. The state space is partitioned at multiple resolutions, which allows the definition of a grid-based filter on discrete regions. The 3D model is used to generate templates at the centre of each region, and these are used to evaluate the likelihood function. A dynamic model is given by transition probabilities between regions, which are learnt from training data obtained by capturing articulated motion. The algorithm was tested on a number of sequences, including rigid body motion and articulated motion in front of cluttered background.

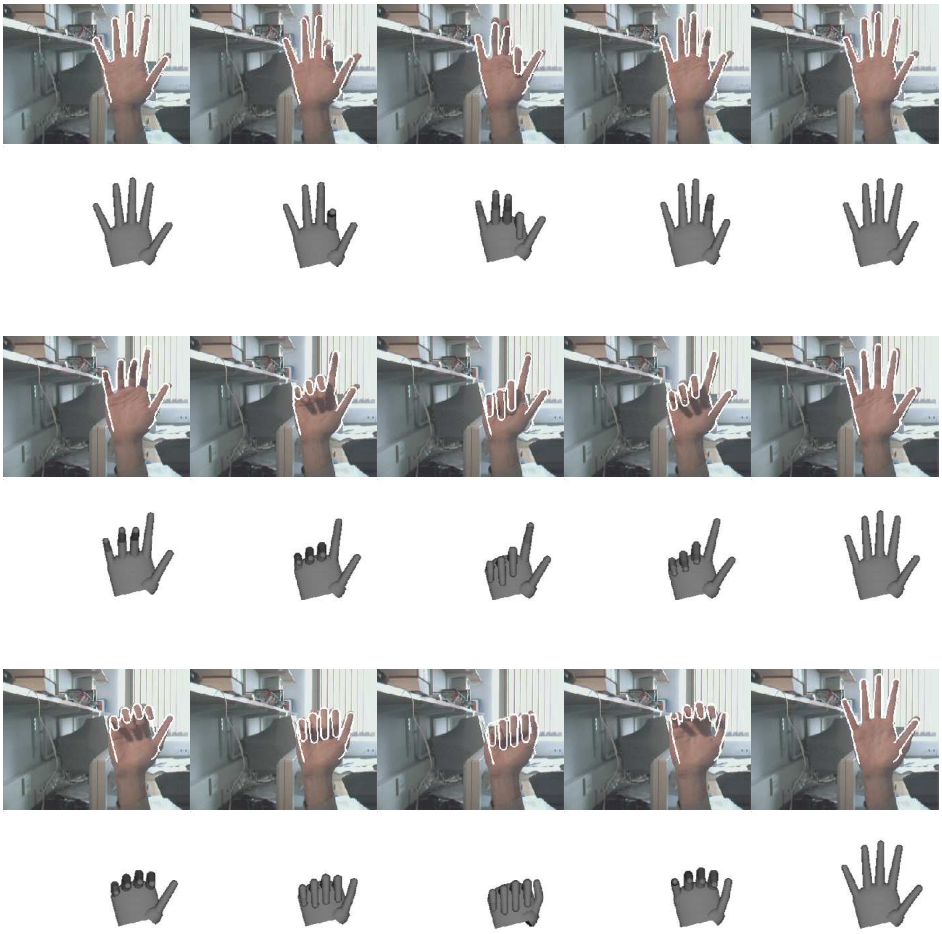


Figure 6.13: **Tracking articulated hand motion.** *In this sequence a number of different finger motions are tracked. The images are shown with projected contours superimposed (top) and corresponding 3D avatar models (bottom), which are estimated using our tree-based filter. The nodes in the tree are found by hierarchical clustering of training data in the parameter space, and dynamic information is encoded as transition probabilities between the clusters.*

Algorithm 6.1 Tree-based filtering equations

Notation: $par(j)$ denotes the parent of node j.

Initialization Step at $t = 1$

At level $l = 1$:

$$p(\Theta_1^{j1}|\mathbf{z}_1) = \frac{p(\mathbf{z}_1|\Theta_1^{j1})}{\sum_{j=1}^{N_1} p(\Theta_1^{j1}|\mathbf{z}_1)} \quad \text{for } j = 1, \ldots, N_1 \ . \tag{6.15}$$

At level $l > 1$:

$$p(\Theta_1^{jl}|\mathbf{z}_1) = \begin{cases} c^{-1}\, p(\mathbf{z}_1|\Theta_1^{jl}) & \text{if } p(\Theta_1^{par(j)}|\mathbf{z}_1) > \tau_t^{l-1}, \\ c^{-1}\, p(\Theta_1^{par(j)}|\mathbf{z}_1) & \text{otherwise,} \end{cases} \tag{6.16}$$

where

$$c = \sum_{j=1}^{N_l} p(\Theta_1^{jl}|\mathbf{z}_1) \ . \tag{6.17}$$

At time $t > 1$

At level $l = 1$:

$$p(\Theta_t^{j1}|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\Theta_t^{j1})\, p(\Theta_t^{j1}|\mathbf{z}_{1:t-1})}{\sum_{j=1}^{N_1} p(\mathbf{z}_t|\Theta_t^{j1})\, p(\Theta_t^{j1}|\mathbf{z}_{1:t-1})} \ , \tag{6.18}$$

where

$$p(\Theta_t^{j1}|\mathbf{z}_{1:t-1}) = \sum_{i=1}^{N_L} p(\Theta_t^{j1}|\Theta_{t-1}^{iL})\ p(\Theta_{t-1}^{iL}|\mathbf{z}_{1:t-1}) \tag{6.19}$$

At level $l > 1$:

$$p(\Theta_t^{jl}|\mathbf{z}_{1:t}) = \begin{cases} c^{-1}\, p(\mathbf{z}_t|\Theta_t^{jl})\, p(\Theta_t^{jl}|\mathbf{z}_{1:t-1}) & \text{if } p(\Theta_t^{par(j)}|\mathbf{z}_{1:t}) > \tau_t^{l-1}, \\ c^{-1}\, p(\Theta_t^{par(j)}|\mathbf{z}_{1:t}) & \text{otherwise,} \end{cases} \tag{6.20}$$

where

$$p(\Theta_t^{jl}|\mathbf{z}_{1:t-1}) = \sum_{i=1}^{N_L} p(\Theta_t^{jl}|\Theta_{t-1}^{iL})\ p(\Theta_{t-1}^{iL}|\mathbf{z}_{1:t-1}) \tag{6.21}$$

and

$$c = \sum_{j=1}^{N_l} p(\Theta_t^{jl}|\mathbf{z}_t) \ . \tag{6.22}$$

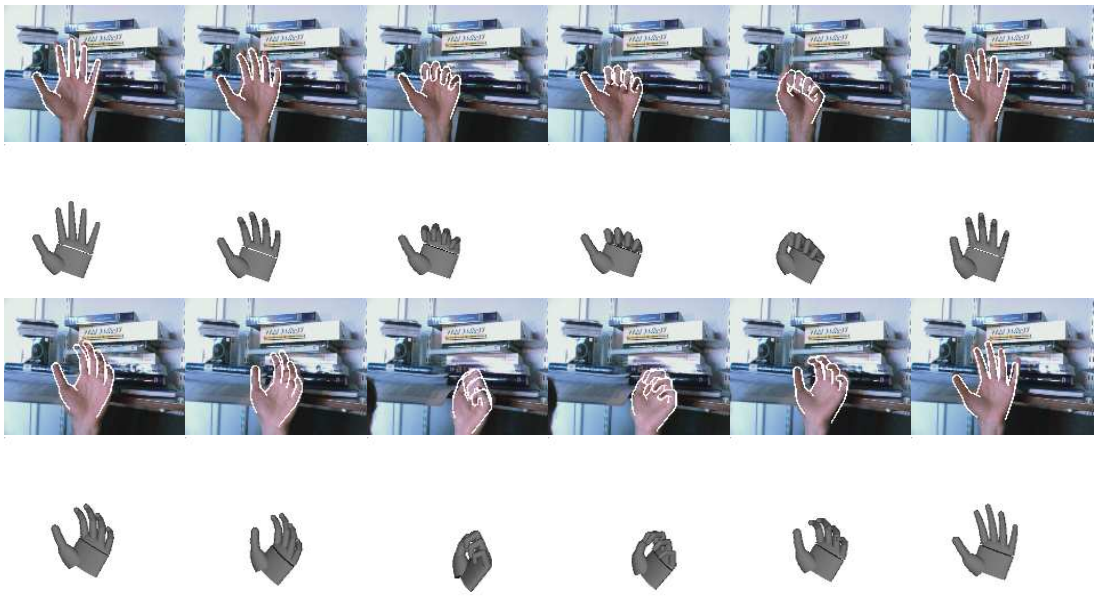


Figure 6.14: **Tracking a hand opening and closing with rigid body motion.** *This sequence is challenging because the hand undergoes translation and rotation while opening and closing the fingers. 6* DOF *for rigid body motion plus 2* DOF *for finger flexion and extension are tracked successfully.*

7 Conclusion

What good are computers? They can only give you answers.

Pablo Picasso (1881-1973)

7.1 Summary

The main contribution of this thesis is the formulation of a framework for recovering the 3D hand pose from image sequences. It draws from two strands of research: hierarchical detection and Bayesian filtering. Reliable detection helps in dealing with difficult problems such as loosing track due to self-occlusion. By using dynamic information, detection is made more efficient by eliminating a significant number of hypotheses. Bayesian methods are attractive as they provide a principled way of encoding uncertainty about parameter estimates. This is particularly necessary for the problem of tracking in clutter as there is much ambiguity, resulting in multi-modal distributions. One of the key issues in Bayesian filtering is how to represent these distributions. Previously grid-based methods, involving partitioning the state space, have proven very successful for propagating distributions in tracking. However, they suffer from the major drawback that they are computationally infeasible in high dimensional spaces. When the state parameters are confined to a compact region, it is suggested to partition this region in a hierarchical fashion and use the resulting tree to represent the distributions.

It has been shown in this work how to construct a 3D hand model from simple geometric primitives. There are several benefits to using a 3D model over parametric contour models or exemplar models. In contrast to parametric contour models, topological shape changes of the contour caused by self-occlusion are handled naturally by a 3D model, without the need of resorting to nonlinear shape spaces [58]. In contrast to exemplar-based models, the use of a 3D parametric model allows to generate further templates online, once an approximate match has been found. Continuous optimization can be used to refine the match or adapt the model to the hand of a different user [132]. A 3D model also has the advantage of semantic correspondence, which allows the definition of a meaning-

ful dynamic model, and can be important for HCI applications. Furthermore, methods based on 3D models can be extended from single to multiple views with relatively little effort, because it simply requires projecting the model into each view.

7.2 Limitations

The range of allowed motion is currently limited by the number of templates. In the current system templates for different scale and orientation parameters are generated off-line, leading to a large number of templates. These templates could also be generated on-line by 2D transformations, and the increase in time required for likelihood computations could be tolerable in the light of memory savings.

On our current hardware (2.4 GHz Pentium IV) the algorithm does not run in real time. Typical run times are two frames per second for a few hundred templates to three seconds per frame for about 35,000 templates. Even though some effort was spent on optimizing the methods, there is still scope for improvement.

Emphasis has been placed on the design of robust similarity measures, which are then used to construct a likelihood function within the filter. For edge detection, the Canny edge detector has been used, which is a standard tool in computer vision [25]. However, it does not always yield consistent edge responses in video sequences due to image noise. The result is also highly dependent on the kernel and threshold parameters in the detector. In practice missing edges sometimes lead to missed detections, and spurious edges can lead to false positive detections. Similarly, the static skin colour features can become unreliable if illumination changes significantly.

7.3 Future work

This section identifies a number of directions of future work. An advantage of the probabilistic approach, which was taken in this work, is that it is modular in the sense that application-specific dynamic models or observation models can be included seamlessly.

- **Observation model:** The design of better similarity measures will directly increase the robustness of the tracker. It will be interesting to investigate using different filter outputs as features, as in [69, 129, 137]. Also the integration of other cues such as texture, motion and shading could lead to better cost functions. If two cameras are available, dense stereo algorithms, can be useful for segmenting the hand in the image.

- **Hand model:** The tracker will also directly benefit from a more accurate geometric hand model and efficient rendering techniques. The model used in this work is comparable to the simple cylinder models of Marr and Nishihara [92] or Rehg and Kanade [109]. Even though such models are a good approximation to the true geometry, advances in computer graphics have led to more complex but accurate models, such as the hand model of Albrecht *et al.* [2]. As computing power increases, it is possible to use these models in computer vision applications.

- **Tree construction:** The tree has been constructed manually in the examples by dividing the parameter space at certain resolutions. Future work should consider methods to make this process automatic, for example by considering the appearance variation within each partition or by clustering the templates based on their appearance similarity.

- **Combining tree-based filtering with continuous methods:** The result of the tree-based filter contains a discretization error due to the partitioning of the state space. However, because the output is a probability distribution, it can be integrated with a continuous filter, such as a particle filter.

- **Extension to multiple views:** The filtering framework presented in chapter 6 can be extended to estimation from multiple calibrated cameras. The same state space partitioning can be used as for the single view case, and the model at the can be projected into each view.

- **Experimental comparison with other approaches:** The step from the research laboratory to commercial applications typically leads via an evaluation of competing methods on benchmark data. Possibly due to the ambiguity of the task, the difficulty to obtain ground truth data, as well as different application goals, there is no publically available data set for three-dimensional hand tracking.

Bibliography

[1] J. K. Aggarwal and Q. Cai. Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[2] I. Albrecht, J. Haber, and H.-P. Seidel. Construction and animation of anatomically based human hand models. In *Proc. Eurographics/SIGGRAPH Symp. on Computer Animation*, 98–109, San Diego, CA, July 2003.

[3] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Boostmap: A method for efficient approximate similarity rankings. In *Boston University Computer Science Tech. Report No. 2003-023*, November 2003.

[4] V. Athitsos and S. Sclaroff. 3D hand pose estimation by finding appearance-based matches in a large database of training views. In *IEEE Workshop on Cues in Communication*, 100–106, December 2001.

[5] V. Athitsos and S. Sclaroff. An appearance-based framework for 3D hand shape classification and camera viewpoint estimation. In *IEEE Conference on Face and Gesture Recognition*, 45–50, Washington DC, May 2002.

[6] V. Athitsos and S. Sclaroff. Estimating 3D hand pose from a cluttered image. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume II, 432–439, Madison, WI, June 2003.

[7] P. M. Baggenstoss. The pdf projection theorem and the class-specific method. *IEEE Transactions on Signal Processing*, (51):672–685, 2003.

[8] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association*. Number 179 in Mathematics in science and engineering. Academic Press, Boston, 1988.

[9] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proc. 5th Int. Joint Conf. Artificial Intelligence*, 659–663, 1977.

[10] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *Proc. International Conference on Computer Vision*, volume I, 454–455, Vancouver, Canada, July 2001.

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intell.*, 24(4):509–522, April 2002.

[12] N. Bergman. *Recursive Bayesian Estimation: Navigation and Tracking Applications*. PhD thesis, Linköping University, Linköping, Sweden, 1999.

[13] P. J. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Analysis and Machine Intell.*, 14(2):239–256, March 1992.

[14] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proc. Conf. Computer Vision and Pattern Recognition*, 232–237, Santa Barbara, CA, June 1998.

[15] A. Blake and M. Isard. *Active Contours : The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer-Verlag, London, 1998.

[16] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Trans. Pattern Analysis and Machine Intell.*, 10(6):849–865, November 1988.

[17] Y. Boykov and D. P. Huttenlocher. A new Bayesian framework for object recognition. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume II, 2517–2523, Fort Collins, CO, June 1999.

[18] M. Brand. Shadow puppetry. In *Proc. 7th Int. Conf. on Computer Vision*, volume II, 1237–1244, Corfu, Greece, September 1999.

[19] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. Conf. Computer Vision and Pattern Recognition*, 8–15, Santa Barbara, CA, June 1998.

[20] L. Bretzner, I. Laptev, and T. Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In *Proc. Face and Gesture*, 423–428, Washington DC, 2002.

[21] T. M. Breuel. Fast recognition using adaptive subdivisions of transformation space. In *Proc. Conf. Computer Vision and Pattern Recognition*, 445–451, Urbana Champaign, IL, June 1992.

[22] T. M. Breuel. *Geometric Aspects of Visual Object Recognition*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, April 1992.

[23] T. M. Breuel. Higher-order statistics in object recognition. In *Proc. Conf. Computer Vision and Pattern Recognition*, 707–8, New York, 1993.

[24] R. S. Bucy and K. D. Senne. Digital synthesis of nonlinear filters. *Automatica*, (7):287–298, 1971.

[25] J. F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intell.*, 8(6):679–698, November 1986.

[26] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, April 1992.

[27] R. Cipolla and A. Blake. The dynamic analysis of apparent contours. In *Proc. 3rd Int. Conf. on Computer Vision*, 616–623, Osaka, Japan, December 1990.

[28] R. Cipolla and P. J. Giblin. *Visual Motion of Curves and Surfaces*. Cambridge University Press, Cambridge, UK, 1999.

[29] R. Cipolla, P. A. Hadfield, and N. J. Hollinghurst. Uncalibrated stereo vision with pointing for a man-machine interface. In *Proc. IAPR Workshop on Machine Vision Applications*, 163–166, Kawasaki, Japan, 1994.

[30] R. Cipolla and N. J. Hollinghurst. Human-robot interface by pointing with uncalibrated stereo vision. *Image and Vision Computing*, 14(3):171–178, April 1996.

[31] R. Cipolla, Y. Okamoto, and Y. Kuno. Qualitative visual interpretation of 3D hand gestures using motion parallax. In *Proc. IAPR Workshop on Machine Vision Applications*, 477–482, Tokyo, Japan, 1992.

[32] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models – their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, January 1995.

[33] F. C. Crow. Summed-area tables for texture mapping. In *Proc. SIGGRAPH*, volume 18, 207–212, July 1984.

[34] Y. Cui and J. J. Weng. Hand segmentation using learning-based prediction and verification for hand sign recognition. In *Proc. Conf. Computer Vision and Pattern Recognition*, 88–93, San Francisco, CA, June 1996.

[35] Y. Cui and J. J. Weng. Appearance-based hand sign recognition from intensity image sequences. *Computer Vision and Image Understanding*, 78(2):157–176, 2000.

[36] Q. Delamarre and O. D. Faugeras. Finding pose of hand in video images: a stereo-based approach. In *Proc. 3rd Int. Con. on Automatic Face and Gesture Recogntion*, 585–590, Nara, Japan, April 1998.

[37] Q. Delamarre and O. D. Faugeras. Articulated models and multi-view tracking with silhouettes. In *Proc. 7th Int. Conf. on Computer Vision*, volume II, 716–721, Corfu, Greece, September 1999.

[38] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume II, 126–133, Hilton Head, SC, June 2000.

[39] J. Deutscher, B. North, B. Bascle, and A. Blake. Tracking through singularities and discontinuities by random sampling. In *Proc. 7th Int. Conf. on Computer Vision*, volume II, 1144–1149, Corfu, Greece, September 1999.

[40] A. Doucet, N. G. de Freitas, and N. J. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[41] T. W. Drummond and R. Cipolla. Real-time tracking of highly articulated structures in the presence of noisy measurements. In *Proc. 8th Int. Conf. on Computer Vision*, volume II, 315–320, Vancouver, Canada, July 2001.

[42] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, New York, second edition, 2001.

[43] P. F. Felzenszwalb. Learning models for object recognition. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume I, 56–62, Kauai, HI, December 2001.

[44] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient matching of pictorial structures. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume 2, 66–73, 2000.

[45] D. A. Forsyth and J. Ponce. *Computer Vision – A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, first edition, 2003.

[46] Y. Freund and R. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *J. of Computer and System Sciences*, 55(1):119–139, 1997.

[47] D. M. Gavrila. Pedestrian detection from a moving vehicle. In *Proc. 6th European Conf. on Computer Vision*, volume II, 37–49, Dublin, Ireland, June/July 2000.

[48] D. M. Gavrila and L. S. Davis. 3-D model-based tracking of humans in action: a multi-view approach. In *Proc. Conf. Computer Vision and Pattern Recognition*, 73–80, San Francisco, June 1996.

[49] D. M. Gavrila and V. Philomin. Real-time object detection for smart vehicles. In *Proc. 7th Int. Conf. on Computer Vision*, volume I, 87–93, Corfu, Greece, September 1999.

[50] A. Gelb. *Applied Optimal Estimation*. The MIT Press, Cambridge, MA, 1974.

[51] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic, 1992.

[52] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to non-linear and non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, 140:107–113, 1993.

[53] C. G. Gross, C. E. Rocha-Miranda, and D. B. Bender. Visual properties of neurons in inferotemporal cortex of the macaque. *J. Neurophysiol.*, 35:96–111, 1972.

[54] I. Haritaoglu, D. Harwood, and L. S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Trans. Pattern Analysis and Machine Intell.*, 22(8):809–830, 2000.

[55] C. G. Harris and C. Stennett. Rapid – a video-rate object tracker. In *Proc. British Machine Vision Conference*, 73–78, Oxford, UK, September 1990.

[56] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2000.

[57] A. J. Heap and D. C. Hogg. Towards 3-D hand tracking using a deformable model. In *2nd International Face and Gesture Recognition Conference*, 140–145, Killington, VT, October 1996.

[58] A. J. Heap and D. C. Hogg. Wormholes in shape space: Tracking through discontinuous changes in shape. In *Proc. 6th Int. Conf. on Computer Vision*, 344–349, Bombay, India, January 1998.

[59] D. Hogg. Model-based vision: a program to see a walking person. *Image and Vision Computing*, 1(1):5–20, February 1983.

[60] N. R. Howe, M. E. Leventon, and W. T. Freeman. Bayesian reconstruction of 3D human motion from single-camera video. In *Adv. Neural Information Processing Systems*, 820–826, Denver, CO, November 1999.

[61] D. P. Huttenlocher. Monte Carlo comparison of distance transform based matching measure. In *ARPA Image Understanding Workshop*, 1179–1183, New Orleans, LA, May 1997.

[62] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Analysis and Machine Intell.*, 15(9):850–863, 1993.

[63] D. P. Huttenlocher, R. H. Lilien, and C. F. Olson. View-based recognition using an eigenspace approximation to the Hausdorff measure. *IEEE Trans. Pattern Analysis and Machine Intell.*, 21(9):951–955, September 1999.

[64] D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge. Tracking non-rigid objects in complex scenes. In *Proc. 4th Int. Conf. on Computer Vision*, 93–101, Berlin, May 1993.

[65] D. P. Huttenlocher and W. J. Rucklidge. A multi-resolution technique for comparing images using the Hausdorff distance. In *Proc. Conf. Computer Vision and Pattern Recognition*, 705–706, New York, June 1993.

[66] M. Isard and A. Blake. CONDENSATION — conditional density propagation for visual tracking. *Int. Journal of Computer Vision*, 29(1):5–28, August 1998.

[67] M. Isard and A. Blake. ICondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proc. 5th European Conf. on Computer Vision*, volume I, 893–908, Freiburg, Germany, June 1998. Springer-Verlag.

[68] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *Proc. 6th Int. Conf. on Computer Vision*, 107–112, Bombay, India, January 1998.

[69] M. Isard and J. MacCormick. BraMBLe: A Bayesian multiple-blob tracker. In *Proc. 8th Int. Conf. on Computer Vision*, volume 2, 34–41, Vancouver, Canada, July 2001.

[70] O. L. R. Jacobs. *Introduction to Control Theory*. Oxford University Press, second edition, 1993.

[71] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, New York, 1970.

[72] B. Jedynak, H. Zheng, and M. Daoudi. Statistical models for skin detection. In *IEEE Workshop on Statistical Analysis in Computer Vision*, 180–193, Madison, WI, June 2003.

[73] I. H. Jermyn and H. Ishikawa. Globally optimal regions and boundaries. In *Proc. 7th Int. Conf. on Computer Vision*, volume I, 20–27, Corfu, Greece, September 1999.

[74] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. *Int. Journal of Computer Vision*, 46(1):81–96, January 2002.

[75] S. J. Julier. *Process Models for the Navigation of High-Speed Land Vehicles*. PhD thesis, Department of Engineering Science, University of Oxford, 1997.

[76] S. J. Julier and J. K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *SPIE Proceedings*, volume 3068, 182–193, April 1997.

[77] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. American Control Conference*, 1628–1632, Seattle, USA, June 1995.

[78] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances. *IEEE Trans. Automatic Control*, 45(3):477–482, March 2000.

[79] I. A. Kakadiaris and D. N. Metaxas. Model-based estimation of 3-D human motion with occlusion based on active multi-viewpoint selection. In *Proc. Conf. Computer Vision and Pattern Recognition*, 81–87, San Francisco, June 1996.

[80] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:35–45, March 1960.

[81] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, editors. *Principles of Neural Science*. McGraw-Hill, New York, fourth edition, 2000.

[82] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proc. 1st Int. Conf. on Computer Vision*, 259–268, London, June 1987.

[83] M. R. Korn and C. R. Dyer. 3D multiview object representations for model-based object recognition. *Pattern Recognition*, 20(1):91–103, 1987.

[84] I. Laptev and T. Lindeberg. Tracking of multi-state hand models using particle filtering and a hierarchy of multi-scale image features. In *IEEE Workshop on Scale-Space and Morphology*, 63–74, Vancouver, Canada, July 2001.

[85] T. Lefebvre, H. Bruyninckx, and J. De Schutter. Comment on 'A new method for the nonlinear transformation of means and covariances in filters and estimators'. *IEEE Trans. Automatic Control*, 47(8):1406–1408, August 2002.

[86] S. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *Proc. 7th European Conf. on Computer Vision*, volume IV, 67–81, Copenhagen, Denmark, May 2002.

[87] J. Lin, Y. Wu, and T. S. Huang. Capturing human hand motion in image sequences. In *Proc. of IEEE Workshop on Motion and Video Computing*, 99–104, Orlando, FL, December 2002.

[88] R. Lockton and A. W. Fitzgibbon. Real-time gesture recognition using deterministic boosting. In *Proc. British Machine Vision Conference*, volume II, 817–826, Cardiff, UK, September 2002.

[89] S. Lu, D. Metaxas, D. Samaras, and J. Oliensis. Using multiple cues for hand tracking and model refinement. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume II, 443–450, Madison, WI, June 2003.

[90] J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Proc. 6th European Conf. on Computer Vision*, volume 2, 3–19, Dublin, Ireland, June 2000.

[91] D. Marr. *Vision*. Freeman, San Francisco, 1982.

[92] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three dimensional structure. *Proc. Royal Society, London*, 200:269–294, 1978.

[93] B. Martinkauppi. *Face Colour under Varying Illumination – Analysis and Applications*. PhD thesis, University of Oulu, Oulu, Finland, 2002.

[94] P. S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, New York, 1979.

[95] I. Mikić, E. Hunter, M. Trivedi, and P. Cosman. Articulated body posture estimation from multi-camera voxel data. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume I, 455–460, Kauai, HI, December 2001.

[96] T. Minka. Exemplar-based likelihoods using the pdf projection theorem. Technical report, Microsoft Research Ltd., Cambridge, UK, March 2004.

[97] M. L. Minsky and S. A. Papert. *Perceptrons : An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, 1969.

[98] T. B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[99] G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *Proc. 7th European Conf. on Computer Vision*, volume III, 666–680, Copenhagen, Denmark, May 2002.

[100] D. D. Morris and J. M. Rehg. Singularity analysis for articulated object tracking. In *Proc. Conf. Computer Vision and Pattern Recognition*, 289–296, Santa Barbara, CA, June 1998.

[101] C. F. Olson and D. P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *Transactions on Image Processing*, 6(1):103–113, January 1997.

[102] J. O'Rourke and N. Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Trans. Pattern Analysis and Machine Intell.*, 2(6):522–536, November 1980.

[103] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill Series in Electrical Engineering: Communications and Signal Processing. McGraw-Hill, New York, third edition, 1991.

[104] V. Pavlović, J. M. Rehg, T.-J. Cham, and K. P. Murphy. A dynamic Bayesian network approach to tracking using learned learned dynamic models. In *Proc. 7th Int. Conf. on Computer Vision*, 94–101, Corfu, Greece, September 1999.

[105] V. Pavlović, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. Pattern Analysis and Machine Intell.*, 19(7):677–695, July 1997.

[106] R. Plänkers and P. Fua. Articulated soft objects for multi-view shape and motion capture. *IEEE Trans. Pattern Analysis and Machine Intell.*, 25:1182–1187, 2003.

[107] D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume II, 467–474, Madison, WI, June 2003.

[108] J. M. Rehg. *Visual Analysis of High DOF Articulated Objects with Application to Hand Tracking*. PhD thesis, Carnegie Mellon University, Dept. of Electrical and Computer Engineering, 1995.

[109] J. M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking. In *Proc. 3rd European Conf. on Computer Vision*, volume II, 35–46, May 1994.

[110] J. M. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *Proc. 5th Int. Conf. on Computer Vision*, 612–617, Cambridge, MA, June 1995.

[111] B. D. Ripley. *Stochastic Simulation*. Wiley, New York, 1987.

[112] S. Romdhani, P. H. S. Torr, B. Schölkopf, and A. Blake. Computationally efficient face detection. In *Proc. 8th Int. Conf. on Computer Vision*, volume II, 695–700, Vancouver, Canada, July 2001.

[113] R. Rosales, V. Athitsos, L. Sigal, and S. Scarloff. 3D hand pose reconstruction using specialized mappings. In *Proc. 8th Int. Conf. on Computer Vision*, volume I, 378–385, Vancouver, Canada, July 2001.

[114] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. Pattern Analysis and Machine Intell.*, 20(1):22–38, 1998.

[115] H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume I, 746–751, Hilton Head, SC, June 2000.

[116] J. G. Semple and G. T. Kneebone. *Algebraic Projective Geometry*. Oxford Classic Texts in the Physical Sciences. Clarendon Press, Oxford, UK, 1998. Originally published in 1952.

[117] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proc. 9th Int. Conf. on Computer Vision*, volume II, 750–757, Nice, France, October 2003.

[118] N. Shimada, K. Kimura, and Y. Shirai. Hand gesture recognition using computer vision based on model-matching method. In *6th International Conference on Human-Computer Interaction*, Yokohama, Japan, July 1995.

[119] N. Shimada, K. Kimura, and Y. Shirai. Real-time 3-D hand posture estimation based on 2-D appearance retrieval using monocular camera. In *Proc. Int. Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems*, 23–30, Vancouver, Canada, July 2001.

[120] N. Shimada and Y. Shirai. 3-D hand pose estimation and shape model refinement from a monocular image sequence. In *Proc. of Int. Conf. on Virtual Systems and Multimedia*, 423–428, Gifu, Japan, September 1996.

[121] N. Shimada, Y. Shirai, Y. Kuno, and J. Miura. Hand gesture estimation and model refinement using monocular camera. In *Proc. 3rd Int. Conf. on Automatic Face and Gesture Recognition*, 268 – 273, Nara, Japan, April 1998.

[122] H. Sidenbladh, M. J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *Proc. 7th European Conf. on Computer Vision*, volume 1, 784–800, Copenhagen, Denmark, May 2002.

[123] L. Sigal, M. Isard, B. H. Sigelman, and M. J. Black. Attractive people: Assembling loose-limbed models using non-parametric belief propagation. In *Adv. Neural Information Processing Systems*, 2004. to appear.

[124] L. Sigal, S. Sclaroff, and V. Athitsos. Estimation and prediction of evolving color distributions for skin segmentation under varying illumination. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume 2, 2152–2159, Hilton Head, SC, June 2000.

[125] C. Sminchisescu and B. Triggs. Estimating articulated human motion with covariance scaled sampling. *Int. Journal of Robotics Research*, 22(6):371–393, 2003.

[126] H. W. Sorenson. *Bayesian Analysis of Time Series and Dynamic Models*, chapter Recursive Estimation for nonlinear dynamic systems, 127–165. Marcel Dekker inc., 1988.

[127] S. Spielberg (director). Minority Report, 20th Century Fox and Dreamworks Pictures, 2002.

[128] T. Starner, J. Weaver, and A. Pentland. Real-time American Sign Language recognition using desk and wearable computer-based video. *IEEE Trans. Pattern Analysis and Machine Intell.*, 20(12):1371–1375, December 1998.

[129] J. Sullivan, A. Blake, and J. Rittscher. Statistical foreground modelling for object localisation. In *Proc. 6th European Conf. on Computer Vision*, volume II, 307–323, Dublin, Ireland, June/July 2000.

[130] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Trans. Pattern Analysis and Machine Intell.*, 13(7):703–714, 1991.

[131] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Learning a kinematic prior for tree-based filtering. In *Proc. British Machine Vision Conference*, volume 2, 589–598, Norwich, UK, September 2003.

[132] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume I, 127–133, Madison, WI, June 2003.

[133] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[134] C. Tomasi, S. Petrov, and A. K. Sastry. 3D tracking = classification + interpolation. In *Proc. 9th Int. Conf. on Computer Vision*, volume II, 1441–1448, Nice, France, October 2003.

[135] K. Toyama and A. Blake. Probabilistic tracking with exemplars in a metric space. *Int. Journal of Computer Vision*, 48(1):9–19, June 2002.

[136] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Proc. 7th Int. Conf. on Computer Vision*, 255–261, Corfu, Greece, September 1999.

[137] J. Triesch and C. Von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *IEEE Trans. Pattern Analysis and Machine Intell.*, 23(12):1449–1453, 2001.

[138] R. van der Merwe, N. de Freitas, A. Doucet, and E. A. Wan. The unscented particle filter. In *Adv. Neural Information Processing Systems*, 584–590, Denver, CO, November 2000.

[139] R. van der Merwe and E. Wan. Sigma-point Kalman filters for probabilistic inference in dynamic state-space models. In *Proc. Workshop on Advances in Machine Learning*, Montreal, Canada, June 2003.

[140] V. Verma, S. Thrun, and R. Simmons. Variable resolution particle filter. In *Proc. Int. Joint. Conf. on Art. Intell.*, August 2003.

[141] P. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume I, 511–518, Kauai, HI, December 2001.

[142] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Proc. 9th Int. Conf. on Computer Vision*, volume II, 734–741, Nice, France, October 2003.

[143] C. Von Hardenberg and F. Bérard. Bare-hand human-computer interaction. In *Proc. ACM Workshop on Perceptive User Interfaces*, Orlando, FL, November 2001.

[144] E. A. Wan and R. van der Merve. The unscented Kalman filter for nonlinear estimation. In *Proc. of IEEE Symposium 2000 on Adaptive Systems for Signal Processing, Communications and Control*, 153–158, Lake Louise, Canada, October 2000.

[145] E. A. Wan, R. van der Merve, and A. T. Nelson. Dual estimation and the unscented transformation. In *Adv. Neural Information Processing Systems*, 666–672, Denver, CO, November 1999.

[146] P. Wellner. The digitaldesk calculator: tangible manipulation on a desk top display. In *Proc. UIST '91 ACM Symposium on User Interface Software and Technology*, 27–33, New York, November 1991.

[147] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *Proc. 9th Int. Conf. on Computer Vision*, volume I, 353–360, Nice, France, October 2003.

[148] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intell.*, 19(7):780–785, 1997.

[149] Y. Wu and T. S. Huang. Capturing articulated human hand motion: A divide-and-conquer approach. In *Proc. 7th Int. Conf. on Computer Vision*, volume I, 606–611, Corfu, Greece, September 1999.

[150] Y. Wu and T. S. Huang. Vision-based gesture recognition: A review. In A. Braffort et al., editor, *Gesture-Based Communication in Human-Computer Interaction*, 103–116, 1999.

[151] Y. Wu and T. S. Huang. View-independent recognition of hand postures. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume II, 88–94, Hilton Head, SC, June 2000.

[152] Y. Wu and T. S. Huang. Human hand modeling, analysis and animation in the context of human computer interaction. *IEEE Signal Processing Magazine, Special issue on Immersive Interactive Technology*, 18(3):51–60, May 2001.

[153] Y. Wu, J. Y. Lin, and T. S. Huang. Capturing natural hand articulation. In *Proc. 8th Int. Conf. on Computer Vision*, volume II, 426–432, Vancouver, Canada, July 2001.

[154] J. Yang, W. Lu, and A. Waibel. Skin-color modeling and adaptation. In *Proc. 3rd Asian Conf. on Computer Vision*, 687–694, Hong Kong, China, January 1998.

[155] H. Zhou and T. S. Huang. Tracking articulated hand motion with eigen-dynamics analysis. In *Proc. 9th Int. Conf. on Computer Vision*, volume II, 1102–1109, Nice, France, October 2003.