H24/3060

Errata

Page xv – Paragraph 2, lines 3 & 4: '2' should be replaced by 'two'.
Page 1 – Paragraph 2, line 4: 'changing' should be replaced by 'change'.
Page 3 – Paragraph 3, line 5: 'within the computer …' should be replaced by 'within computer vision …'.
Page 4 – Paragraph 5, line 4: 'chapter 3 is used' should be replaced by 'chapter 3 used'.
Page 7 – Section 1.4.2, references (3) and (4): the journal name should be taken out and 'in preparation' included within brackets.
Page 11 – Paragraph 3, line 2: '2' should be replaced by 'two'.
Page 35 – Section 3.2.1, line 3: 'produce' should be replaced by 'product'.
Page 68 – Paragraph 5, line 2 & 3: '?' should be removed.
Page 73 – Figure caption, line 5: 'theMahalanobis….' should be replaced by 'the Mahalanobis….'.
Page 74 – Section 4.4, line 3: 'approach is that, the …' should be replaced by 'approach is that the …'.
Page 75 – Paragraph 2, line 2: 'Under modeling, occurs…' should be replaced by 'Under modeling occurs…'.
Page 77 – Paragraph 1, lines 2: 'the residual v_j and estimated…' be replaced by 'the residual v_j, and estimated…'. Line 3: 'S_j are …' should be replaced by 'S_j, are …'. Line 3: 'seeAppendix …' should be replaced by 'see Appendix …'.
Page 78 – Paragraph 2, line 1: 'residuals v_j, are …' should be replaced by 'residuals, v_j, are …'.
Page 79 – Section 4.5.2, line 1: 'is that, their …' should be replaced by 'is that their …'.
Page 103 – Section 4.10.1, paragraph 1, line 3: 'can't' should be replaced by 'can not'.
Page 104 – Paragraph 4, line 3: 'S has …' should be replaced by 'S, has …'. Line 6: '2' should be replaced by 'two'.
Page 105 – Paragraph 3, line 2: 'proposed my Mayback …' should be replaced by 'proposed by Maybeck …'.
Page 108 – Section 4.11, paragraph 2, line 3: 'condition' should be replaced by 'conditions'; line 4: 'or even converge to …' should be replaced by 'or even result in convergence to ...'.
Page 140 – Paragraph 1, line 5: 'thus indicating to the …' should be replaced by 'thus indicating the…'.
Page 149 – Paragraph 2, line 4: 'σ …' should be replaced by 'The value of σ …'.
Page 153 – Paragraph 1, line 1: '4' should be replaced by 'four' (in both instances), 'gives' should be replaced by 'give'; line 8: '2/3's' replaced by '2/3'.
Page 158 – Paragraph 3, line 6: 'the 4 cones are …' should be replaced by 'the four cones is …'. Section 6.8, line 4: 'Then by …' should be replaced by 'Then, by …'.
Page 175 – Section 7.8.1, paragraph 2, line 3: 'Bake' should be replaced by 'Blake'.
Page 204 – Paragraph 2, line 1: 'give extended …' should be replaced by 'give an extended …'; line 2: 'CONDESATION' should be replaced by 'CONDENSATION'.
Page 206 – Paragraph 2, line 1: 'CONDESATION' should be replaced by 'CONDENSATION'.
Page 210 – Section 8.4.3, paragraph 1, line 5: 'All' should be replaced by 'all'; line 9: 'Where' replaced by 'where'.
Page 211 – Paragraph 1, line 1: 'Where' should be replaced by 'Here'; line 3: '*m*' should be replaced by 'The value *m*'.
Page 215 – Paragraph 1, line 5: 'CONDESATION' should be replaced by 'CONDENSATION'. Section 8.5.3, line 1: 'condition' should be replaced by 'conditions'.
Page 216 – Paragraph 2, line 2: 'test' should be replaced by 'tests'.
Page 217 – Paragraph 1, line 1: 'CONSENSATION' should be replaced by 'CONDENSATION'.
Page 218 – Paragraph 3, line 2: 'CONDENATION' should be replaced by 'CONDENSATION'.
Page 272 – References [2], [4] and [18]: The abbreviation '*IJCV*' should read '*International Journal of Computer Vision*'.
Page 278 – References [121] and [123]: The abbreviation '*IEEE PAMI*' should read '*IEEE Transaction on Pattern Analysis and Machine Intelligence*'.

Visual Tracking: Development, Performance Evaluation, and Motion Model Switching

Prithiviraj Tissainayagam

BEng.(Hons.), Electronic and Electrical Engineering, University of Leeds, UK, 1992
MEng.(Sc.), Signal Processing, University of Melbourne, Australia, 1995

A Thesis Submitted in Fulfillment of the Requirements for the Degree of Doctor of Philosophy of Monash University

Department of Electrical and Computer Systems Engineering
Monash University
Clayton, Victoria 3168
Australia

April 2001

Table of Contents

List of Figures

List of Tables

Summary

This thesis focuses on developing efficient point feature and contour tracking algorithms to track objects. A particular emphasis is on the incorporation of multiple motion models within the tracking framework. The algorithms presented are capable of automatically switching motion models in order to track an object of interest within a sequence of images. The thesis qualitatively demonstrates the promising performance of the trackers developed on a variety of image sequences. We also provide empirical and theoretical techniques to quantitatively assess and support the performance of the tracking algorithms.

The first part of the project deals with formulating an efficient '*point feature*' tracking algorithm. We initially carry out an empirical study on the selection of feature point detectors for point feature tracking applications. Four well-known corner detectors are considered and their performance is assessed against corner properties such as 'corner localization' and 'corner stability'.

We then select two corner detectors: those that were considered most suitable for point feature tracking based upon the earlier work in the thesis. The corner detectors were employed as part of a model switching point feature tracking algorithm that we proposed. The algorithm combines the Multiple Hypothesis Tracking technique with an Interacting Multiple Model filtering framework. The resulting algorithm (named as MHT-IMM algorithm) is shown to provide promising results: including the ability to track point (corner) features that move with variable motion. As a further study, we address the question of how to assess the performance of tracking algorithms. We attempt to formulate closed-formed solutions and empirical evaluation methods for predicting a feature tracker's performance when employing different motion models. The evaluation is considered under varied levels of clutter and noise.

The second part of the thesis focuses on formulating an efficient '*contour tracking*' algorithm. Initially an attempt is made to extend the MHT based algorithm for contour tracking of rigid objects. The MHT technique is applied in 2 stages. First, it is applied to group segments of edges that belong to the same object (object identification stage), and this stage is followed by temporal tracking of 'key points' from the object of interest using the MHT-IMM tracker (object tracking stage). This tracker presents several limitations and cannot be easily extended to track complex deformable objects. To overcome the limitations of this tracker, a cubic B-spline based tracking algorithm is formulated to track deformable object contours. Modified versions of Blake et al. and Hogg et al.'s trackers are combined and then coupled with the IMM algorithm to track deformable objects. The new tracker is capable of automatically switching motion models to track object contours that move with variable motion. The resulting algorithm (named the CONT-IMM tracker) is shown to provide impressive results.

Finally, the CONT-IMM algorithm's performance is quantitatively assessed against the Condensation algorithm of Blake and Isard and the Pedestrian tracker of Hogg and Baumberg. The results have shown that the CONT-IMM tracker is comparable to the other 2 trackers in terms of the quality of results, and in some cases outperforms the other 2 algorithms.

Declaration

I declare that this thesis contains no material which has been accepted for the award of any other degree or diploma in any university or institution, and to the best of my knowledge and belief, it contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

[redacted]

Prithiviraj Tissainayagam

April 2001

Acknowledgements

I wish to express my utmost appreciation and gratitude to my supervisor Assoc. Prof. David Suter for his invaluable guidance, advice, and constructive criticism throughout this research work, which made this work possible within the allocated time. The motivation to technical excellence and the freedom he gave me was essential for my work. I also extend my sincere thanks to him for agreeing to be my supervisor despite my work commitments at NEC (Australia) Ltd. Pty. I also thank him for providing the funds I needed to attend 3 international conferences. I wish to thank my associate supervisor Prof. Greg Eagan for his support and advice over the last 5 years, particularly for the recommendation that he gave me to obtain a place in the Department of Electrical and Computer Systems Engineering as a part time postgraduate.

Many thanks are due to NEC (Australia) Pty. Ltd. for sponsoring me to undertake a PhD degree. Particularly I wish to thank Mr. P. Taylor (Group General Manager, Transmission Network Division), Mr. J. DelPapa (General Manger, Transmission Network Division), and Dr. E. Stumpf (Manager, Transmission Systems Department) for their valuable support and continuous encouragement.

I would also like to thank my fellow postgraduate colleagues and postdoctoral fellows with whom I had the opportunity to interact with during my stay at Monash University. Most of them have in one way or another contributed to the friendly, inspiring and scholastic environment in the Electrical and Computer Systems Engineering Department.

I also like to thank the following people for their advice and help at various stages of this project: Prof. Y. Bar-Shalom (Dept. of Electrical Engineering, University of Connecticut, USA), Dr. I. Cox (Senior Research Scientist, NEC Research Institute, NJ, USA), Dr. J. M. Roberts (Research Scientist, CSIRO, Australia), Dr. Ngan (Research Leader, Industrial Research Limited, Auckland, NZ), Dr. M. Isard (Research Fellow, Oxford University, UK), Dr. R. Curwen (Research Scientist, INRIA, France), and Richard Morris (Vision Group, Leeds University, UK).

Last but not least, I wish to thank my wife Shobana, daughter Joanna, and son Jonathan for their patience, encouragement and support throughout my candidature. For that, I wish to dedicate this thesis to them.

Finally, all praise and thanks to the Lord Jesus for His guidance and strength, without which I would never have completed this thesis.

Chapter 1

Introduction

The study of visual tracking has become a vital area of research within the computer vision and image processing community within the last decade. One reason for the increased attention towards visual motion study is because of the falling cost of computational power and the availability of sufficient storage to process large amounts of image data (image sequences). Another reason is the development of powerful algorithms that can be implemented in real or near-real time on relatively modest computers.

For a human brain, identifying and tracking an object (static or dynamic) over a period of time is a relatively simple task. For a machine vision system, such a seemingly simple problem is a highly challenging task. The basic problem that needs solving in visual tracking is the correspondence of objects over a number of image frames (the object can possibly changing shape and position over time). A sequence of images collected at or near video rate typically does not change radically from frame to frame, and this redundancy of information over multiple images can be extremely helpful in disambiguating the visual input, whether to track individual objects or to perform a more general motion segmentation.

1.1 Visual Tracking

A track is defined in terms of the clustering (association) of a set of measurements that originate with the same target, and of the estimation of that target's state trajectory:

"*A track is a state trajectory estimated from a set of measurements that have been associated with the same target*" – Bar-Shalom and Fortmann [6].

In visual tracking the targets are objects in the scene, and the measurements are of the 2D image positions corresponding to 3D points on those objects. Visual tracking solves for the correspondence between measurements made in the images of a motion sequence. The track is a description of the relative motion of the object over time, and this allows image features which originate from the same physical point on the object to be associated.

The importance of tracking, as a visual competence, is illustrated by many potential applications (described in chapter 2). In most cases, by solving the correspondence problem over a number of frames. Tracking also allows high level abstraction from visual data, by recovering the trajectory of an object over time. More recently computer vision researchers have focused their attention in areas such as the monitoring and interpretation of scenes (including reconstruction of scenes), creating artificial environments, learning human-computer interaction, and learning behavioral patterns of objects within a scene using visual tracking techniques. Some details of these applications are discussed in the next chapter.

A trivial form of visual tracking is based on the assumption that the target moves only a small distance between each frame of the image sequence. This assumption results in a simple scheme whereby in each image a search is made in the vicinity of the target's location in the previous image. The small search region means that this technique is fast, however it will fail if the target velocity becomes too large.

For larger target velocities, some form of prediction is required. This is achieved by introducing a model of the dynamics of the target. The tracking task becomes that of finding the model which best fits the target's trajectory. For a full treatment of tracking, the noise properties of the measurements must also be taken into account. High confidence measurements must be given greater weight than low confidence, noisy measurements. But it is not enough to blindly include all measurements in the estimation of the feature trajectory. Where measurements have some discriminating attributes, such as orientation in the case of curvature segments, these may be used to decide which measurements originate from which target. This is known as data association. For the single target case, data association is simply deciding when to reject observations that are unlikely to originate from the target. For a multiple target case, the problem becomes somewhat complicated, and methods such as the multiple hypothesis approach have to be considered to solve the data association problem for each of the targets considered.

1.2 Motivation for this Research

Despite the vast wealth of visual tracking material that is available in the literature, there is a lack of knowledge concerning the performances of many tracking algorithms. Most of the algorithms published have been reported to be successful for a narrow band of applications. How those algorithms perform under different environments and how effective they are for a variety of different applications remains an open question. The literature survey carried out at the beginning of the project revealed that very little theoretical and empirical work has been done in the area of performance analysis and the assessment of visual tracking algorithms.

Another area that is not sufficiently addressed in the literature is that of tracking objects that move with multiple motions. A fundamental assumption of most tracking algorithms is that the object moves with a constant motion (eg: constant velocity). Such an assumption may be computationally efficient, but cannot cope with tracking objects that move with multiple motions. An example test case is to track a moving pedestrian who might be walking, running or even standing still. In this example, a single model based tracker (such as a single model Kalman filter) will invariably fail to track the person completely.

While visual tracking is a relatively new area of research for the computer vision community, the target tracking researchers (mainly control and signal processing scientists) have studied tracking in general to a large extent. Early contributions in tracking date back as far as the 1950's and 60's. Many powerful tracking algorithms developed in particular within the last 3 decades have not received sufficient attention within the computer vision researchers. This is mainly because of the lack of computational power to run vision algorithms in real or near real time. Since current day machines are able to process large amount of data in fast times and can store large amounts of data, a review of some of these algorithm for vision related applications seemed worthwhile.

This thesis address the issues mentioned above. We study visual tracking in two parts. The first part deals with the study of tracking a single pixel (point features or corners) through an image sequence, while the second part deals with temporal tracking of outlines of objects (contours of rigid and deformable objects). Initially we attempt to formulate a point-feature tracking algorithm that is capable of switching motion models according to the object's motion (a multiple motion model tracker). In the process we also formulate performance prediction techniques that can be applied to a tracker that employs different motion models. We assess the tracker's performance under varied clutter and noise, and where possible, we have also assessed the tracker's performance against other established point feature trackers. The latter part of the thesis tries to address tracking contours of objects that move with variable motion (multiple motion). Again, a motion-model switching tracker is introduced which can adapt to multiple motions of deformable objects. Finally, we provide empirical evaluation methods to quantitatively assess the performance of a contour tracker. We have also compared the performance of our tracker with other established contour trackers.

In focussing on the model switching aspect of tracking algorithms (which is inadequately addressed in the current literature), we introduce recursive tracking algorithms (previously, these were mainly reported in the control literature) that can perform model switching operation very efficiently. We combine some of these algorithms with computer vision techniques to formulate robust visual trackers. The visual tracking algorithms developed in this thesis have been applied on artificial and real image sequences. The trackers have been demonstrated to perform well with promising results.

1.3 Thesis Overview

Chapter 1 – Introduction – This chapter gives a general introduction to visual tracking and the motivation for this research. It also provides the thesis overview with the contributions made towards visual tracking. The chapter concludes by giving a list of published papers that arose from this research.

Chapter 2 – Visual Tracking: A Survey – Provides a literature survey on existing visual tracking techniques. We provide brief description of well known research papers in the area of optical flow tracking, point feature tracking, region tracking, curve tracking, model based tracking, and tracking algorithms that are employed in computer vision research. Techniques that closely match our work are elaborated more in the relevant chapters of this thesis.

Chapter 3 – Assessing the Performance of Corner Detectors for Point Feature Tracking – The chapter provides performance assessment methods for the suitability of corner extractors for tracking point (corner) features in long image sequences. We propose empirical evaluation methods based on simple statistical performance test to assess corner properties such 'corner localization' and 'corner stability' which are crucial for point feature tracking [179]. The assessment tests are conducted using static image sequences (without moving objects) and assessed at varied noise levels.

Chapter 4 – Point Feature Tracking with Automatic Motion Model Switching – In this chapter we propose a feature tracker that can track features moving with multiple motions. The corner features are extracted using detectors that were deemed suitable for tracking in long sequences (results from chapter 3 is used). The point feature tracker that we propose is based on the combination of the Multiple Hypothesis Tracking (MHT) algorithm [58, 152] and the Interacting Multiple Model (IMM) algorithm [5, 28], hence the tracker is named the MHT-IMM algorithm. We demonstrate the model switching ability of the tracker by employing image sequences which contain objects that move with variable motion. Qualitative and quantitative results are presented to support the promising performance of the tracker [180, 181, 185].

Chapter 5 – Performance Prediction Analysis of a Point Feature Tracker based on Different Motion Models – The main focus of this chapter is an attempt to formulate theoretical closed form solutions for predicting the performance of a feature point tracker under clutter. Formulations are developed for 3 motion models (a constant position, a constant velocity and a constant acceleration model) for predicting the correct data association at the 'next time-step', assuming that up to the 'current time-step' data association has been correct. We also make an attempt to provide closed form solutions for a tracker when recovering from a false match and predicting a correct match at the 'next time-step' (based on each of the 3 motion models in turn). Theoretical and empirical formulations are evaluated against Monte-Carlo simulations using synthetic and real image sequences. We show that the theoretical performance predictions are a credible representation for experimental performance [183, 186]. The performance prediction of a tracker is measured using 2 quantities: The 'track-purity' and 'track-life'. Finally the MHT-IMM tracker is assessed against an established tracker (the KLT tracker) under varied noise levels to evaluate the robustness of the tracker.

Chapter 6 – Extension of a Point Feature Tracker for Rigid Object Tracking – The primary objective of this chapter is to present a rigid object tracker based on the combination of a point feature tracking algorithm [180] and a contour segmentation algorithm [57]. Both algorithms employed are based on the MHT principle [58]. We have demonstrated the object tracker's ability by tracking simple rigid objects using real image sequences [182]. The primary contribution of this chapter is to apply the MHT technique for object tracking (as opposed to tracking point features only).

Chapter 7 – Contour Tracking with Automatic Motion Model Switching – This chapter presents a deformable contour-tracking algorithm for objects that move with multiple motion. The roots of the tracker are based on Blake et al.'s [25, 24] and Hogg et al.'s [11] tracking algorithms. A decomposed shape space is tracked using the IMM algorithm (similar to that reported in Chapter 4) to achieve multiple model switching. The resulting tracking algorithm is named the CONT-IMM tracker. We have demonstrated the ability of the tracker to track deformable objects (including those that move with multiple motions) employing a variety of image sequences with promising results [184].

Chapter 8 – Performance Measures for Assessing Contour Trackers – In this chapter the performance of the CONT-IMM tracker is assessed against the Condensation algorithm of Blake and Isard [104] and the Pedestrian tracker of Baumberg and Hogg [11, 10]. The chapter also provides empirical performance measures to quantitatively assess the output of the trackers. We have shown that CONT-IMM outperforms the other two trackers in terms of quality of results achieved for the experiments carried out.

Chapter 9 – Conclusion – Finally this chapter provides a general conclusion for the research undertaken. We provide a general discussion on the merits and demerits of the algorithms developed, and where possible propose methods for improvement. We also discuss possible avenues for future research directions and possible approaches that can be taken to accomplish some of the tasks.

1.4 Publications Arising from this Project

The following papers have been published or are in the process of being reviewed for publication.

1.4.1 National and International Conferences

(1) P. Tissainayagam and D. Suter, "Comparison of Corner Extractors for Tracking Objects in Long Image Sequences", Proc. *International Workshop on Image Analysis and Information Fusion (IAIF '97)*, pp. 171-181, Adelaide, Australia. Oct. 1997.

(2) P. Tissainayagam and D. Suter, "Visual Tracking and Motion Determination using the IMM Algorithm", *International Conference on Pattern Recognition (ICPR '98)*, pp. 289-291, Brisbane, Australia. Aug. 1998.

(3) P. Tissainayagam and D. Suter, "Visual Feature Tracking with Automatic Motion Model Switching", *First International Workshop on Computer Vision, Pattern Recognition and Image Processing (CVPRIP '98)*, pp. 322-325, Durham, NC, USA. Oct. 1998.

(4) P. Tissainayagam and D. Suter, "Object Tracking in Image Sequences using the Multiple Hypothesis Approach", *First International Workshop on Computer Vision, Pattern Recognition and Image Processing (CVPRIP '98)*, pp. 473-475, Durham, NC, USA. Oct. 1998.

(5) P. Tissainayagam and D. Suter, "Visual Tracking with Multiple Motion Models", *IAPR Machine Vision Applications (MVA '98)*, pp. 414-417, Chiba, Japan. Nov. 1998.

(6) P.Tissainayagam and D. Suter, "Performance Prediction Analysis for Visual Tracking Algorithms", *Irish Machine Vision and Image Processing Conference (IMVIP '99),* pp.141-158, Dublin, Ireland. Sept.1999.

(7) P.Tissainayagam and D. Suter, "Performance of Visual Tracking Algorithms", *Digital Image Computing: Techniques and Application (DICTA '99),* pp.206-211, Perth, Western Australia. Dec. 1999.

(8) P.Tissainayagam and D. Suter, "Contour Tracking in Image Sequences", *Digital Image Computing: Techniques and Application (DICTA '99),* pp.110-115, Perth, Western Australia. Dec. 1999.

(9) P.Tissainayagam and D. Suter, "Tracking multiple object contours with automatic motion model switching", *International Conference on Pattern Recognition (ICPR '2000),* pp.1146-1149, Barcelona, Spain. Sept. 2000.

1.4.2 Journal Publications

(1) P.Tissainayagam and D. Suter, "Visual Tracking with Automatic Motion Model Switching", *International Journal of Pattern Recognition*", Elsevier Publications, Vol.(34), pp.641-660, 2001.

(2) P.Tissainayagam and D. Suter, "Performance Prediction Analysis of Linear Point Feature Trackers Based on Different Motion Models", *Computer Vision and Image Understanding (CVIU)* – 2001. (accepted for publication).

(3) P. Tissainayagam and D. Suter, "Assessment of corner detectors for point feature tracking applications", *International Journal of Computational Intelligence and Applications*, 2001 - (to be submitted)

(4) P. Tissainayagam and D. Suter, "Deformable contour tracking with automatic motion model switching", *Journal of Pattern Recognition*, 2001 - (to be submitted)

1.4.3 Technical Research Reports

(1) P. Tissainayagam and D. Suter, "Performance Analysis of Corner Detectors for tracking Features in Image Sequences", *Technical Report MECSE-1997-3*, Dept. of Electrical and Computer Systems Engineering, Monash University, Clayton, Australia. 1997.

(2) P. Tissainayagam and D. Suter, "Motion Model Selection for Visual Feature Tracking", *Technical Report MECSE-1997-4*, Dept. of Electrical and Computer Systems Engineering, Monash University, Clayton, Australia. 1997.

(3) P. Tissainayagam and D. Suter, "Variable Motion Determination and Tracking using the IMM Algorithm", *Technical Report MECSE-1998-4*, Dept. of Electrical and Computer Systems Engineering, Monash University, Clayton, Australia. 1998.

(4) P. Tissainayagam and D. Suter, "Tracking Objects in Image Sequences", *Technical Report MECSE-1998-5*, Dept. of Electrical and Computer Systems Engineering, Monash University, Clayton, Australia. 1998.

(5) P. Tissainayagam and D. Suter, "Performance analysis of Point-Feature Trackers", *Technical Report MECSE-1998-6*, Dept. of Electrical and Computer Systems Engineering, Monash University, Clayton, Australia. 1998.

(6) P. Tissainayagam and D. Suter, "Efficient Contour Tracking in Extended Image Sequences", *Technical Report MECSE-1999-2*, Dept. of Electrical and Computer Systems Engineering, Monash University, Clayton, Australia. 1999.

(7) P. Tissainayagam and D. Suter, "Performance Analysis of Contour Trackers", *Technical Report MECSE-2000-1*, Dept. of Electrical and Computer Systems Engineering, Monash University, Clayton, Australia. 2000.

Chapter 2

Visual Tracking: A Literature Survey

2.1 Introduction

The literature on tracking tends to be split roughly into two broad categories - methods which track by looking at flows in the image and methods which track by matching a model of the object being tracked to part of the image. The flow based methods generally assume little or no prior knowledge of the object being tracked, and tend to work by grouping together sets of small, low level image features (single pixels, corners, etc., or even small regions) with consistent motion together. By following these groupings over time they achieve tracking. Model-based methods hypothesize a model of either the target's shape, expected deformation, motion, intensity characteristics or other distinguishing attributes. The tracking process is reduced to finding the parameters which make the model fit the video image best (particularly of interest to us is tracking contours of rigid and deformable objects that pertain to a model of interest).

2.2 Importance of Visual Tracking

Tracking has been studied extensively in the computer vision literature, both because of its intrinsic interest and because of the large number of applications. For example, tracking human movement for security applications [11,154, 159, 76]; tracking of human body organs such as the left-ventricle, lungs etc. for medical diagnostics [1, 92, 94, 108]; tracking head and faces for people identification [16, 122, 123, 195, 202]; tracking components in production line [51]; and tracking applications in agriculture [155]. Another area of tracking includes that of autonomous robots being able to follow objects in their environment [153]. One commonly studied special case of this concerns autonomous guided vehicles for driving on roads [129], which must track the features of the road [66] and also other moving vehicles [172, 173, 157, 72]. Static systems may also be used to track vehicles, either to collect traffic data from highway scenes [72, 117, 118] or to analyse complex environments such as airports [3, 175, 69]. Tracking may also be used in robot arm applications to capture multiple views of an object from a moving camera and thus compute trajectories for exploring free-space, or to select an optimal grasp to pick up the object [27].

There is increasing interest in using computer vision for lip tracking to aid speech recognition [34, 114] and reliable hand tracking for a variety of applications [91] such as sign interpretation. Various other systems have been proposed for both tracking [150, 151, 25] and gesture recognition [30, 29, 20]. Hand gestures are a special case of the developing field of "perception of action" which attempts to use tracking information to infer knowledge about a scene. This has roots in the tracking of people [95, 10, 11, 13, 14, 20, 32, 83, 33] for surveillance applications, as well as creating artificial environments [100, 101, 201] which respond to human actions, for example creating an interactive playroom for children [101]. There is much current interest in learning to classify the output of such trackers into behaviours, for example [30, 110, 29]. General techniques for tracking, not tied to any particular application, include the use of optic-flow information, for example [96, 111, 37], rigid three-dimensional models [84, 127] and contour outlines [113, 50, 51, 24]. Other successful tracking methodologies which do not use an explicit object model include the Hausdorff-distance tracker [98], and systems which track point features in an image stream and use geometrical rigid-body constraints to group sets of features into clusters belonging to the same object [192, 193]. More details of some of these techniques are given in the sections which follow.

We have informally classified our visual tracking survey into 5 main sections. They are: optical flow tracking (general tracking of light flow without following any particular object of interest), point feature tracking (tracking distinguished points from an object or a scene, such as corner points), region tracking (tracking a region that contain objects of interest), curve/contour tracking (tracking the silhouette of rigid and deformable objects), and model based tracking (tracking an object whose characteristics are known prior to tracking). In the following sections we shall provide examples of each of the tracking methods mentioned.

2.3 Optic Flow Tracking

Optic-flow has long been used (Horn and Schunk [96], Black and Anandan [17], Ju et al. [111]) as a way both to estimate dense motion fields over the entire visible region of an image sequence (e.g.Black and Anandan [17], Ju et al [111]), and to segment areas of consistent flow into discrete objects (e.g. Black and Jepson [18], Weber and Malik [198]). In order to solve the optic-flow constraint equation it is necessary to either apply regularisation, assuming change in motion is smooth over an image region, or parameterise the motion in an entire region using a low-dimensional model, for example an affine model. Black et al. have developed a series of robust methods for determining optic flow ([17], [109], [18], [111]). The "skin and bones" model Ju et al. [111] combines many of the techniques in their earlier papers to determine a dense motion field as a tiling of the image. Each tile may contain multiple affine motions, and these motions are robustly regularised across adjoining tiles to provide smooth motion information even in regions with little texture. Modern developments of correlation tracking employ similar techniques to parameterised optic-flow estimation. For example the framework adopted by Hager and Toyoma [81] for correlation tracking of a rectangular image patch undergoing affine deformations is closely related to parameterised optic-flow based methods; where optic-flow methods estimate affine parameters of deformation between consecutive images, the correlation tracker estimates parameters relative to an initial template image. A very efficient algorithm is presented in [81] which transfers most of the computation to an off-line processing stage and allows affine correlation tracking to proceed in real time.

Some well-known papers in the literature are surveyed in the following paragraphs.

Most approaches for estimating optical flow assume that, within a finite image region, only a single motion is present. This single motion assumption is violated in common situations involving transparency, depth discontinuities, independently moving objects, shadows, and specular reflections. To robustly estimate optical flow, the single motion assumption must be relaxed. Black and Anandan [17] describe a framework based on robust estimation that addresses violations of the brightness constancy and spatial smoothness assumptions caused by multiple motions. They show how the robust estimation framework can be applied to standard formulations of the optical flow problem thus reducing their sensitivity to violations of their underlying assumptions. The approach has been applied to three standard techniques for recovering optical flow: area-based regression, correlation, and regularization with motion discontinuities. This work focuses on the recovery of multiple parametric motion models within a region as well as the recovery of piecewise-smooth flow fields and provides examples with natural and synthetic image sequences.

Bab-Hadiashar and Suter [2] present a robust optical flow technique. The problem is formulated as a set of over determined simultaneous linear equations. The authors introduce and study 2 new robust optical flow methods. The first technique is based on using the Total Median of Squares to detect the outliers. Then the inlier group is solved using the least squares technique. The second method employs a new robust statistical method named the Least Median of Squares Orthogonal Distances to identify the outliers and then uses total least squares to solve the optical flow problem. The performances of the methods are studied on real and synthetic data. The authors indicate that the results obtained outperform many of the other techniques published in the literature.

Weber and Malik [198] address the problem of segmenting images and then building three-dimensional models of the objects in the image. They attempt to do this by using optic flow to provide a dense displacement map for points in a scene (a mapping of the motion of individual points over two or more frames). Clusters of points which share common fundamental matrices are grouped together into individual objects. The inverse depth of these points is then recovered from their displacements and the fundamental matrix by using an affine camera approximation. This gives a 3D surface map of the object.

Yacoob and Davis [202] provide an approach for learning and estimating temporal flow models from image sequences. The temporal flow models are represented as a set of orthogonal temporal flow bases that are learned using principal component analysis of instantaneous flow measurements. Spatial constraints on the temporal flow are also developed for modelling the motion of regions in rigid and coördinated motion. The performance of these models is demonstrated on several long image sequences of rigid and articulated bodies of motion.

Black and Yacoob [20, 21] describe a system that explores the use of local parameterized models of image motion for recovering and recognizing the non-rigid and articulated motion of human faces. Parametric flow models (for example affine) are used for estimating motion in rigid scenes. They observe that within local regions in space and time, such models not only accurately model non-rigid facial motions but also provide a concise description of the motion in terms of a small number of parameters. These parameters are intuitively related to the motion of facial features during facial expressions and it is shown how expressions such as anger, happiness, surprise, fear, disgust, and sadness can be recognized from the local parametric motions in the presence of significant head motion. The motion tracking and expression recognition approach was reported to perform with high accuracy in extensive laboratory experiments involving 40 subjects as well as in television and movie sequences.

Black et. al. [18] also describe an approach named “Eigen Tracking” for tracking rigid and articulated objects using a view-based representation. The approach builds on and extends work on eigenspace representations, robust estimation techniques, and parameterized optical flow estimation. First, it is noted that the least-squares image reconstruction of standard eigenspace techniques has a number of problems and Black et. al. reformulate the reconstruction problem as one of robust estimation. Second, a definition for a "subspace constancy assumption" is made that allows to exploit techniques for parameterized optical flow estimation to solve for both, the view of an object and the affine transformation between the eigenspace and the image. To account for large affine transformations between the eigenspace and the image, Black et. al. define a multi-scale eigenspace representation and a coarse-to-fine matching strategy. Finally, these techniques are used to track objects over long image sequences in which the objects simultaneously undergo both affine image motions and changes of view. In particular this "EigenTracking" technique was used to track and recognize the gestures of a moving hand.

Jepson and Black [109] provide another approach to dealing with issues such as the treatment of out-liers in component velocity measurements and the modelling of multiple motions within a patch which arise from occlusion boundaries or transparency. The algorithm is based on the use of a probabilistic mixture model to explicitly represent multiple motions within a patch. The authors use a simple extension of the EM algorithm to compute a maximum likelihood estimate for the various motion parameters. The approach is reported to be computationally efficient and claims to provide robust estimates of the optical flow values in the presence of out-liers and multiple motions.

Beauchemin and Barron [15] investigated the computation of optical flow in a survey they conducted. Widely known methods for estimating optical flow are classified and examined by scrutinising the hypothesis and assumptions they use. The survey concludes with a discussion of current research issues. In another paper, Barron et al. [4] also present a comprehensive performance analysis of optical flow techniques. For a common set of real and synthetic image sequence, they report the results of a number of regularly cited optic flow techniques, including instances of differential, matching, energy-based, and phase-based methods. Their comparisons are primarily empirical and concentrate on the accuracy, reliability, and density of velocity measurements. They show that performance can differ significantly among the techniques they had considered.

2.4 Point Feature Tracking

Point features are distinctive image points corresponding to objective 3D scene elements that are in most instances accurately locatable and recur in successive images, which makes them explicitly trackable over time. The term "corners" is used to refer to point features that are loci of two-dimensional intensity change, i.e. 'second-order features'. This includes points of occlusion (e.g. T, Y and X junctions), structural discontinuities (e.g. L junctions) and various curvature maxima (e.g. texture flecks or surface markings). Corners impose more constraint on the motion parameters than edges, therefore the full optic flow field is recoverable at corner locations [168]. Corners are also often more abundant than straight edges in the natural world making them ideal features to track in an indoor and outdoor environment. To find further details on various corner detectors, the reader is referred to [140, 134]. However, in this section we are interested in providing only a brief survey on point feature tracking methods reported in the literature.

One of the earliest image registration technique was presented by Lucas and Kanade [128], that makes use of the spatial intensity gradient of the images to find a good match using a type of Newton-Ralphson iteration. The technique is fast, as it examines far fewer potential matches between the images than other existing techniques. Furthermore, this registration technique can be generalised to handle rotation, scaling and shearing. The authors also showed how the technique could be adapted for use in a stereo vision system.

No feature-based vision system can work unless good features can be identified and tracked from frame to frame. Although the problem of tracking itself is addressed to a large extent, selecting features that can be tracked well and correspond to physical points in the world is still hard. A feature selection criterion was presented by Shi and Tomasi in [171], that is optimal by construction because it is based on how the tracker works. They also present a feature monitoring method that can detect occlusions, dis-occlusions, and features that do not correspond to points in the world. The methods provided are based on a new tracking algorithm that extends previous Newton-Raphson style search methods to work under affine image transformations. A further improvement of this algorithm was proposed by Tommasini et al. [191] which is reported to improve the quality of the results over Shi and Tomasi's method.

Broida and Chellappa [36] proposed a method for estimating the kinematics and structure of a rigid object from a sequence of monocular images. The problem they consider involves the use of a sequence of noisy monocular images of a three-dimensional moving object to estimate both its structure and kinematics. The object is assumed to be rigid, and its motion is assumed to be smooth. A set of object match points is assumed to be available, consisting of fixed features on the object, the image plane coordinates of which have been extracted from successive images in the sequence. Structure is defined as the 3-D positions of these object feature points, relative to each other. Rotational motion occurs about the origin of an object-centered coordinate system, while translational motion is that of the origin of this coordinate system. Impressive results using real imagery is presented. Other noteworthy contributions by Chellappa et al. in the area of point feature tracking can be found in [38, 203, 207].

Kang, Szeliski, and Shum [112] present a feature tracker for long image sequences based on simultaneously estimating the motions and deformations of a collection of adjacent image patches. By sharing common corner nodes, the patches achieve greater stability than independent patch trackers. Modelling full bilinear deformations enables tracking in sequences that have large non-translational motions and/or foreshortening effects. They demonstrate the superiority of their results with respect to previous algorithms. One attraction of the system is that the feature detection and tracking procedures complement each other, thus providing an efficient tracking algorithm.

Gennery [75] describe a method for tracking a known 3D object as it moves with 6 degrees of freedom. The method uses the predicted position of known features on the object to find the features in images from one or more cameras, then the system measures the position of the features in the images, and uses these measurements to update the estimates of position, orientation, linear velocity, and angular velocity of the object model. The features usually used are brightness edges that correspond to markings or the edges of solid objects, although point features can also be used. The solution for object position and orientation is a weighted least squares adjustment that includes filtering over time, which reduces the effects of errors, allows extrapolation over times of missing data, and allows the use of stereo information from multiple camera images that are not coincident in time. The filtering action is derived so as to be optimum if the acceleration is random. The filtering is equivalent to a Kalman filter, but for efficiency it is formulated differently in order to take advantage of the dimensionality of the observations and the state vector which occur in this problem. The method can track accurately with arbitrarily large velocities, as long as the angular acceleration is small. Results are presented showing the successful tracking of partially obscured objects with clutter.

Chetverikov and Verestoy [44] present a point feature tracking algorithm that was designed to efficiently track and resolve features that temporally disappear and appear from the field of view. Correspondences between moving points are established in a competitive linking process that develops as the trajectory grows. Appearing and disappearing points are treated in a natural way as the points that do not link. The algorithm also addresses the issue of handling incomplete trajectories, especially when the number of points and their speeds are large, and trajectory ambiguities are frequent.

Sethi and Jain [167] formulate the point feature correspondence (between frames) problem as an optimisation problem and propose an iterative algorithm to find trajectories of points in a monocular image sequence. A modified form of this algorithm is also studied to handle occlusion. Results have been reported on a variety of scenes.

A way to recover a sparse image flow field, which doesn't rely on the motion constraint assumptions is to track the motion of small distinct image features from frame to frame. One such image feature often used is the corner, as this enables both components of the image flow field to be locally determined. In the ASSET-2 system, Smith segments and tracks vehicles in real-time [172, 173] using matched corners to obtain the optic-low field. Sets of points with similar motion are then clustered together into individual objects. Following these groupings over time enables the relative motion of objects in the real world to be inferred.

Reid and Murray [153] also track objects by following the motion of corners. Constant velocity Kalman filters are used to track individual corners between frames. An interesting addition to the usual corner tracking is that sets of corners matched over three frames are used to create an affine coordinate basis and a fixation point located in this basis. The bases can be used to locate the fixation point in a new frame, even if no actual corner or feature exists at that point - it is only necessary to match enough corners to re-create the basis. Different sets of points may be used to form the basis used to locate the fixation point in each frame. This allows corners to drop in and out (a well-known property of corner detectors) without affecting the ability to localize the fixation point on the target.

Tracking line segments as opposed to tracking point features have also been considered by some authors. Deriche and Faugeras [65] propose a line tracking system based on a prediction and matching strategy, while Mirmehdi and Ellis [133] propose a parallel approach to tracking edge segments in dynamic scenes using a modified version of Kalman filter.

2.5 Region Tracking

Region tracking in general refers to an area being tracked in the image plane. The area could contain one or more objects of interest. One advantage of such techniques is that one does not have to consider the shape or characteristics of the object being tracked. Another advantage is to save computational cost. In the following sections we briefly discuss some well known region tracking methods that have been presented in the literature.

Hager and Belhumeur [80] present an efficient region tracking algorithm which uses parametric models of geometry and illumination. They first develop a computationally efficient method for handling the geometric distortions produced by changes in pose. Then they combine geometry and illumination into an algorithm that tracks large image regions using no more computation than would be required to track with no accommodation for illumination changes. Finally, they augment these methods with techniques from robust statistics and treat occluded regions on the object as statistical outliers. Experimental results are given to demonstrate the effectiveness of their methods.

Salama and Abbot [164] describe an approach to visual tracking for monocular and binocular image sequences. The method combines Kalman type prediction with steepest descent search for correspondences, using 2D affine mapping between images. The approach differs from many recent tracking systems, which emphasize the recovery of 3D motion structure of objects in the scene. The authors argue that 2D area based matching is sufficient in many situations of interest. Results are provided to support their argument.

Bascle, Bouthemy, Deriche, and Meyer [8, 132] describe an approach to track complex primitives along image sequences – integrating snake based contour tracking and region based motion analysis. First, a snake tracks the region outline and performs segmentation. Then the motion of the extracted region is estimated by a dense analysis of the apparent motion over the region, using spatio-temporal image gradients. Finally, this motion measurement is filtered to predict the region location in the next frame, and thus to guide (initialize) the tracking snake in the next frame. The two approaches collaborate and exchange information to overcome the limitations of each of them. The method is illustrated by experimental results on real images. Extensions and further improvements of this work can be found in [9].

Cohen and Medioni [46] address the problem of detecting and tracking of moving objects in a video stream obtained from a moving airborne platform. The method proposed relies on a graph representation of moving objects, which enables to derive and maintain a dynamic template of each moving object by enforcing their temporal coherence. The template with the graph representation provides characterisation of object trajectories as an optimal path in a graph. The tracker has mechanisms to deal with partial occlusions, stop and go motion in very challenging situations. The tracking algorithm has been applied to a number of real image sequences with promising results.

Sclaroff and Isidoro [165] present a new region-based approach to non-rigid motion tracking. Shape is defined in terms of a deformable triangular mesh that captures object shape plus a color texture map that captures object appearance. Photometric variations are also modelled. Non-rigid shape registration and motion tracking are achieved by posing the problem as an energy-based, robust minimization procedure. The approach provides robustness to occlusions, wrinkles, shadows, and specular highlights. The formulation is tailored to take advantage of texture mapping hardware available in many workstations, PC's, and game consoles. This enables non-rigid tracking at speeds approaching video rate.

Gil et. al. [77] provide a vehicle tracking method by combining estimates provided by multiple motion models. Two tracking systems, based on the bounding-box and on the 2D pattern of the targets, provide individual motion parameter estimates to the combined method, which in turn produces a global estimate. The algorithm is applied to image sequences that are taken under varying weather and road conditions. Performances of the local and global estimates of the algorithm are also analyzed. In another line of work [78], the authors also provide a feature selection criterion that is efficiently utilized for the tracking of vehicles.

Shi and Malik [170] propose a motion segmentation algorithm that aims to break a scene into its most prominent moving groups. A weighted graph is constructed on the image sequence by connecting pixels that are in the spatio-temporal neighbourhood of each other. At each pixel they define motion profile vectors which capture the probability distribution of the image velocity. The distance between motion profiles is used to assign a weight on the graph edges. Using normalized cuts they find the most salient partitions of the spatio-temporal graph formed by the image sequence. For segmenting long image sequences, they have developed a recursive update procedure that incorporates knowledge of segmentation in previous frames for efficiently finding the group correspondence in the new frame.

Bremond and Thonnat [35] propose a method of tracking multiple non-rigid objects in a cluttered scene. First, the characteristics of non-rigid objects are considered. To cope with object characteristics, a tracked target is defined as a moving region tracked individually or as a group of moving regions tracked globally. Then they show how to compute the trajectory of a target and the correspondences between known targets and moving regions newly detected. In the case of an ambiguous correspondence, a compound track is defined to freeze the associations between targets and moving regions until more accurate information is available. Promising results are provided.

Huwer and Niemann [99] provide a tracking system based on projection-histograms. They have observed that tracking with projection histograms provided remarkable results compared with standard correlation methods. In their work, a new template-based method relying on projection histograms (RPH) is described and compared with two commonly known template-based methods namely the normalized cross-correlation (NCC) and displaced-frame-distance (DFD) methods. The input to the system consists of live or recorded video data where filter-based pre-processing can be applied before tracking in order to enhance features such as edges, textures etc. A region of interest (ROI) is taken as a template for tracking. In subsequent images tracking exploits a Kalman-filtered local search in order to renew correspondence between the object template and the new object location. Comparative tests are demonstrated with real-life image sequences taken in underground stations.

2.6 Curve (Contour) tracking

Curve or contour tracking is the tracking of outlines of objects. The outlines (silhouettes) can be that of a rigid object or a deformable object. Within the last two decades curve tracking has become one of the main areas of research within the image processing and computer vision community. We describe some of the methods that have been published in the literature.

The snake of Kass et al. [113] is the forerunner to a whole host of work on physics-based tracking. A snake is a flexible contour with certain internal stiffness properties. It tracks by being 'attracted' to various image features. The scenario is formulated in terms of energy: the image is abstracted as an energy landscape, with desirable features (usually edges) having low energy. A snake, when placed on such a landscape, locks onto features by sliding down into these energy minima whilst simultaneously minimising its internal potential energy. In practical terms, the energy gradient is evaluated (via image analysis) at a set of control points along the snake (the image first undergoes a Gaussian blur in order to widen the energy wells in the landscape) and the snake is deformed iteratively until it reaches a stable position. The whole process can alternatively be thought of in terms of force-based tracking: external gravity-like forces pull the snake downhill in the energy landscape and internal forces maintain its smoothness. This is a local optimisation process and so extends naturally from object location to object tracking. In addition, the physical properties of the snake can be extended to include momentum, thus providing some form of temporal prediction. Terzopoulos and Szeliski reformulate the snake dynamics within a probabilistic framework and introduce the Kalman snake [176] (based on a Kalman filter) which, as well as predicting the snake's position, can provide confidence limits for such predictions.

Another successful curve tracker is the active contour of Blake, Curwen et al [61, 27]. They show how snake technology can be used with B-spline contours, and also introduce a more efficient method for feature search [163], whereby image edges are sought along contour normals using a divide-and-conquer strategy. This avoids the need for the Gaussian blur and 2D gradient calculations. In further work [22, 23, 60], Blake et al also combine their approach with the Kalman Filter, which affords several advantages. One benefit is that the spatial search scale is controlled automatically according to certainty; if no feature is found, the search scale is increased. Also, the temporal scale (i.e. memory) is adaptive; inertia is effectively reduced when features are lost, allowing fast recovery. When features are found, the memory is extended to exploit motion coherence.

Blake, Isard and Reynard [25, 24, 102] went on to develop the adaptive contour from [22, 23] into one able to learn an appropriate dynamical model (by a least squares analysis) and using the six parameters necessary to specify an affine deformation as its state. They also generalized the tracker so that key-frames (prototypical non-affine deformations) could be incorporated into the tracker. The reduction in the size of the state space allowed the tracker to come off specialized hardware and run at frame rate on ordinary desktop workstations.

In another application, Ayache et al [1] use a snake based approach in order to track the mitral valve and left auricle of a heart in ultrasound images. These images are typically very noisy, and so Ayache smoothes them both temporally and spatially. A finite element contour model is used to track the global structures, and additional shape constraints used to localize specific points in the structure. Among other medical tracking applications, the contribution by Jacob et al. [108] on tracking the left ventricle in echocardiographic sequences is noteworthy.

Terzopoulos and Metaxas [177] propose an extension of Snakes to 3D objects. In their work they first review the physically motivated formulation of snake models. They then propose a probabilistic interpretation of the approach that leads to optimal estimation as a means of extracting reliable information from noisy observations. For the purposes of real-time tracking, estimation proceeds sequentially as new observations become available. They show how to construct continuous Kalman filters that incorporate the dynamic snake into their system and prior models. The promising techniques of Kalman snakes for image-based tracking of rigid and, especially, non-rigid objects are forged to create a link between the physical and probabilistic modelling approaches to active vision.

Terzopoulos and Metaxas [178] considered the use of 3D deformable models to track non-rigid and articulated objects, such as human bodies, moving in three-dimensional space. They describe a class of dynamic modelling primitives that can deform locally and globally as they move freely in space. Although the primitives are useful for non-rigid motion tracking per se, the authors enhance their capabilities by applying simulated physical constraints between them. These constraints enable them to automatically construct dynamic models of articulated objects with deformable parts. Differential equations of motion derived using Lagrangian dynamics make the models responsive to applied forces derived from visual data, such as images that are sparse, noisy 3D observations. They employ these differential equations as the system model of a recursive non-rigid motion estimator. The application described employs a sophisticated model of non-rigid dynamics. The estimator synthesizes non-rigid motions using the system model. It expresses the discrepancy between the observations and the estimated model state as generalised forces that formally account for uncertainty in the observations. A Riccati procedure updates a covariance matrix that further transforms the forces due to the current observations in accordance with the system dynamics and the prior observation history.

Cootes and Taylor describe Active Shape Models (ASMs or 'Smart Snakes') [51, 47, 50]: the application to tracking of the Point Distribution Model (PDM) [93]. The approach is similar to Lowe's (discussed in section 2.7) in that image measurements are projected into the model parameter space and parameter errors are then minimised. However, in this case the minimisation is linear least-squares, which has a closed form solution and is thus faster to calculate. The maths involved is further simplified by the fact that the PDM's deformation modes are orthonormal. Also, because there are generally only a few model parameters, this approach is faster than previous snake-like techniques. Performance and speed can be improved further still by employing a multi-resolution search [47, 52] whereby earlier iterations proceed at lower image resolution and fewer shape parameters are allowed to vary, with refinement being permitted in the later stages.

Baumberg and Hogg show how ASMs can be coupled with a Kalman filtering framework to produce a more robust system [12, 10, 14, 11]. This method is very efficient because the filters for each shape parameter can be decoupled, allowing independent filtering of each parameter and thus avoiding large matrix computations.

A recent development in curve tracking is the use of level-set snakes (Paragios and Deriche [142]) to replace traditional B-spline based snakes. An energy function is defined over the image, and fast algorithms are used to track level sets of this function. An advantage of the approach is that the topology of the level sets may change, although there is no parametric representation of the object, so the problem addressed is more akin to motion segmentation than tracking. Also, existing methods have only been applied where background subtraction can be used, and have not been demonstrated in image clutter.

Extensions of 'Snake' type algorithms have been proposed by several other authors. Among them the regularised Gsnake algorithm by Lai et al. [120, 121], the active rays of Denzler et al [62, 63], the finite element based method for snakes and balloons by Cohen et al. [45], the velocity snakes [145, 146] and the PDAF based active contours [147] by Peterfreund are noteworthy contributions.

2.7 Model Based Tracking

Model based tracking primarily requires the characteristics of the object prior to tracking. Tracking techniques are formulated based on a 2D template of an object, or a 3D model of the object being available. The advantage of incorporating model knowledge into tracking is that clutter can be rejected efficiently. The tracking algorithms are formulated assuming that the model deforms or changes shape within acceptable limits from the template or model shape (eg: allow only affine transformation of the template). In the following section we provide a short survey of some popular model based tracking techniques. For clarity, we have informally classified the model based tracking survey into the following groups: 'general model' tracking (can be applied to any model), and 'specific model' tracking (pertains to tracking a particular type of model, eg: vehicles only).

2.7.1 General Model-Template Tracking

A good example of a model based tracker is the RAPID (Real-time Attitude and Position Determination) tracker of Harris et al [84, 86]. A 3D model of an object in the world (the target which is to be tracked) is built by hand. As each video frame arrives, the model is back-projected into the image, using a prediction of its position. The perpendicular distance from image points, lying on high contrast edges, to the predicted positions of such edges are then used as input to a Kalman filter. This filter updates a six degree of freedom model state corresponding to the target's position in the real world. Perpendicular distances are used because the aperture problem only allows motion perpendicular to an image boundary to be determined. The tracker works at video rate, but relies on being able to reliably locate the high contrast edges, and on a well calibrated camera.

Baumberg and Hogg [12] build on the work of Cootes et al [93], providing a way of automatically generating PDM's from video sequences. The models are built by subtracting each video frame from a median filtered background image, and then thresholding to give a binary image of the target. Points are placed equally spaced around the target, and indexed, based on their position relative to the principal axis of the target. Several such point sets are collected over time and the principal components of the shape variation extracted to give the PDM. In [10, 14] Baumberg introduces learnt dynamics into the PDM, connecting parts of the PDM together with springs and dampers. This dynamical model enables a tracker to predict the motion of the target forward through time, allowing tracking to continue in the temporary absence of image measurements.

Baumberg and Hogg [10] also show how to construct temporal models from training sequences using FEM model analysis [143, 144]. The models produced exhibit a number of independent modes of vibration which reflect the motions experienced in the training sequences. These motions can then be used directly as prediction models for tracking, again, within a Kalman filtering framework. The use of modal analysis means that, unlike Blake et al 's model, the Kalman filter can be decoupled for extra speed.

Huttenlocher et al [98] developed a target tracking system based around matching a binary template image of the target. This template image is the output of an edge detector, and is updated each frame. Tracking proceeds by finding the region of a new image which is most likely to contain the template. This is done by computing the generalized Hausdorff distance, between the template and each possible target position within the image. The image location with the minimum Hausdorff distance is taken as being the target's new location. Various fall back strategies and alternative templates are employed to enable the tracker to continue, even if no positions in the image match the template satisfactory.

Huttenlocher et. al. [97] describe a model-based method for tracking non-rigid objects moving in a complex scene. The method operates by extracting two-dimensional models of an object from a sequence of images. The basic idea underlying the technique is to decompose the image of a solid object moving in space into two components: a two-dimensional motion and a two-dimensional shape change. The motion component is factored out and the shape change is represented explicitly by a sequence of two-dimensional models, one corresponding to each image frame. The major assumption underlying the method is that the two-dimensional shape of an object will change slowly from one frame to the next. There is no assumption, however, that the two-dimensional image motion between successive frames will be small.

Lowe [127] developed a computer vision system for real-time motion tracking of 3D objects, including those with variable internal parameters. This system provides for the integrated treatment of matching and measurement errors that arise during motion tracking. These two sources of error have very different distributions and are best handled by separate computational mechanisms. These errors can be treated in an integrated way by using the computation of variance in predicted feature measurements to determine the probability of correctness for each potential matching feature. In turn, a best-first search procedure uses these probabilities to find consistent sets of matches, which eliminates the need to treat outliers during the analysis of measurement errors. The most reliable initial matches are used to reduce the parameter variance on further iterations, thus minimizing the amount of searching required for matching more ambiguous features. These methods allow for much larger frame-to-frame motions than most previous approaches. The resulting system can robustly track models with many degrees of freedom while running on relatively inexpensive hardware. These same techniques can be used to speed up verification during model -based recognition.

2.7.2 Eye Tracking

Yuille and Hallinan [204] explore the problem of accurately locating an object in an image. The object they attempt to locate is the eye. A detailed model is built representing the various parts of the eye (the whites, iris and pupil), and this model has the degrees of freedom of the various parts of the eye built into it - the iris and pupil are allowed to move round the white of the eye together for instance. The model is fitted to an image by performing an energy minimization. This minimization is based around both shape deformation constraints and intensity constraints – the iris and pupil are assigned a low energy when they lie over dark parts of the image and the whites of the eye a low energy when they lie in light regions.

2.7.3 Vehicle Tracking

Another excellent example of a model based tracking system is that developed by Sullivan and Worrall [175, 200, 3]. The problem they attack is the tracking of vehicles. Accurate three-dimensional wire-frame models of prototypical cars are used, together with a model of the behavioural characteristics of the vehicles. The tracker has the ground plane constraint implicitly built in - the only free parameters in the tracker are the X, Y position and the vertical rotation of the car. The tracker predicts a position for the car on the ground plane, and an evaluation score is calculated based on how well the first and second spatial derivatives of the image fit a back-projection of the car model. The predicted pose is then refined by one-dimensional linear searches on each free parameter of the model, evaluating the pose score at each position. The pose with the highest score is then used as input to a Kalman filter. The tracker has been successfully applied to views of cars on a road, and also to the tracking of service vehicles attending to a large aeroplane. In [73] the 3D models are enhanced by principal component analysis of manually sampled car data, allowing cars to be more generically fitted. A more complex model evaluation score is also used, based on the positions and orientations of prominent lines in the image, relative to the model.

Koller et.al. [118] address the problem of occlusion in tracking multiple 3D objects in a known environment. For that purpose they employ a contour tracker based on intensity and motion boundaries. The motion of a contour enclosing the image of a vehicle is assumed to be well describable by an affine motion model with a translation and a change in scale. Contours are represented by closed cubic splines, the position and motion of which are estimated along the image sequence. In order to employ linear Kalman Filters they decompose the estimation process in two filters: one for estimating the affine motion parameters and one for estimating the shape of the contours of the vehicles. Occlusion detection is performed by intersecting the depth ordered regions associated to the objects. The intersection part is then excluded in the motion and shape estimation. Occlusion reasoning also improves the shape estimation in case of adjacent objects where shape estimates can be corrupted by image data of other objects. In this way they obtain robust motion estimates and trajectories for vehicles even in the case of occlusions, as they show in some experiments with real world traffic scenes. This work by the authors follows their previous research in robust vehicle tracking [117].

Dickmanns [66] addresses the problem of real-time guidance of a moving vehicle along roads. Kalman filters are used together with sophisticated non-linear models of the vehicle's motion, camera calibration and the road. The road model is used to determine what features are expected to be found where in the scene, and then this expectation used to direct a feature search towards these regions. Specifically line segments are searched for which match the expected orientation of the road. The difference between the orientations expected and those measured is used as a measure of the likelihood that the measurements originate from the side of the road, and this probability used to weight the inputs to the filter.

2.7.4 Human Tracking

Intille and Bobick [100, 101] discuss a system for tracking American football players as they move around a field, viewed with a panning and tilting zoom camera. The camera motion is assumed to be unknown, however the line markings on the field provide an excellent and consistent set of features from which the plane/plane projectivity between the view of the field, and an overhead view of the field is calculated. A detailed model of football field is built up, and areas of the field which are likely to cause tracking problems (the markings on the field, for example) highlighted. A gray level image of each player on the rectified image of the field is built, and used for correlation matching from one frame to the next. Areas of the image which contain strong features are masked out before the correlation is done, otherwise they tend to pull the template of the players. Variations in the views of the players are accommodated by updating the template at each frame. As each player on the field is being tracked, it is possible to detect when occlusions/collisions are likely to be happening (and hence the correlation search fail) and switch to a different tracking algorithm for that period of time. The algorithm suggested for dealing with potential occlusions is bright spot tracking of the player's helmets. The tracking results presented by Intille are impressive, however, there is a tendency for the adaptive templates to slip off the players. No motion model is used which, while enabling them to track abrupt changes in a player's velocity, means that motion coherence is not exploited.

Haritaoglu, Harwood, and Davies [83] describe a real time visual surveillance system for detecting and tracking people and monitoring their activities in an outdoor environment by integrating real time stereo computation into an intensity based detection and tracking system. Unlike many systems for tracking people, their system makes no use of colour cues, instead employs a combination of stereo, shape analysis and tracking to locate people and their parts, and create models of people's appearance so that they can be tracked through interactions such as occlusions. The authors claim that the system is capable of simultaneously tracking multiple people even with occlusion.

Bregler et. al. [32] describes a probabilistic decomposition of human dynamics at multiple abstractions, and shows how to propagate hypotheses across space, time, and abstraction levels. Recognition in this framework is the succession of very general low level grouping mechanisms to increased specific and learned model based grouping techniques at higher levels. Hard decision thresholds are delayed and resolved by higher level statistical models and temporal context. Low-level primitives are areas of coherent motion found by EM clustering, mid-level categories are simple movements represented by dynamical systems, and high-level complex gestures are represented by Hidden Markov Models as successive phases of ample movements. They show how such a representation can be learned from training data, and apply it to the example of human gait recognition.

Rosales and Sclaroff [160, 161] provide a combined 2D, 3D approach that allows for robust tracking of moving bodies in a given environment as observed via a single un-calibrated video camera. The method combines low level (image processing) and mid-level (recursive trajectory estimation) information obtained during the tracking process. The resulting system can segment and maintain the tracking of moving objects before, during, and after occlusion. At each frame, the system also extracts a stabilized coordinate frame of the moving objects. This stabilized frame can be used as input to motion recognition modules. The approach enables robust tracking without constraining the system to know the shape of the objects being tracked beforehand.

2.7.5 Lip Tracking

In [34], Bregler and Omohundro have applied tracking techniques for speech recognition. A technique for representing and learning smooth nonlinear manifolds was presented and applied to several lip reading tasks. Given a set of points drawn from a smooth manifold in an abstract feature space, the technique is capable of determining the structure of the surface and of finding the closest manifold point to a given query point. They use this technique to learn the "space of lips" in a visual speech recognition task. The learned manifold is used for tracking and extracting the lips, for interpolating between frames in an image sequence and for providing features for recognition. They describe a system based on hidden Markov models and this learned lip manifold that significantly improves the performance of acoustic speech recognizers in degraded environments. Other noteworthy contributions on lip motion include the work of Kaucic et al [114], who have demonstrated the use of real-time lip tracking for audio-visual speech recognition.

2.7.6 Face Tracking

An attractive approach for face tracking was proposed by Lanitis et al. [122, 123]. They describe an application where an active shape model (ASM) (also called a point distribution model (PDM) [50]) is used to track facial features. This PDM is derived from a principal component analysis of the shape variations of an object (the target) in a series of images. In [122] the features making up a face are located by using local gray level models and then used to fit the PDM to the image. This fitted PDM is then used to warp the video image of a face to a canonical frame. An eigenface decomposition is then performed to transform the face image into a set of principal gray level components. These components, together with the parameters of the PDM, describe the entire facial appearance of subjects with only 79 parameters. This work is based on Cootes, Taylor et al 's earlier work on PDM's [49, 50, 52]. Information about the local gray level structure of the target is incorporated into the model fitting procedure by building a simple statistical template of the gray level values along search lines perpendicular to the line sections of the model. The model is then located by searching for positions which minimize a Mahalanobis distance between the gray level template and the image.

Cootes, Edwards, and Taylor [53] recently have demonstrate a novel method of interpreting images using Active Appearance Model (AAM). An AAM contains a statistical model of the shape and grey-level appearance of the object of interest that can generalize to almost any valid example. During a training phase the relationship between model parameter displacements and the residual errors induced between a training image and a synthesized model example is learnt. To match to an image they measure the current residuals and use the model to predict changes to the current parameters, leading to a better fit. A good overall match is obtained in a few iterations, even from poor starting estimates. A successful application of face recognition using AAM is reported in [71].

2.7.7 Articulated Object Tracking

Computer sensing of hand and limb motion is an important problem for applications in human computer interaction and computer graphics. Rehg et al [150, 151] describe a framework for local tracking of self occluding motion, in which one part of an object obstructs the visibility of another. The approach uses a kinematic model to predict occlusions and windowed templates to track partially occluded objects. They present offline 3D tracking results for hand motion with significant self occlusion.

Hogg's well known 'Walker' model [95] is an early example of a non-trivial temporal model. The kinematics are coupled to a pre-learned periodic walk sequence, modelled via a series of cubic B-splines, which is used to derive predictions for plausible object states in each successive frame.

Bregler and Malik [33] also demonstrate a new visual motion estimation technique that is able to recover high degree-of-freedom articulated human body configurations in complex video sequences. They introduce the use of a novel mathematical technique, the product of exponential maps and twist motions, and its integration into a differential motion estimation. This results in solving simple linear systems, and enables the algorithm to recover robustly the kinematic degrees-of-freedom in noise and complex self occluded configurations. They demonstrate this on several image sequences of people doing articulated full body movements, and visualize the results in re-animating an artificial 3D human model. They are also able to recover and re-animate the famous movements of Eadweard Muybridge's motion studies from the last century [33].

Pentland and Horowitz [143] introduce a physically correct model of elastic non-rigid motion. This model is based on the finite element method, but decouples the degrees of freedom by breaking down object motion into rigid and non-rigid vibration or deformation modes. The result is an accurate representation for both rigid and non-rigid motion that has greatly reduced dimensionality, capturing the intuition that non-rigid motion is normally coherent and not chaotic. Because of the small number of parameters involved, this representation is used to obtain accurate overstrained estimates of both rigid and non-rigid global motion. It is also shown that these estimates can be integrated over time by use of an extended Kalman filter, resulting in stable and accurate estimates of both three-dimensional shape and three-dimensional velocity. The formulation is then extended to include constrained non-rigid motion. Examples of tracking single non-rigid objects and multiple constrained objects were also demonstrated.

Several other model based tracking methodologies are also worth mentioning. Among them the LAFTER algorithm of Oliver et al [141], Wren et al.'s Pfinder [201], Wacheter et al.'s human tracking algorithm [197], Cai and Aggarwal's indoor person tracker [39], Heap and Hogg's 3D hand tracking [91, 88], and Birchfield's head tracking algorithm [16] are note worthy contributions.

2.8 Data Association and Tracking Algorithms

In this section we survey some of the tracking algorithms that are employed in computer vision and target tracking research.

2.8.1 Data association

When using visual features to track a target against a cluttered background, there is a significant chance that part of the background may be mistaken for the target. A simple feature or target detection scheme will either correctly identify the target, fail to identify anything or identify part of the background as the target. The scheme may also find that several objects (or parts of the image) meet its criteria for being part of the object being tracked. The data-association problem is to correctly determine which observation (if any) actually corresponds to the target. A whole range of tracking algorithms, to resolve the above issue, has been developed over a number of years. Among them the Kalman Filter (KF), and algorithms which stemmed from KFs, have been popular within the computer vision community [149]. More recently, the proposal of the Condensation algorithm [103, 104] has opened a way for a whole range of new tracking applications. In the following sections we survey some of the well know tracking techniques.

2.8.2 Kalman Filter based Tracking Algorithms

Spatio-temporal estimation, the tracking of shape and position over time, has been dealt with thoroughly by Kalman filtering [5, 6, 74, 199, 130], in the case in which the state's probability density function (p.d.f.) can satisfactorily be modelled as Gaussian ([66], [84], [75], [150]). In this case the Kalman filter can be applied to track image curves ([176, 22, 23, 25]). For the state density to remain Gaussian it is necessary that the prior, process and measurement densities be Gaussian also (in the usual case the measurement and process noise are Gaussian and the update equations are linear, which results in a Gaussian state density as required).

Bar-Shalom and Fortmann [6] describe a number of standard extensions to the Kalman filter for dealing with situations where non-Gaussian densities may be encountered. The Extended Kalman Filter (EKF) is appropriate in the case of a non-linear but unimodal process which can be well approximated over the length of a single time step by its local linearization. The EKF has been used in visual tracking, e.g. ([84, 157]). In some cases an exact filter may be derived even in the case that the dynamics are non-linear, for example Maybank et al. [129] construct a filter for tracking a car based on position and steering angle parameters. The introduction of clutter can cause the observation density to be highly non-Gaussian, by introducing multiple modes corresponding to the clutter features. A multi-modal measurement density necessarily induces multi-modality into the state density.

The "Probabilistic Data Association Filter" (PDAF) (Bar-Shalom and Fortmann [6]) is designed for the case of image clutter where the process is linear and Gaussian, and the observation density is a mixture of Gaussians. The PDAF continues to use the standard Kalman filter framework by approximating all the visible measurements, weighted by their predicted likelihoods, into a single Gaussian-distributed feature, and so it continues to represent the state density as a single Gaussian. When a multi-modal state density is required, one solution is to use a mixture of Gaussians to represent the state density.

The Joint PDAF (JPDAF) (Bar-Shalom and Fortmann [6]) is an extension of the PDAF where in principle the state density is evaluated exactly and represented as a mixture of Gaussians [82]. The number of terms in the mixture increases exponentially, however, so pruning and merging of hypotheses is required to run within a fixed computational bound. A multi-modal process density also results in the state density becoming multi-modal. The Interacting Multiple Model (IMM) filter (Blom and Bar-Shalom [5], [28]) is analogous to the JPDAF when it is the process rather than (or as well as) the observation density which is multi-modal. The details of IMM are further discussed in chapters 4 and 7 of this thesis.

2.8.3 Condensation Algorithm

Isard and Blake's Condensation algorithm [103, 104] provides a much richer environment for temporal prediction. The model state is represented not as a single, deterministic set of model parameters, but as a probability density function over the whole parameter space. This allows for non-Gaussian (arbitrary, in fact) uncertainty and multiple hypotheses. A model of conditional probability (learned from training sequences) is used to propagate the pdf over time. Propagation dynamics are learned from training sequences; Isard and Blake [104] demonstrate the construction of second order models which can predict constant velocity, oscillatory and decaying dynamics. The result is highly robust tracking of agile motion. Notwithstanding the use of stochastic methods, the algorithm is reported to run in near real-time. Further details and evaluation of Condensation is discussed in Chapter 8.

The benefits of Condensation are as follows: It can support multiple hypotheses; this is represented by a pdf with multiple peaks. It recovers well from failure; the stochastic nature of the algorithm allows it to escape from local maxima. It incorporates a level of prediction, which improves the speed of convergence and the quality of results over, for example, a Genetic Algorithm [88].

The prediction aspect of Condensation is embedded in the propagation equations. Currently these have two elements; a deterministic term which allows for simple drifting of the pdf, and a stochastic term which encourages spreading of the pdf. Although the tracker can escape from local maxima (due to the stochastic term), the underlying dynamical model is still based on an assumption of smooth, continuous object movement. Such an assumption is not always valid [88].

2.8.4 Extensions of Condensation

Isard and Blake [107] further improve their Condensation algorithm by introducing a smoothing filter at the output. Clutter can cause the probability distribution to split temporarily into multiple peaks, each representing a different hypothesis about the object configuration. When measurements become unambiguous again, all but one peak, corresponding to the true object position, die out. While several peaks persist, estimating the object position is problematic. 'Smoothing' in this context is interpreted to be a statistical technique of conditioning the state distribution on both past and future measurements once tracking is complete. After smoothing, peaks corresponding to clutter are reduced, since these trajectories eventually die out. The result can be a much improved state-estimate during ambiguous time-steps.

The tracking research community has diverged into two camps; those using low-level approaches which are typically fast and robust but provide little fine-scale information, and those using high-level approaches which track complex deformations in high-dimensional spaces but must trade off speed against robustness. Real-time high-level systems perform poorly in clutter, and initialisation for most high-level systems is either performed manually or by a separate module. Isard and Blake [105] extend their Condensation technique to combine low and high-level information in a consistent probabilistic framework, using the statistical technique of importance sampling combined with the Condensation algorithm. The general framework, which they call the I-Condensation, is applied on a hand tracker which combines colour blob-tracking with a contour model. The resulting tracker is reported to be robust to rapid motion, heavy clutter and hand-coloured distracters, and re-initialises automatically. The system is also claimed to run comfortably, in real time, on an entry-level desktop workstation.

Condensation tracker can also be extended to cope with automatic model-switching for objects that move with variable motion (Isard and Blake [106, 102]). The authors present a significant development of random sampling methods to allow automatic switching between multiple motion models as a natural extension of the tracking process. The Bayesian mixed state framework is described in its generality, and the example of a bouncing ball is used to demonstrate that a mixed state model can significantly improve tracking performance in heavy clutter. The relevance of the approach to the problem of gesture recognition is then investigated using a tracker which is able to follow the natural drawing action of a hand holding a pen, and switches state according to the hand's motion.

As a further improvement to [106], Rittscher and Blake [156] have developed 'Partial Importance Sampling' to enhance the efficiency of the mixed state Condensation filter [106]. They also show that the importance sampling can be done in linear time. 'Tying' of discrete states is used to obtain further efficiency improvements. Automatic segmentation is demonstrated on video sequences of aerobic exercises. Performance is reported to be promising, but there remains a residual mis-classification rate, and possible explanations for this are also discussed in [156].

2.8.5 Applications of Condensation

Black and Jepson [19] propose an incremental recognition strategy that is based on the Condensation algorithm. Gestures are modelled as temporal trajectories of some estimated parameter over time (velocity in this case). The Condensation algorithm is used to incrementally match the gesture models to the input data. The method is demonstrated with an example of an augmented office whiteboard in which a user makes simple hand gestures to grab regions of the board, print them, save them, etc.

Standard techniques (Yule-Walker) are available for learning Auto-Regressive process models of dynamical processes. When sensor noise means that dynamics are observed only approximately, learning has still been achieved via Expectation-Maximisation (EM) together with Kalman Filtering. This cannot handle more complex dynamics, involving multiple classes of motion. For that case, Bake et al [26] demonstrate how EM can be combined with the Condensation algorithm, which is based on propagation of random sample-sets. Experiments have been performed with visually observed juggling, and plausible dynamical models are found to emerge from the learning process.

Existing object tracking algorithms generally use some form of local optimisation, assuming that an object's position and shape change smoothly over time. In some situations this assumption is not valid: the trackable shape of an object may change discontinuously, for example if it is the 2D silhouette of a 3D object. Heap and Hogg [89] propose a novel method for modelling temporal shape discontinuities explicitly. Allowable shapes are represented as a union of (learned) bounded regions within a shape space. Discontinuous shape changes are described in terms of transitions between these regions. Transition probabilities are learned from training sequences and stored in a Markov model. In this way they show how to create wormholes in shape space. Tracking with such models is via an adaptation, of the Condensation algorithm.

2.9 Conclusion

Our survey has revealed certain areas of visual tracking which are insufficiently addressed in the literature. One area that needs consideration is a multiple motion model framework within a tracking system. Though a single model assumption (this is the case for most tracking applications) is computationally efficient, such a tracking system does not cope with multiple motions that are captured within a sequence of images. Another area that is not adequately addressed is the performance analysis of tracking algorithms. Most of the performance analysis techniques given in the literature are specific to a narrow band of applications, and cannot be easily extended for different types of object tracking. Finally knowledge already existing in other areas of engineering (control, signal processing etc.) seems new (unknown) within the computer vision community.

This thesis therefore tries to address most of these issues in two broad categories. The first section of the thesis (Chapters 3, 4, and 5) deals with point feature tracking algorithms and their performances. The second section (Chapters 6, 7, and 8) deals with contour tracking algorithms and their performances. In addressing these areas, we have also considered the multiple motion model framework within a tracking system.

Chapter 3

Assessing the Performance of Corner Detectors for Point Feature Tracking

Abstract

In this chapter we assess the performance of corner feature detecting algorithms for feature tracking applications. We analyze four different types of corner extractors, which have been widely used for a variety of applications. They are the Kitchen-Rosenfeld corner detector [116], the Harris corner detector [85], the Kanade-Lucas-Tomasi detector [190] and the Smith corner detector [173]. We use the corner stability and corner localization properties as measures to evaluate the quality of the features extracted by the 4 detectors. For effective assessment of the corner detectors, we employed image sequences with no motion (simply static image sequences), so that the appearance and disappearance of corners in each frame is purely due to image plane noise and illumination conditions. Such a setup is ideal to analyze the stability and localization properties of the corners. The corners extracted from the initial frame are then tracked (matched) through the sequence using a corner matching strategy. We employed 2 different types of matchers, namely the GVM (Gradient Vector Matcher) and the Product Moment Coefficient Matcher (PMCM). Each of the corner detectors was tested with each of the matching algorithms to evaluate their performance in tracking the features. The experiments were carried out on a variety of image sequences. They included indoor and outdoor sequences.

3.1 Introduction

Low-level descriptors may be broadly classified into three main types: region-based, edge-based and point-based [168].

Regions (or "blobs") [194, 168, 169] normally correspond to smooth surface patches. Tracking such regions is not always easy, since minor differences between frames (due to image noise or image motion) can lead to very different segmentation in consecutive image frames [168]. Despite recent progress (for example: Meyer and Bouthemy [132] tracked convex-hull approximations to region boundaries, Etoh and Shirai [70] used advanced statistical region descriptors), further theoretical and empirical work is needed before reliable region tracking becomes feasible.

Edges are loci of one-dimensional spatial change [168], located where the change in intensity is significant in one direction. They are generally detected by finding either maxima in the first image derivative [40], or zero-crossings in the Laplacian of the Gaussian of the image [79]. Their usefulness in motion algorithms is limited by the "*aperture problem*", which arises from a locally linear expansion of the spatio-temporal image intensity function; without assumptions about the nature of the flow, only the component of flow *perpendicular* to the edge element can be found [68]. Unfortunately, these assumptions invariably lead to inaccuracies in the estimated flow, particularly at motion boundaries [168]. The use of higher order derivatives is unsatisfactory since differentiation accentuates noise. Moreover, until the advent of snakes [113, 24, 27], arbitrarily curving edges were difficult to describe and track, and simultaneous tracking of multiple open edge contours with automatic snake initialization still largely remains an open problem [168, 24] (discussed more in chapters 6-8). An additional problem with tracking edges through image sequences is that edge segments tend to *split* and *merge*, which complicates the tracking process considerably.

Point features are distinctive image points corresponding to objective 3D scene elements that are in most instances accurately locatable and recur in successive images, which makes them explicitly trackable over time. The term "corners" is used to refer to point features that are loci of two-dimensional intensity change, i.e. *second-order features*. This includes points of occlusion (e.g. T, Y and X junctions), structural discontinuities (e.g. L junctions) and various curvature maxima (e.g. texture flecks or surface markings). Corners impose more constraint on the motion parameters than edges, therefore the *full* optic flow field is recoverable at corner locations [168]. Corners are also often more abundant than straight edges in the natural world making them ideal features to track in an indoor and outdoor environment. To find further details on various corner detectors, the reader is referred to [134, 168].

Despite the large amount of material reported in the literature in the area of low level feature tracking, very little has been published in terms of a performance analysis for many of these algorithms. Our primary contribution in this chapter is to evaluate the suitability of corners extracted by 4 different corner detectors for tracking purposes. For point (corner) feature tracking it is essential that corners extracted in each frame be well localized and temporally stable throughout an image sequence. To test these corner properties it is preferred to use static image sequences where object (or camera) motion will not be a concern. Using indoor and outdoor static image sequences with varied levels of illumination, we assess the quality of corners extracted by the *Kanade-Lucas-Tomasi (KLT)*, the *Harris*, the *Kitchen-Rosenfeld* and the *Smith* corner detectors in the presence of varied noise.

A direct comparison of the performance of corner detectors is difficult because establishing ground truth for 'corner points' is non-trivial, particularly for complex scenes (as studied in this chapter). Even for human eyes declaring the best N corners from a complex scene is very difficult and the choice of selection can vary from person to person. Therefore, for each corner detector considered in this chapter, we provide the allowance of selecting 'their best N corners'. The quality of corners extracted by each detector is then assessed against the localization and stability properties (discussed later). The assessment based on these 2 measures will reveal the number of corners that are most resilient to image plane noise.

This chapter is organized as follows. Section 3.2 briefly describes the 4 corner detectors that we analyze. Section 3.3 provides the performance measures that we employ for our analysis. Section 3.4 gives the inter-frame corner matching strategies employed. Section 3.5 derives the tests that are applied for empirical evaluation of the corner detectors. Section 3.6 gives the results and section 3.7 provides a discussion on the evaluation outcome. Finally section 3.8 gives the conclusion.

3.2 Corner (Point) Features as Tracking Tokens

A number of algorithms for corner detection have been reported in recent years [64, 67, 70, 85, 116, 134, 140, 162, 168, 169, 171, 173, 190]. They can be divided into two groups. Algorithms in the first group involve extracting edges and then finding the points having maxima curvature or searching for points where edge segments intersect. The second, and largest group, consists of algorithms that search for corners directly from the grey-level image. In this chapter we focus on the second group of feature detectors.

We decided to assess the performance of the *Kitchen-Rosenfeld* [116], the *Harris* [84,85], the *KLT* [190] and the *Smith* [173] corner detectors when applied on real data. The choice of these 4 corner detectors was made because, the *Kitchen-Rosenfeld* method uses second and first order derivatives in calculating the cornerness value, while *Harris* method uses only first order derivatives. The *Smith* method uses a geometrical criteria in calculating the cornerness value (no derivations are required), while *KLT* detector uses information from an inter-frame point displacement technique to declare corners. Therefore an assessment in terms of the localization and stability properties for these 4 corner detection methods seemed useful when they are applied to a variety of image sequences. The following sub-sections briefly describe the corner detectors employed.

3.2.1 The Kitchen-Rosenfeld Corner Detector

Kitchen-Rosenfeld algorithm [116] is one of the earliest corner detectors reported in the literature, hence it has been used as a bench mark for future researchers developing corner detection algorithms. This algorithm calculates the 'cornerness' value *C* as the produce of the local gradient magnitude and the rate of change of gradient direction. The quantity *C* is given by,

$$C = \frac{\mathbf{I}_{xx}\mathbf{I}_y^2 + \mathbf{I}_{yy}\mathbf{I}_x^2 - 2\mathbf{I}_{xy}\mathbf{I}_x\mathbf{I}_y}{\mathbf{I}_x^2 + \mathbf{I}_y^2} \tag{3.1}$$

where $\mathbf{I}$ is the grey-level value, and $\mathbf{I}_{xx}$ is the second derivative of $\mathbf{I}$, etc. Points in an image are declared corners if the 'cornerness value' meets some threshold requirement (ie. the lower the value of C, better the corner). Many well-known corner detectors use this threshold (eg:[25,10]). The primary reason for considering this detector is to use it as a 'bench mark' to evaluate the performances of the *Harris*, the *KLT*, and the *Smith* corner detectors.

3.2.2 The Harris Corner Detector

This algorithm, known as the *Harris* corner detector [85] is based on an underlying assumption that corners are associated with maxima of the local autocorrelation function. It is less sensitive to noise in the image than most other algorithms, because the computations are based entirely on first derivatives. The algorithm has proved popular due to its high reliability in finding L junctions and its good temporal stability [140], making it an attractive corner detector for tracking. It should be noted that because these algorithms rely on spatial derivatives, image smoothing is often required to improve their performance. While improving the detection reliability, it has been shown that smoothing may result in poor localization accuracy [157]. The *Harris* corner detector was used successfully to detect features for the DROID 3D vision project [85, 84].

The *Harris* corner detector also computes a *cornerness* value, C, for each pixel in an image. A pixel is declared a *corner* if the value of C is below a certain threshold. Where C is calculated as follows:

- Calculate the intensity x-gradient, $\mathbf{I}_x$, and the intensity y-gradients, $\mathbf{I}_y$ using 3x3 convolution masks.
- Calculate $\mathbf{I}_x^2, \mathbf{I}_y^2, \mathbf{I}_x\mathbf{I}_y$.
- Using a Gaussian smoothing kernel of standard deviation σ, calculate the sampled means $<\mathbf{I}_x^2>, <\mathbf{I}_y^2>$, and $<\mathbf{I}_x\mathbf{I}_y>$. See Fig. 3.1.
- Calculate the cornerness value of a pixel, C as follows:

$$C = \frac{<\mathbf{I}_x^2> + <\mathbf{I}_y^2>}{<\mathbf{I}_x^2><\mathbf{I}_y^2> - <\mathbf{I}_x\mathbf{I}_y>^2} \tag{3.2}$$

A good corner is defined as having a small value of C; the *best* corner thus having the lowest value of C. The number of surrounding pixels required to calculate C is determined by the size of the Gaussian smoothing kernel. A 3x3 pixel smoothing kernel gives a 5x5 pixel computation area, a 5x5 pixel smoothing kernel gives a 7x7 pixel computation area, *etc.* (Figure 3.1).

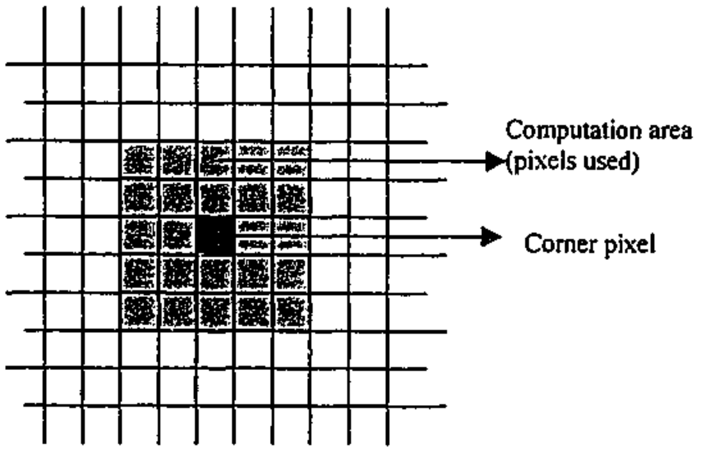


Figure 3.1: Pixel area used when calculating the Harris cornerness C, assuming a Gaussian smoothing kernel of 3x3 pixels.

3.2.3 The Smith (SUSAN) Corner Detector

Smith [173] developed a very simple corner detector that uses no spatial derivatives at all. The *Smith* corner detector does not require any smoothing and so there is no degradation in localization accuracy due to smoothing. This detector has been implemented as part of a scene segmentation algorithm ASSET (A Scene Segmenter Establishing Tracking) [172].

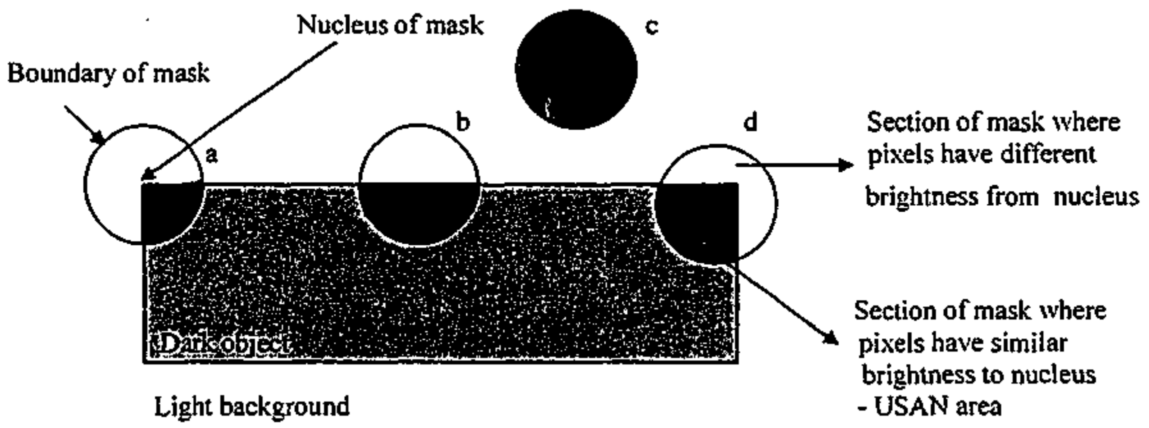

Nucleus of mask
Boundary of mask
a
b
c
d
Section of mask where pixels have different brightness from nucleus
Section of mask where pixels have similar brightness to nucleus - USAN area
Dark object
Light background

Figure 3.2: Four Smith corner finding masks at different positions in an image.

The *Smith* corner detector [173] is different from the other detectors in nature. Each pixel in an image is used as the center of a small circular mask. The greyscale values of all the pixels within this circular mask are compared with that of the center pixel (the *nucleus*). All pixels with *similar* brightness to that of the nucleus are assumed to be part of the same structure in the image.

Figure 3.2 shows the masks with pixels of *similar* brightness to the nucleus coloured black, and pixels with *different* brightness coloured white. Smith calls the black image area the *Univalue Segment Assimilating Nucleus* (USAN). He argues that the USAN corresponding to a corner (case (a) in Figure 3.3) has an USAN area of less than half the total mask area. It is clear from Figure 3.3 that a local minimum in USAN area will find the exact point of the corner.

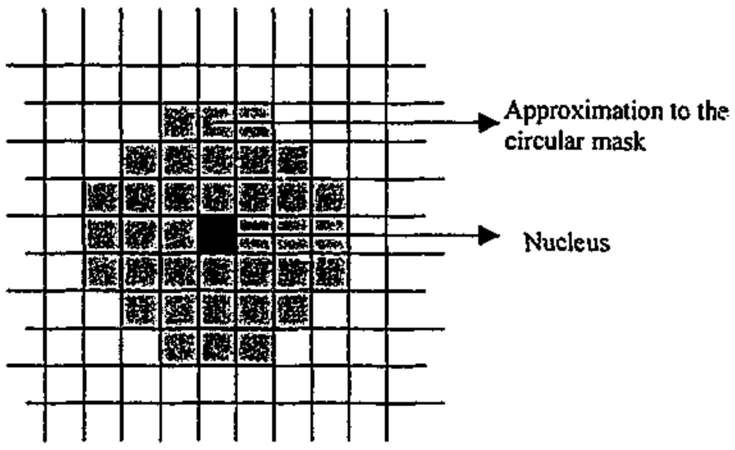


Figure 3.3: The Smith USAN (SUSAN) corner finding mask.

In practice, the circular mask is approximated using a 5 x 5 pixel square with 3 pixels added on to the center of each edge (Figure 3.3). The intensity of the nucleus is then compared with the intensity of every other pixel within the mask using the following comparison function:

$$c(r,r_0) = 100e^{-\left(\frac{I(r)-I(r_0)}{t}\right)^6} \tag{3.3}$$

where r_o is the position of the nucleus, r is the position of any other point within the mask, I*(r)* is the brightness of any pixel, and t is the so-called *brightness difference threshold.* Eq. (3.3) is chosen to allow a pixel's brightness to vary slightly without having too large an effect on c, even if it is near the threshold position. The sixth power is used to obtain the theoretical optimum, see [173] for details. This comparison is done for each pixel in the circular mask, and a running total, n, of the outputs, c, is made:

$$n = \sum_r c(r,r_0) \tag{3.4}$$

The total n is 100 times the USAN's area (the factor of 100 coming from Equation 3.3). The USAN area n is then thresholded to extract the corners. A pixel is declared a corner if its USAN area, n, is less than half the maximum possible USAN area (The maximum USAN area is given by the area of the circular mask times 100, which is 3700). The *geometric threshold*, g, therefore sits at 1850 [(25+12)*100/2]. Smith points out that the value of g affects the shape of the corners detected, and that reducing the value of g results in only the sharpest corners being detected [173]. The brightness difference threshold t, affects the quantity of corners detected by determining the allowed variation in brightness within the USAN.

Finally, an intermediate image is created from the value n calculated for each pixel in the image. If n is greater than the geometric threshold, g, then a zero is placed in the intermediate image, otherwise the value *(g-n(x,y))* is used. The intermediate image is then searched over a square 5 pixel by 5 pixel region for local maxima, and it is these local maxima pixels that are declared corners.

3.2.4 The Kanade-Lucas-Tomasi (KLT) Corner Detector (includes the Tracker)

The KLT corner detector [190] operates by comparing a patch of image information in 2 consecutive frames of an image sequence (developed for the KLT tracking algorithm [190, 171]). It assumes that images taken at near time instants are usually strongly related to each other, because they refer to the same scene taken from only slightly different view points. This property can be explained by the following equation:

$$\mathbf{I}(x, y, t+\tau) = \mathbf{I}(x-\Delta x, y-\Delta y, t)$$

where $\mathbf{I}$ is the image intensity function having 3 parameters (space and time variables x, y & t). The inter-frame displacement $\mathbf{d} = (\Delta x, \Delta y)$ is the displacement of point $\mathbf{x} = (x, y)$ between time instants t and $(t+\tau)$. For the rest of this section, the notation x, y, t are dropped for convenience.

An important problem in finding the displacement $\mathbf{d}$ of a point from one frame to the next is that a single pixel cannot be reliably tracked, unless it has a very distinct character with respect to all of its neighbors. This is because of image plane noise, clutter etc. Because of these problems, KLT does not track a single pixel, but windows of pixels, and windows are looked for that contain sufficient texture. Using small window size is considered important because only small amount of change would have been accounted for within a small area. Any discrepancy between successive windows that cannot be explained by a translation is considered to be an error, and the displacement vector is chosen to minimize this residue error. This is expressed for a given window size W as:

$$\varepsilon = \int_W [\mathbf{I}(\mathbf{x}-\mathbf{d}) - \mathbf{J}(\mathbf{x})]^2 w d\mathbf{x}$$

where $\mathbf{J}(\mathbf{x}) = \mathbf{I}(\mathbf{x}-\mathbf{d}) + n(\mathbf{x})$, with noise n. Assuming the inter-frame displacement vector is small, using Taylor series (truncated to linear term) expansion, the following is possible: $\mathbf{I}(\mathbf{x}-\mathbf{d}) = \mathbf{I}(\mathbf{x}) - \mathbf{g}.\mathbf{d}$. Now the residue equation reduces to,

$$\varepsilon = \int_W (h - \mathbf{g}.\mathbf{d})^2 w d\mathbf{x},$$

where $h = \mathbf{I}(\mathbf{x}) - \mathbf{J}(\mathbf{x})$. Differentiating the residue equation with respect to **d** and setting the result equal to zero provides the following easily solvable expression:

$$\mathbf{Gd} = \mathbf{e},$$

where the 2x2 matrix $\mathbf{G} = \int_W \mathbf{g}\mathbf{g}^T w dA$, and the 2 dimensional vector $\mathbf{e} = \int_W (\mathbf{I} - \mathbf{J})\mathbf{g} w dA$. With these expressions, **d** can be evaluated (see [190] for complete details). For a stable system, the 2x2 coefficient matrix **G** must be both above the image noise level and be well-conditioned. In turn, the noise requirement implies that both eigenvalues of **G** must be large, while the conditioning requirement means that they cannot differ by several orders of magnitude. If the two eigenvalues of G are λ_1 and λ_2, then a corner is accepted in a window if $\min(\lambda_1, \lambda_2) > \lambda$, where λ is a predefined threshold. The KLT corner detector and tracker (the process of corner detection and tracking are interrelated) complement each other and have been reported to perform well [190]. In our implementation of KLT, we independently extracted corners from each frame, thus eliminating any bias that might be introduced by the KLT tracker.

3.3 Performance Measures for Assessing the Quality of Corners for Tracking

A requirement for point feature tracking is that, having found corners in one frame, the same corners should be found and matched in successive frames, thereby constructing a time history of corners and allowing their motion to be analyzed. The ability to consistently find and match corners in this way relies on the corners being *temporally stable* and *well localized* [157]:

- **Good temporal stability** - corners should appear in every frame of a sequence (from the time they are first detected), and should not *flicker* (turn on and off) between frames.

- **Accurate localization** - the calculated image-plane position of a corner, given by the detector, should be as close to the *actual* position of the corner as possible.

Apart from the above two properties that are crucial for tracking features, a good corner detector should also be robust with respect to noise, and be efficient (computationally cheap to calculate) to run in real-time (or near real-time).

The *Kitchen-Rosenfeld*, The *KLT*, the *Harris* and the *Smith* corner detectors have been used for tracking applications in the past and have been reported to have good corner detection properties [134]. The *Harris* detector was originally developed as part of the DROID 3D vision algorithm [84] and was designed to be temporally stable. The *Smith* detector was used in the ASSET series projects [172]. ASSET used the 2D image-plane flow of corners to segment a scene into independently moving objects. The *Kitchen-Rosenfeld* detector is widely reported and has been used in many varied applications [162, 67]. The *Lucas-Kanade-Tomasi* detector was employed for the KLT tracker [190] successfully. Cox *et al* have also used a variant of this detector for their MHT tracker [58]. All four corner detectors were reported to perform well as part of their respective motion algorithms.

In this chapter, we use the corner localization and corner temporal stability properties to assess the quality of corner detectors for tracking applications (it is worth noting here that the internal parameters of each corner detector were adjusted to give the best possible result). A common ground to assess these corner detectors was essential. The best possible scenario was to use static image sequences. By definition, static scenes contain no moving objects and therefore no moving corners. If images could be captured with zero noise, all corners in a static scene would remain completely stationary in the image and would be *seen* in every frame. Unfortunately, this is not the case, and noise is always present in an image. A static scene is therefore an ideal way to assess the performance of corner detectors, because the motion induced by the movements of the camera are known to be zero and any failures of the detectors are due entirely to image-plane (sensor) noise.

The experiments were carried out on a variety of image sequences. First, an indoor image sequence of a toy dog is used with only artificial light interference (page 51). Secondly, an outdoor sequence of a building is considered (illuminated only with natural lighting, page 56). Then we considered an indoor lab sequence (page 60) with plenty of identifiable corners (also contained direct light sources). Finally, we used a computer image sequence (page 62) with lots of light reflections and curved objects. All four sequences (30 frames in length) were static (with no motion), so that the appearance and disappearance of corners in each frame was purely due to image plane noise and illumination conditions. Such a setup is ideal to analyze the stability and localization properties of the corners. Finally we considered 2 sequences with very small motion (about a pixel per frame motion) to assess the robustness of the corner detectors under very small motion (the Coke and the Rubic sequences, see Appendix G). Since the analysis was based on an automated tracking process, matching a corner in every frame required a suitable corner matcher (we avoided manual matching to emulate a true tracking scenario). In the following section we discuss two corner matchers, and later in the chapter we evaluate their effectiveness (reliability) for corner matching.

3.4 Matching Corners

The feature tracking process is implemented by first extracting corner features from every frame of a static image sequence using a corner detector, and then finding a match between every corner in the initial frame and the subsequent frames. Therefore, it is important to employ a reliable feature matching strategy to correlate features between frame (no special tracking algorithms were required for this experiment).

Feature motion prediction is never completely accurate due to image noise, poor motion prediction or random motions of the camera. It is therefore very common to search for a matched feature in a *Region of Interest (ROI)*, around the predicted position of the feature in the image-plane [168]. The simplest method of corner matching is to declare the strongest corner within the ROI as being the same corner feature as the one in the previous frame using only corner positional information (commonly known as the nearest neighbor block matching technique). Although this is computationally very efficient, it is not very robust due to the presence of noise, and more significantly due to the presence of other (stronger) corners that may enter the *ROI*. A more reliable matching scheme is therefore required to prevent mismatches occurring. In the following sub-sections we discuss 2 matching schemes that have been successfully employed in many tracking applications. They are the Gradient Vector Matcher [84, 157] and the Product Moment Coefficient Matcher [168].

3.4.1 Gradient Vector Matcher (GVM)

The GVM was developed as part of the DROID project [84]. The DROID algorithm generated a match confidence by comparing the image-plane intensities and spatial gradients of the corner pixels to be matched. All corners with a low value of C (*i.e.* strong corners) are considered candidate corners for a match with the current corner. The philosophy behind this matcher is that as much of the information as possible already available via the feature extraction process should be used. Three image attributes are compared, these being $\mathbf{I}$ (grey-level intensity), $\mathbf{I}_x'$ (*average* grey-level x-gradient), and $\mathbf{I}_y'$ (*average* grey-level y-gradient). $\mathbf{I}_x'$ and $\mathbf{I}_y'$ are calculated by taking the square roots of $< \mathbf{I}_x^2 >$ and $< \mathbf{I}_y^2 >$ at each pixel, and by averaging over a 3 x 3 neighborhood.

Grey level intensities tend to vary from frame to frame, and so directly comparing $\mathbf{I}, \mathbf{I}_x$ and $\mathbf{I}_y$ of the candidate corner's pixels is not very robust. Most cameras have an automatic iris which regulates the amount of light falling on to the CCD, preventing it from becoming flooded. The implication of this is that bright objects in one part of an image will effect the grey-level values of objects in other regions of the same image. A vector is constructed from the three components, $\mathbf{I}$, $\mathbf{I}_x'$ and $\mathbf{I}_y'$ and compared to the equivalent vector of the current corner (see Fig. 3.4). A match confidence value m(v,w) may then be calculated by comparing the normalised magnitude of the difference vectors between each candidate corner vector (**w**) and the current corner's vector (**v**). As shown by Equation (3.5).

$$m(\mathbf{v},\mathbf{w}) = \frac{|\mathbf{v}-\mathbf{w}|}{\sqrt{|\mathbf{v}||\mathbf{w}|}} \tag{3.5}$$

Because linear changes in $\mathbf{I}$ result in linear changes in both $\mathbf{I}_x'$ and $\mathbf{I}_y'$, this method is invariant to linear changes in lighting conditions. The candidate corner which has the minimum value of m(v,w), as long as it is below a predefined threshold, is then declared the matched corner. This threshold therefore sets the *quality* of the match.

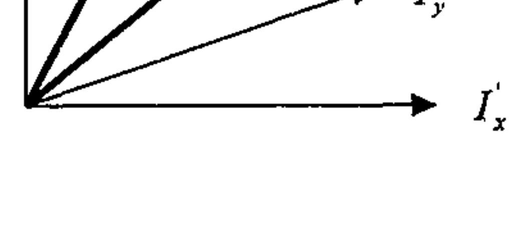


Figure 3.4: The Gradient Vector Matcher (GVM) - match vector

3.4.2 Product Moment Coefficient Matcher (PMCM)

Shapiro, *et al.* [168] used a template matching technique to find corner matches. The confidence measure used was the *product moment coefficient*, given by Equation (3.6).

$$cor = \frac{\sum_{i=1}^{n}(t_i - \bar{t})(p_i - \bar{p})}{\sqrt{\sum_{i=1}^{n}(t_i - \bar{t})^2 . \sum_{i=1}^{n}(p_i - \bar{p})^2}}, \qquad -1 \le cor \le 1, \qquad (3.6)$$

Where t_i and p_i are the intensity values of the template and patch respectively (the template is the area of image about the corner to be matched, and the patch is the area of image about the candidate corner pixel in the subsequent image), $\bar{t}$ and $\bar{p}$ are their means. As with the gradient vector matcher, this measure is also invariant to lighting changes and therefore compares the *structure* of the patches, rather than their absolute intensities. Only positive values of *cor* are considered since a negative value would imply an intensity inversion. A perfect correlation is obtained when *cor* = 1. As with the GVM, a threshold is used to set the quality of the matches. Only matches having a value of *cor* above this threshold are therefore considered successful matches.

3.5 The Localization and Stability Tests of Corners

The temporal stability test/comparison was constructed so that a stable corner was defined as a corner that had been successfully tracked throughout the image sequence from the first frame until the current frame. The localization accuracy test differed from the temporal stability test, in that corners that had been successfully tracked for *d* frames (*d = 3* in the experiments reported in this chapter) up to and including the current one were used to compare the positional accuracy of the corner detectors. The temporal stability (number of stable corners) and the localization accuracy (corner displacement-CD) measures were defined as follows:

The number of stable corners is defined by Eq. (3.7),

$$\text{No. of stable corners} = \sum_{t=1}^{F}\sum_{i=1}^{N} a(i,t), \qquad (3.7)$$

where,

$$a(i,t) = \begin{cases} 1 \text{ if the } i\text{-th corner has been tracked for } t \text{ frames} \\ 0 \text{ otherwise} \end{cases}$$

where N equals the total number of corners and F the total number of frames in the image sequence.

The Corner Displacement-CD (pixels) is given by Eq. (3.8) for the t-th frame.

$$CD_t = \frac{\sum_{i=1}^{N} \{b(i,t) \times ([x_i(t) - x_i(t-1)]^2 + [y_i(t) - y_i(t-1)]^2)^{1/2}\}}{\sum_{i=1}^{N} b(i,t)} \tag{3.8}$$

where.

$$b(i,t) = \begin{cases} 1 \text{ if the } i\text{-th corner has been tracked for } d \text{ frames, and has appeared in frames } t\text{-}1 \text{ and } t \\ 0 \text{ otherwise} \end{cases}$$

The mean corner displacement – μ, variance of corner displacement - σ^2, the percentage of stable corners (γ) successfully tracked, and the number of corner matches in a frame are also useful indicators of the overall performance of each of the corner detectors. These measures are defined as given below.

The Mean Corner Displacement (MCD):

$$\mu CD = \frac{1}{(F-d)} \sum_{t=d}^{F} CD_t \tag{3.9}$$

The variance of corner displacement (σ^2):

$$\sigma^2(CD) = \frac{1}{(F-d)} \sum_{t=d}^{F} (CD_t - \mu CD)^2 \tag{3.10}$$

Note: The value d (taken to be 3 in this chapter, but can be set to any value) appears because we declare a track valid only if it has been tracked for more than d frames.

The percentage of stable corners at the *t*-th frame:

$$\gamma_t = \left(\frac{Number\ of\ stable\ corners\ in\ frame\ t}{Total\ no\ of\ corners\ in\ frame\ 1} \right) \times 100 \tag{3.11}$$

The number of corner matches:

The number of corners found in each frame of a sequence that appeared in the initial frame (not necessarily stable corners). The mean matches - $\mu(mat)$, is simply the average of matches across the image sequence, and $\sigma^2\ (mat)$, is the variance for the corner matches.

These overall performance measures are calculated for each corner detector using both matching methods (GVM and PMCM) for the test image sequences considered.

3.6 Results

1. **Corner stability result**: The corner stability result reveals the number of stable corners identified throughout the sequence. In other words, the corners that appeared in every frame of the sequence are considered stable corners. The result is given as a percentage of the total number of corners extracted in the initial frame.

2. **First frame corner matches result**: This quantity indicates the number of corners found in the initial frame that appeared in the subsequent frames. These corners might not have appeared in every frame (appeared and disappeared throughout the sequence), but might have made their appearance in most number of frames. The mean of this quantity will give the average number of corners that were matched throughout the sequence. The first frame corner matches along with corner stability result gives a good indication of the corner detector's stability property.

3. **Corner displacement result**: The corner displacement result reveals the displacement of a corner in the *n*-th frame from its position in the initial frame (assuming that the corner considered appeared in the *n*-th frame). If the corner considered did not make an appearance in any one of the subsequent frames, then a value of $CD = 3$ pixels (displaced by 3 pixels) are assigned, indicating poor localization of that corner. The mean corner displacement value will indicate the average displacement of a corner from its initial position. An important aspect to notice for this test is the

assumption that the positions of the corners in the first frame of a sequence are considered correct. Such an assumption may not be correct from a theoretical point of view, but from a practical sense, it is acceptable. This is because one would normally like to track object features from the initial frame that they view, therefore it makes sense to make the positions of the corners in the initial frame as the reference positions.

The following experiments are carried out using the above measures to assess the quality of corners extracted by the 4 corner detectors.

Experiment 1 - General performance: The above properties (1-3) are measured for each of the sequences considered.

Experiment 2 - Performance under noise: Uncorrelated Gaussian noise is added to each frame of a sequence at a specified level (experiments were carried out with noise variance ranging from 0 – 25). The 4 corner detectors are applied on the noisy sequence, and the 3 corner properties are observed at each noise level considered.

Experiment 3 - Performance with very small motion: Two sequences (the Coke and the Rubic sequence) with very small motion (around 1 pixel inter-frame motion) are considered (The results are reported in *Appendix A*). The 3 corner properties are measured to assess the quality of corners extracted by each detector when a small motion is present.

The complete results for all the experiments carried out are tabulated in Tables 3.1 – 3.12. The results are also quantitatively and qualitatively displayed in Figures 3.5 - 3.14.

Dog Sequence Results (Comparing 2 different patch sizes)

Detectors		*Harris*		*KLT*		*Kitchen-Rosenfeld*		*SUSAN*	
Threshold		*gvm-0.009 / pmcm-0.7*		*gvm-0.009 / pmcm-0.7*		*gvm-0.009 / pmcm-0.7*		*gvm-0.009 / pmcm-0.7*	
Patch size		**5x5**	**7x7**	**5x5**	**7x7**	**5x5**	**7x7**	**5x5**	**7x7**
GVM	μ (CD)	0.4928	0.4928	0.3038	0.3038	0.3875	0.3875	1.1894	1.1894
GVM	σ^2(CD)	0.0073	0.0073	0.0017	0.0017	0.0036	0.0036	0.0141	0.0141
GVM	μ (mat)	91.6897	91.6897	97.2759	97.2759	92.1379	92.1379	81.5862	81.5862
GVM	σ^2(mat)	2.69	2.69	0.61	0.61	1.70	1.70	4.44	4.44
GVM	γ (%)	82.5	82.5	93.0	93.0	79.8	79.8	62.2	62.2
PMCM	μ (CD)	0.4465	0.4523	0.3329	0.3323	0.3407	0.3407	1.3716	1.3517
PMCM	σ^2(CD)	0.0034	0.0034	0.0010	0.0006	0.0005	0.0005	0.0051	0.0041
PMCM	μ (mat)	91.8276	92.2069	94.6897	94.2069	93.0345	93.0345	74.7241	74.5500
PMCM	σ^2(mat)	2.76	2.71	1.38	1.06	1.89	1.89	6.61	5.69
PMCM	γ (%)	82.5	82.5	88.0	88.0	82.8	82.8	49.0	50.0

Table 3.1: Dog sequence result. The performance of the 4 corner detectors using different image patch sizes to match the corners in successive frames.

Building Sequence Results (Comparing 2 different patch sizes)

Detectors		*Harris*		*KLT*		*Kitchen-Rosenfeld*		*SUSAN*	
Threshold		*gvm-0.009 / pmcm-0.7*		*gvm-0.009 / pmcm-0.7*		*gvm-0.009 / pmcm-0.7*		*gvm-0.009 / pmcm-0.7*	
Patch size		**5x5**	**7x7**	**5x5**	**7x7**	**5x5**	**7x7**	**5x5**	**7x7**
GVM	μ (CD)	0.5515	0.5515	0.2756	0.2756	0.8817	0.8817	1.2463	1.2463
GVM	σ^2(CD)	0.0029	0.0029	0.0010	0.0010	0.0073	0.0073	0.0048	0.0048
GVM	μ (mat)	130.68	130.68	141.03	141.03	115.89	115.89	118.31	118.31
GVM	σ^2(mat)	5.17	5.17	3.82	3.82	32.50	32.50	18.28	18.28
GVM	γ (%)	76.82	76.82	84.6	84.6	53.7	53.7	49.0	49.0
PMCM	μ (CD)	0.5779	0.5029	0.4509	0.4160	0.8164	0.8013	1.3986	1.3366
PMCM	σ^2(CD)	0.0012	0.0005	0.0008	0.0008	0.0078	0.0070	0.0098	0.0061
PMCM	μ (mat)	126.62	129.27	130.48	133.51	114.20	114.75	103.41	106.00
PMCM	σ^2(mat)	7.54	6.61	7.76	5.00	32.78	29.35	23.55	21.58
PMCM	γ (%)	70.2	72.8	70.0	74.0	51.6	53.0	38.4	41.7

Table 3.2: Building sequence result. The performance of the 4 corner detectors using different image patch sizes to match the corners in successive frames.

Dog Sequence Results (effect of synthetic noise)

Corner Detector		KLT											
Patch Size		5 x 5											
Noise Variance		σ = 0		σ = 5		σ = 10		σ = 15		σ = 20		σ = 25	
Threshold		T1	T2	T1	T2	T1	T2	T1	T2	T1	T2	T1	T2
GVM	μ (CD)	0.6154	0.3038	1.6935	0.5753	2.5644	1.0194	2.8511	1.3481	2.9669	1.7534	2.9992	1.9681
GVM	σ^2(CD)	0.0011	0.0017	0.0229	0.0058	0.0150	0.0121	0.0080	0.0161	0.0018	0.0094	0.0000	0.0079
GVM	μ (mat)	85.82	97.27	62.58	93.37	37.65	87.86	20.75	82.44	13.86	77.17	9.55	71.93
GVM	σ^2(mat)	3.65	61.36	14.58	3.26	26.63	4.80	10.04	8.86	6.18	11.38	9.2818	6.13
GVM	γ (%)	69.0	93.0	27.0	80.0	8.0	67.0	1.0	55.0	0.0	44.0	0.0	42.0
PMCM	μ (CD)	0.3329	0.4463	0.8426	1.6325	1.5879	2.6482	2.3133	2.9498	2.7082	3.0000	2.8887	3.0000
PMCM	σ^2(CD)	0.0010	0.0007	0.0034	0.0166	0.0072	0.0128	0.0047	0.0002	0.0047	0.0000	0.0024	0.0000
PMCM	μ (mat)	94.68	90.24	83.89	60.68	62.06	29.17	43.03	8.96	26.10	2.24	16.20	0.5517
PMCM	σ^2(mat)	1.38	4.11	2.78	14.21	6.47	10.07	9.96	5.13	3.54	2.25	6.85	0.5922
PMCM	γ (%)	88.0	77.0	65.0	32.0	40.0	7.0	21.0	1.0	9.0	0.0	2.0	0.0

Table 3.3: Indoor Dog Sequence result using KLT: Comparison of GVM (T1=0.0007, T2=0.009) and PMCM (T1=0.7, T2=0.9) performances at varied noise levels for the KLT corner detector. The corner displacement (CD), number of matches (mat) and the percentage of features at the end of tracking γ) are performance measures for the comparison.

Corner Detector		Harris											
Patch Size		5 x 5											
Noise Variance		σ = 0		σ = 5		σ = 10		σ = 15		σ = 20		σ = 25	
Threshold		T1	T2	T1	T2	T1	T2	T1	T2	T1	T2	T1	T2
GVM	μ (CD)	0.3934	0.3403	2.0322	1.2753	2.5305	1.6972	2.3553	2.1034	2.9470	2.5024	2.9420	2.6703
GVM	σ^2(CD)	0.0104	0.0083	0.1415	0.0256	0.0984	0.0120	0.1880	0.0247	0.0008	0.0089	0.0029	0.0050
GVM	μ (mat)	83.24	90.41	38.24	80.65	35.34	74.48	25.24	68.86	16.20	58.13	14.75	44.03
GVM	σ^2(mat)	6.45	1.27	28.99	8.91	16.91	7.83	24.39	19.77	10.44	19.36	10.73	19.06
GVM	γ (%)	68.0	86.0	8.0	60.0	5.0	46.0	3.0	34.0	2.0	23.0	1.0	14.0
PMCM	μ (CD)	0.3641	0.3114	1.2724	1.7583	1.5822	2.2304	2.0836	2.2716	2.7022	2.9958	2.6502	2.9543
PMCM	σ^2(CD)	0.0060	0.0038	0.0742	0.2150	0.0293	0.0764	0.0744	0.1445	0.0161	0.0001	0.0068	0.0028
PMCM	μ (mat)	90.96	88.48	73.24	24.51	71.72	35.89	56.13	13.10	38.89	3.86	37.17	19.58
PMCM	σ^2(mat)	2.51	3.00	19.01	7.56	12.06	25.81	31.84	7.05	15.61	2.94	12.69	14.03
PMCM	γ (%)	86.0	79.0	41.0	7.0	37.0	8.0	19.0	2.0	6.0	0.0	10.0	0.0

Table 3.4: Indoor Dog Sequence result using Harris: Comparison of GVM (T1=0.0007, T2=0.009) and PMCM (T1=0.7, T2=0.9) performances at varied noise levels for the Harris corner detector. The corner displacement (CD), number of matches (mat) and the percentage of features at the end of tracking γ) are performance measures for the comparison.

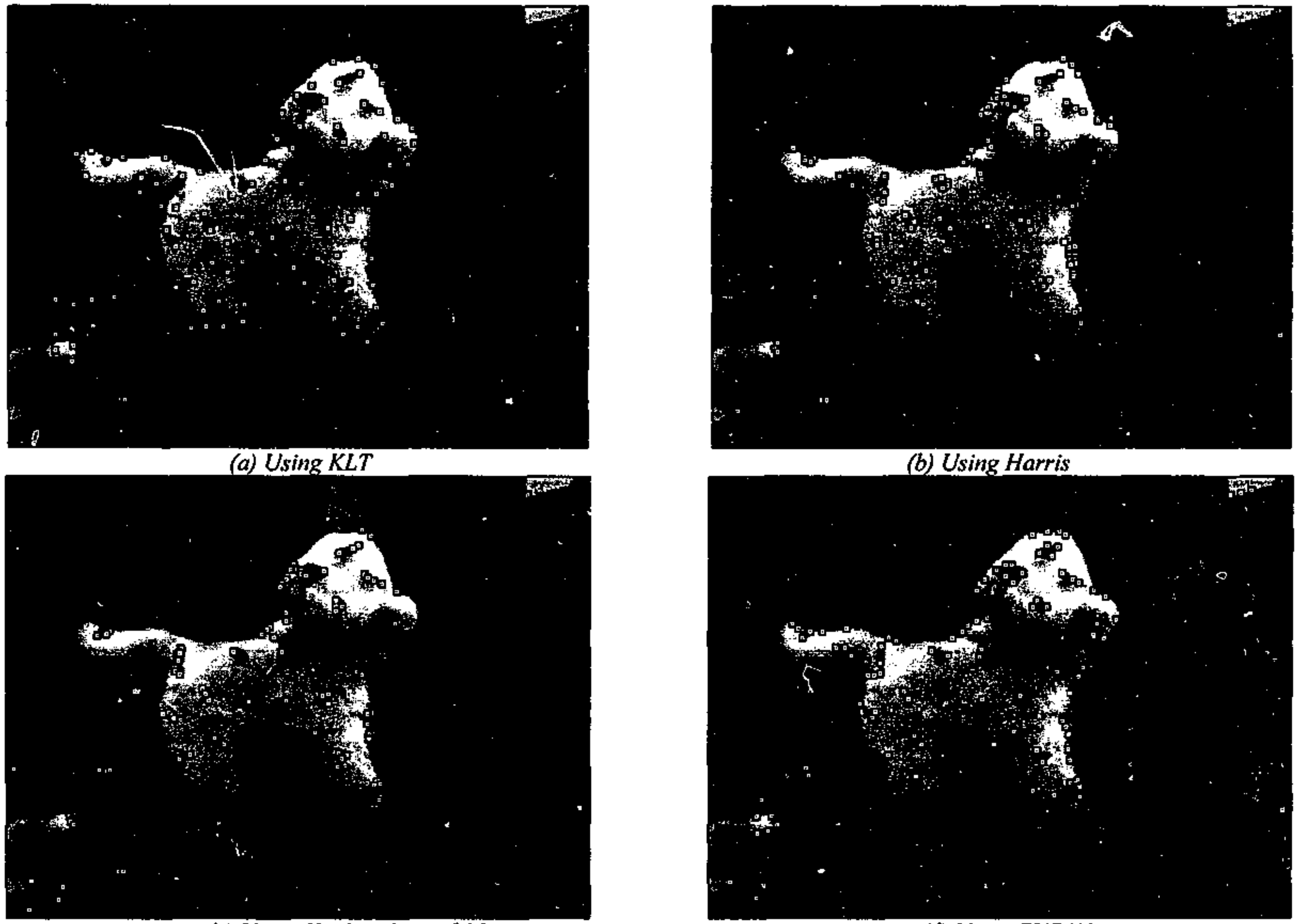
Dog Sequence Results (effect of synthetic noise) Cont.

Corner Detector		*Kitchen-Rosenfeld*											
Patch Size		5 x 5											
Noise Variance		$\sigma = 0$		$\sigma = 5$		$\sigma = 10$		$\sigma = 15$		$\sigma = 20$		$\sigma = 25$	
Threshold		**T1**	**T2**	**T1**	**T2**	**T1**	**T2**	**T1**	**T2**	**T1**	**T2**	**T1**	**T2**
GVM	μ(CD)	0.3205	0.3814	2.2135	1.4076	2.5095	2.0331	2.4975	2.2025	2.6869	2.5025	2.9684	2.8622
GVM	σ^2(CD)	0.0100	0.0083	0.1487	0.0411	0.1623	0.0207	0.1094	0.0391	0.0280	0.0290	0.0008	0.0058
GVM	μ(mat)	86.68	92.13	38.27	71.96	36.10	63.13	24.13	54.96	15.58	46.13	13.37	32.10
GVM	σ^2(mat)	2.21	1.70	19.57	13.48	21.12	19.22	18.18	14.37	8.86	16.32	18.02	30.36
GVM	γ(%)	74.7	79.8	8.1	46.4	3.0	30.3	3.0	24.24	1.0	13.1	0.0	5.0
PMCM	μ(CD)	0.2779	0.3616	1.4155	2.1143	1.9138	2.2344	2.2036	2.5165	2.8421	3.0000	2.8773	2.9543
PMCM	σ^2(CD)	0.0075	0.0176	0.0802	0.2044	0.0541	0.0811	0.0798	0.1951	0.0091	0.0000	0.0073	0.0029
PMCM	μ(mat)	93.03	89.86	63.17	20.10	60.86	33.58	43.44	11.44	28.20	2.3793	25.93	13.68
PMCM	σ^2(mat)	1.89	3.08	11.86	8.98	19.63	27.22	15.00	5.14	18.85	2.2354	21.65	23.38
PMCM	γ(%)	82.8	74.7	31.3	5.0	25.2	6.0	15.1	1.0	4.0	0.0000	4.0	1.0

Table 3.5: Indoor Dog Sequence result using KitchenRosenfeld: Comparison of GVM (T1=0.0007, T2=0.009) and PMCM (T1=0.7, T2=0.9) performances at varied noise levels for the KitchenRosenfeld corner detector. The corner displacement (CD), number of matches (mat) and the percentage of features at the end of tracking (γ) are performance measures for the comparison.

Corner Detector		*SUSAN*											
Patch Size		5 x 5											
Noise Variance		$\sigma = 0$		$\sigma = 5$		$\sigma = 10$		$\sigma = 15$		$\sigma = 20$		$\sigma = 25$	
Threshold		**T1**	**T2**	**T1**	**T2**	**T1**	**T2**	**T1**	**T2**	**T1**	**T2**	**T1**	**T2**
GVM	μ(CD)	1.4463	1.1466	2.8709	2.3452	2.9810	2.5889	2.8080	2.7320	3.0000	2.9413	3.0000	2.9667
GVM	σ^2(CD)	0.0847	0.0222	0.1195	0.0211	0.0046	0.0271	0.2359	0.0453	0.0000	0.0045	0.0000	0.0031
GVM	μ(mat)	60.82	81.58	19.00	69.10	10.58	54.34	6.27	46.10	3.62	36.68	3.96	34.51
GVM	σ^2(mat)	11.72	4.44	28.62	19.67	12.72	9.19	7.44	16.78	6.51	34.48	3.75	19.28
GVM	γ(%)	28.5	62.2	0.0	21.4	0.0	11.2	0.0	7.1	0.0	4.0	0.00	1.0
PMCM	μ(CD)	1.1751	1.3405	2.5527	2.9301	2.8621	2.9575	2.8499	2.9259	3.0007	3.0000	2.9932	2.9991
PMCM	σ^2(CD)	0.0232	0.0428	0.0726	0.0375	2.0203	0.0147	0.1057	0.1427	0.0001	0.0000	0.0004	0.0000
PMCM	μ(mat)	74.72	62.51	40.72	12.20	31.75	11.0	19.72	2.34	11.48	0.31	20.10	6.34
PMCM	σ^2(mat)	6.61	15.90	30.68	18.64	18.32	7.10	25.99	2.70	16.04	0.62	16.92	12.15
PMCM	γ(%)	48.9	31.6	5.1	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 3.6: Indoor Dog Sequence result using SUSAN: Comparison of GVM (T1=0.0007, T2=0.009) and PMCM (T1=0.7, T2=0.9) performances at varied noise levels for the SUSAN corner detector. The corner displacement (CD), number of matches (mat) and the percentage of features at the end of tracking γ) are performance measures for the comparison.


(a) Using KLT

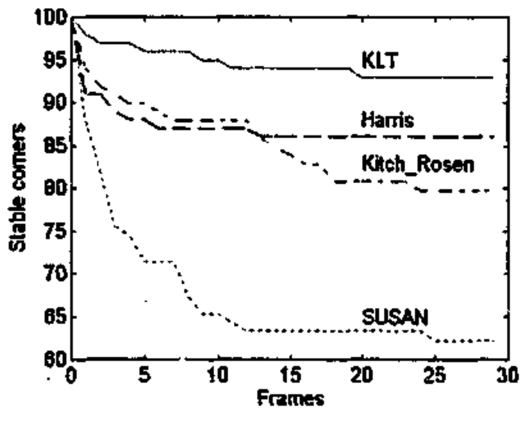
(b) Using Harris

(c) Using Kitchen-Rosenfeld

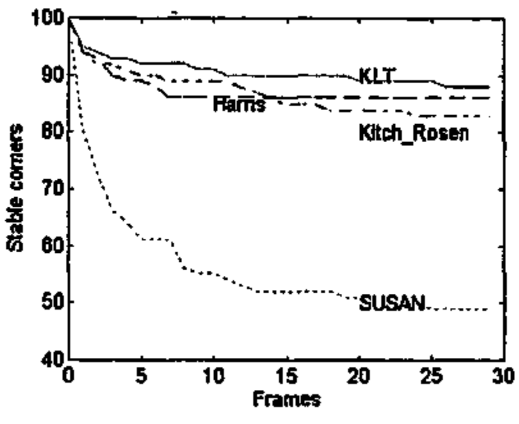
(d) Using SUSAN

Figure 3.5: The best 100 corners extracted from the indoor static dog sequence. (a) Using KLT corner detector. (b) Using Harris corner detector. (c) Using Kitchen-Rosenfeld corner detector. (d) Using SUSAN corner detector.

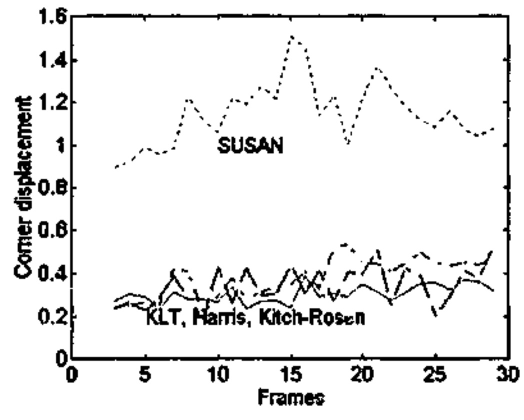
Dog Sequence Results (Performance vs. Number of Frames)



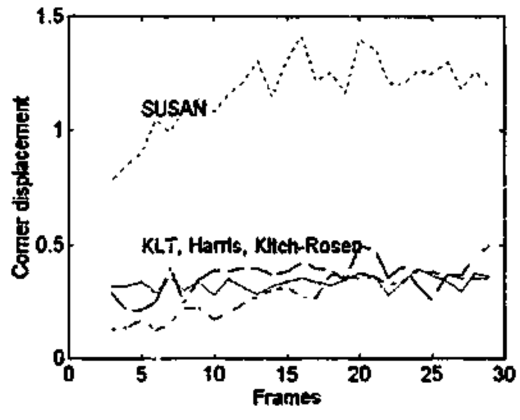
(a) Using GVM (Thresh=0.009)



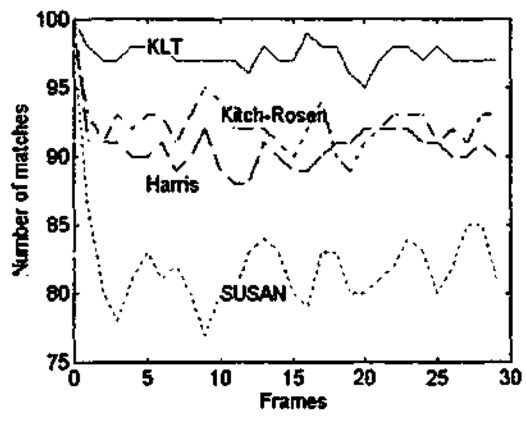
(b) Using PMCM (Thresh = 0.7)



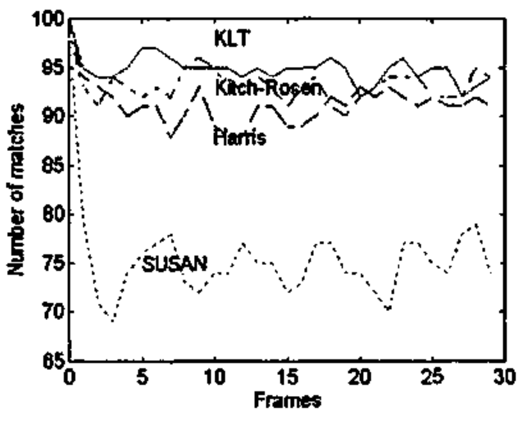
(c) Using GVM (Thresh=0.009)



(d) Using PMCM (Thresh = 0.7)



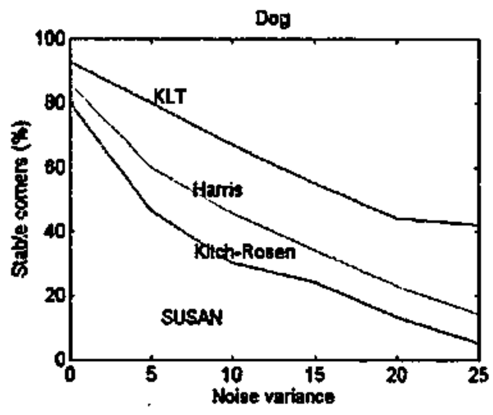
(e) Using GVM (Thresh=0.009)



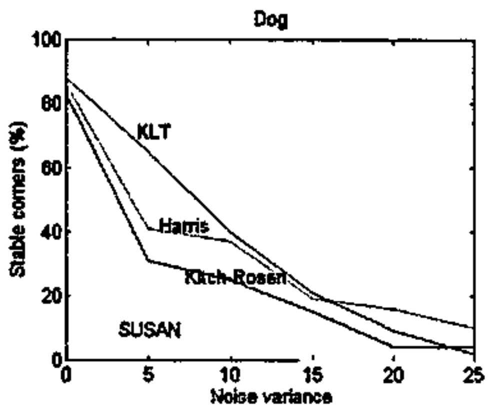
(f) Using PMCM (Thresh = 0.7)

Figure 3.6: Corner detector performance-test for the ***'static dog'*** *sequence (the best 100 corners as seen by each detector are extracted from each frame). (a) The KLT, Harris, Kitchen-Rosenfeld and SUSAN corner detectors are assessed for stable corners (percentage) using GVM matcher. (b) Stable corners compared using the PMCM matcher. (c) Corner displacement test using GVM matcher. (d) Corner displacement test using PMCM matcher. (e) Number of successful first frame corner matches using GVM. (f) Number of successful first frame corner matches using PMCM. In all 3 tests the KLT corner detector gives the best performance (for both matchers) followed by Harris, Kitchen-Rosenfeld and SUSAN detectors.*

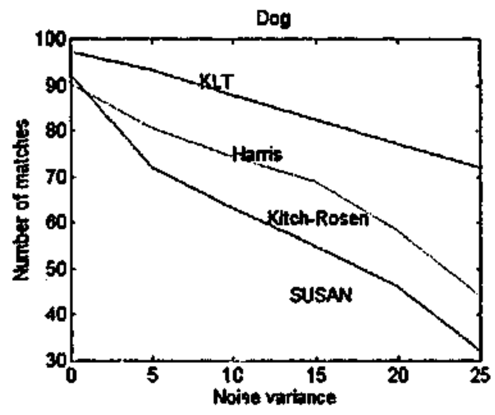
Dog Sequence Results (variation with noise)



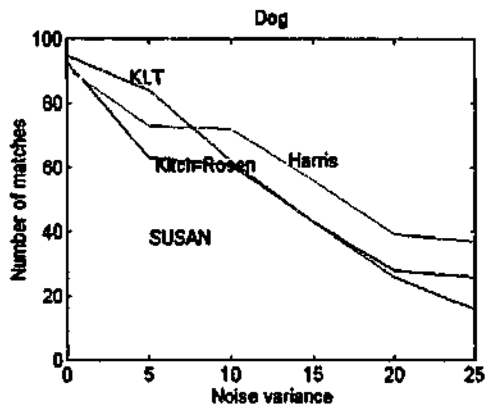
(a) Using GVM (Thresh=0.004)



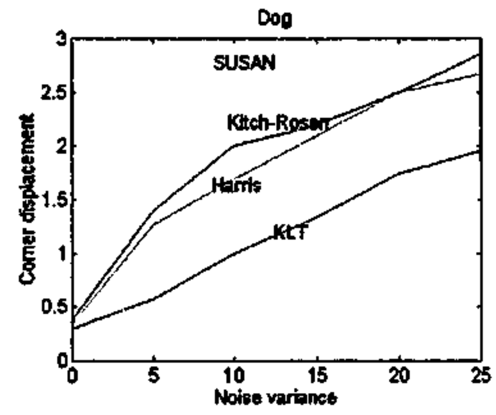
(b) Using PMCM (Thresh = 0.8)



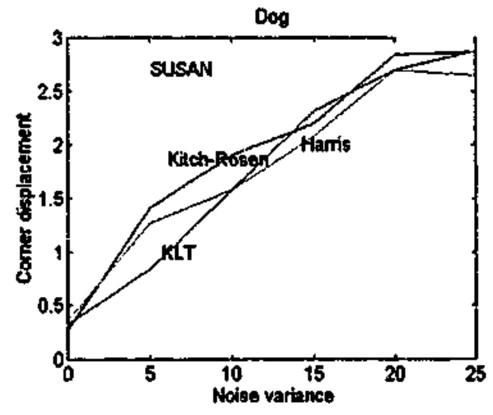
(c) Using GVM (Thresh=0.004)



(d) Using PMCM (Thresh = 0.8)



(e) Using GVM (Thresh=0.004)



(f) Using PMCM (Thresh = 0.8)

Figure 3.7: Performance of the corner detectors when applied to the static dog sequence at varied noise levels (noise variance ranging from 0 – 25). (a) The percentage stable corners using GVM matcher. (b) The percentage stable corners using PMCM matcher. (c) The number of first frame corner matches using GVM. (d) The number of first frame corner matches using PMCM. (e) Corner displacement using the GVM. (f) Corner displacement using the PMCM.

Building Sequence Results (effect of synthetic noise)

Corner Detector		KLT											
Patch Size		5 x 5											
Noise Variance		$\sigma = 0$		$\sigma = 5$		$\sigma = 10$		$\sigma = 15$		$\sigma = 20$		$\sigma = 25$	
Threshold		T1	T2	T1	T2	T1	T2	T1	T2	T1	T2	T1	T2
GVM	μ (CD)	0.8852	0.2782	2.6980	0.8931	2.9343	1.5726	2.9671	2.0177	2.9897	2.4095	2.9946	2.6025
GVM	σ^2 (CD)	0.0041	0.0009	0.0171	0.0030	0.0010	0.0035	0.0001	0.0056	0.0001	0.0047	0.0001	0.0040
GVM	μ (mat)	109.96	140.82	43.27	130.06	14.41	117.48	6.7586	103.31	3.4828	89.82	2.4483	75.65
GVM	σ^2 (mat)	29.68	4.48	38.82	9.37	14.10	17.90	10.87	31.45	7.42	33.65	1.8335	57.46
GVM	γ (%)	46.6	83.3	4.6	64.0	1.3	48.6	0.6	30.6	0.0	19.3	0.0	16.0
PMCM	μ (CD)	0.4556	0.8026	1.8354	2.6074	2.8133	2.9896	2.9642	3.0000	2.9934	3.0000	3.0000	3.0000
PMCM	σ^2 (CD)	0.0008	0.0031	0.0135	0.0137	0.0055	0.0002	0.0001	0.0000	0.0001	0.0000	0.0000	0.0000
PMCM	μ (mat)	130.24	112.79	79.86	38.82	29.10	3.9310	8.5862	0.6207	2.9310	0.2414	0.8966	0.0345
PMCM	σ^2 (mat)	8.25	20.92	33.70	18.55	9.88	1.58	5.2081	0.4423	1.2366	0.2521	0.4376	0.0333
PMCM	γ (%)	69.3	52.0	25.3	7.3	4.6	0.0	0.6	0.0	0.0	0.0	0.0	0.0

Table 3.7: Outdoor Building Sequence using KLT: Comparison of GVM (T1=0.0007, T2=0.009) and PMCM ,T1=0.7, T2=0.9) performances at varied noise levels for the KLT corner detector. The corner displacement (CD), number of matches (mat) and the percentage of features at the end of tracking γ) are performance measures for the comparison.

Corner Detector		Harris											
Patch Size		5 x 5											
Noise Variance		$\sigma = 0$		$\sigma = 5$		$\sigma = 10$		$\sigma = 15$		$\sigma = 20$		$\sigma = 25$	
Threshold		T1	T2	T1	T2	T1	T2	T1	T2	T1	T2	T1	T2
GVM	μ (CD)	0.5255	0.4416	2.4286	1.3513	2.9414	2.7288	3.0135	2.7603	2.9975	2.8751	2.9965	2.9490
GVM	σ^2 (CD)	0.0253	0.0122	0.0364	0.0104	0.0036	0.0177	0.1150	0.1451	0.0001	0.0027	0.0001	0.0007
GVM	μ (mat)	117.89	130.68	68.82	115.10	26.34	87.51	13.17	72.82	6.6897	52.68	4.2414	38.86
GVM	σ^2 (mat)	17.26	5.17	42.41	16.98	28.63	33.28	19.72	44.14	4.00	46.42	3.2176	58.73
GVM	γ (%)	62.2	76.8	7.2	50.3	0.6	22.5	0.6	9.2	0.0	3.9	0.0	2.6
PMCM	μ (CD)	0.4822	0.5094	1.4535	2.2332	2.5327	2.9641	2.9123	2.9978	2.9908	2.9995	2.9911	3.0000
PMCM	σ^2 (CD)	0.0180	0.0263	0.0127	0.0244	0.0215	0.0028	0.0023	0.0001	0.0002	0.0000	0.0005	0.0000
PMCM	μ (mat)	126.62	120.93	103.51	76.62	61.10	15.89	26.31	2.2759	11.89	1.06	5.3793	0.0000
PMCM	σ^2 (mat)	7.54	10.89	18.18	24.78	50.92	18.43	28.00	1.9929	12.23	0.61	7.8216	0.0000
PMCM	γ (%)	70.19	64.2	34.4	15.2	9.9	0.0	2.0	0.0	0.0	0.0	0.0	0.0

Table 3.8: Outdoor Building Sequence result using Harris: Comparison of GVM (T1=0.0007, T2=0.009) and PMCM (T1=0.7, T2=0.9) performances at varied noise levels for the Harris corner detector. The corner displacement (CD), number of matches (mat) and the percentage of features at the end of tracking γ) are performance measures for the comparison.

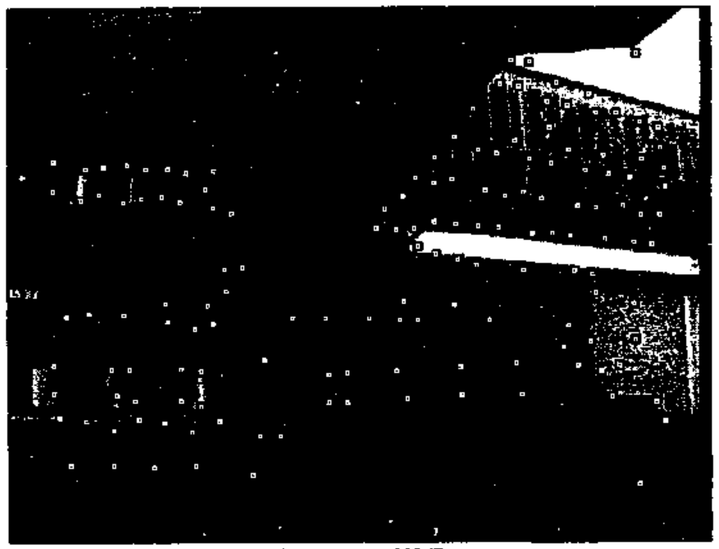
Building Sequence Results (effect of synthetic noise) Cont.

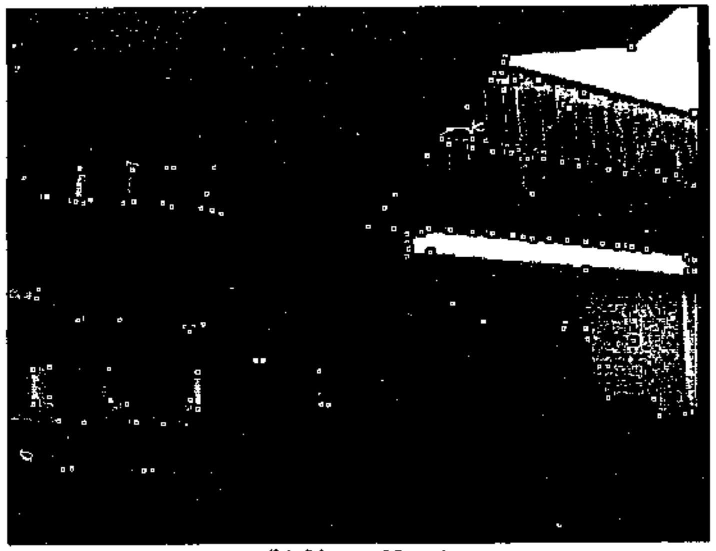
Corner Detector		*Kitchen-Rosenfeld*											
Patch Size		5 x 5											
Noise Variance		$\sigma = 0$		$\sigma = 5$		$\sigma = 10$		$\sigma = 15$		$\sigma = 20$		$\sigma = 25$	
Threshold		T1	T2	T1	T2	T1	T2	T1	T2	T1	T2	T1	T2
GVM	μ (CD)	0.8621	0.9031	2.6223	2.3226	2.8746	2.7730	2.9377	2.8623	2.9708	2.8804	2.9806	2.9610
GVM	σ^2(CD)	0.0824	0.0573	0.0101	0.0131	0.0016	0.0027	0.0020	0.0018	0.0005	0.0014	0.0000	0.0006
GVM	μ (mat)	107.89	115.89	53.24	75.41	25.06	52.00	16.34	38.55	10.96	37.44	6.82	22.03
GVM	σ^2 (mat)	44.09	32.50	55.97	41.34	18.47	43.24	18.43	57.14	12.37	32.17	5.52	22.51
GVM	γ (%)	46.9	53.6	8.7	17.44	3.3	8.0	1.3	4.7	0.6	4.7	0.6	2.0
PMCM	μ (CD)	0.8224	0.8465	2.3928	2.6107	2.8486	2.8915	2.9069	2.9820	2.9625	2.9955	2.9797	3.0000
PMCM	σ^2(CD)	0.0801	0.1010	0.0211	0.0127	0.0058	0.0015	0.0010	0.0016	0.0010	0.0001	0.0000	0.0000
PMCM	μ (mat)	114.20	109.34	66.93	50.03	36.13	19.13	21.03	9.5172	13.48	3.10	6.7586	0.4828
PMCM	σ^2 (mat)	32.78	45.53	52.13	45.6	34.32	7.36	18.72	4.1807	4.9394	1.47	9.1486	0.3187
PMCM	γ (%)	51.6	46.3	10.7	7.3	3.3	2.0	2.6	0.0	0.6	0.0	0.6	0.0

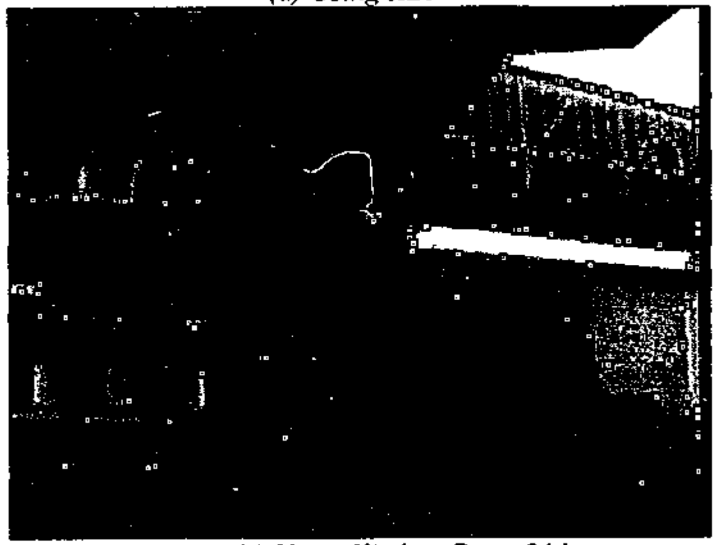
Table 3.9: Outdoor Building Sequence result using KitchenRosenfeld: Comparison of GVM (T1=0.0007, T2=0.009) and PMCM (T1=0.7, T2=0.9) performances at varied noise levels for the KitchenRosenfeld corner detector. The corner displacement (CD), number of matches (mat) and the percentage of features at the end of tracking (γ) are performance measures for the comparison.

Corner Detector		*SUSAN*											
Patch Size		5 x 5											
Noise Variance		$\sigma = 0$		$\sigma = 5$		$\sigma = 10$		$\sigma = 15$		$\sigma = 20$		$\sigma = 25$	
Threshold		T1	T2	T1	T2	T1	T2	T1	T2	T1	T2	T1	T2
GVM	μ (CD)	1.4536	1.1980	2.9560	2.4322	2.9948	2.8880	2.9940	2.9871	2.9934	2.9910	2.9993	3.0067
GVM	σ^2(CD)	0.1444	0.0572	0.0015	0.0147	0.0002	0.0082	0.0001	0.0024	0.0001	0.0006	0.0000	0.0006
GVM	μ (mat)	87.41	118.31	36.93	94.96	14.06	66.17	8.0690	48.96	3.8276	37.72	5.44	36.96
GVM	σ^2 (mat)	27.82	18.28	23.99	30.44	15.23	58.69	25.65	59.55	4.9013	47.71	6.31	53.89
GVM	γ (%)	25.1	49.0	1.3	13.9	0.0	3.9	0.0	0.0	0.0	0.6	0.0	0.0
PMCM	μ (CD)	1.2676	1.3243	2.7669	2.9494	2.9684	2.9875	2.9937	2.9936	2.9926	3.0000	2.9985	3.0000
PMCM	σ^2(CD)	0.0768	0.1020	0.0175	0.0067	0.0012	0.0002	0.0002	0.0000	0.0001	0.0000	0.0000	0.0000
PMCM	μ (mat)	103.41	91.51	69.86	40.93	35.86	11.34	19.44	3.1034	10.55	0.7931	9.6552	0.4483
PMCM	σ^2 (mat)	23.55	34.11	36.73	28.96	37.22	11.19	30.17	5.95	16.38	1.1986	8.9845	0.5922
PMCM	γ (%)	38.41	31.78	3.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 3.10: Outdoor Building Sequence result using SUSAN: Comparison of GVM (T1=0.0007, T2=0.009) and PMCM (T1=0.7, T2=0.9) performances at varied noise levels for the SUSAN corner detector. The corner displacement (CD), number of matches (mat) and the percentage of features at the end of tracking γ are performance measures for the comparison.


(a) Using KLT


(b) Using Harris


(c) Using Kitchen-Rosenfeld

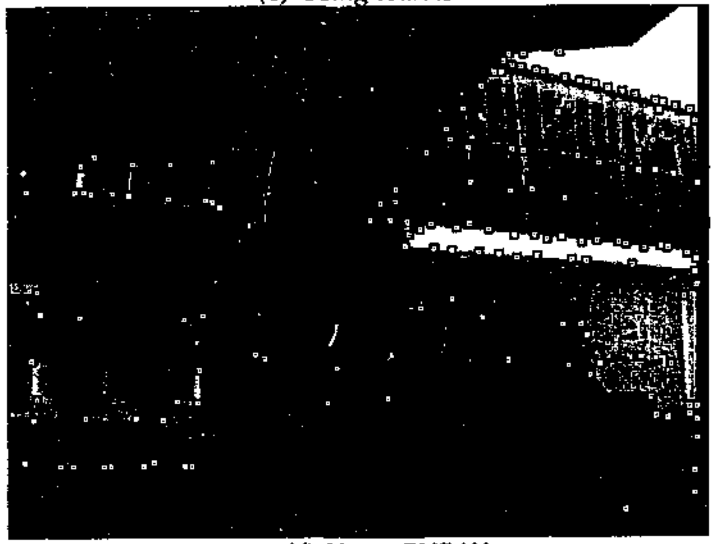
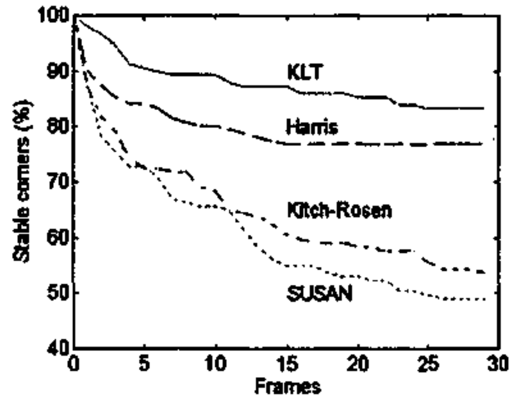
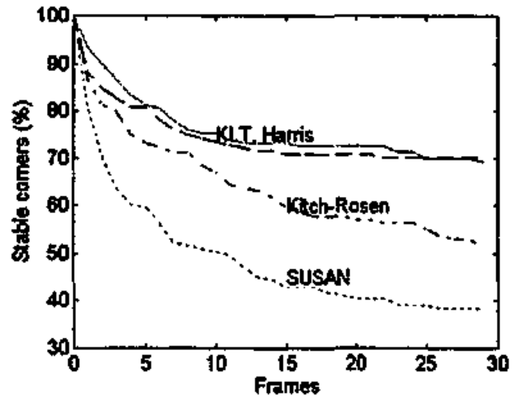

(d) Using SUSAN

Figure 3.8: The best 150 corners extracted from the outdoor static building sequence. (a) Using KLT corner detector. (b) Using Harris corner detector. (c) Using Kitchen-Rosenfeld corner detector. (d) Using SUSAN corner detector

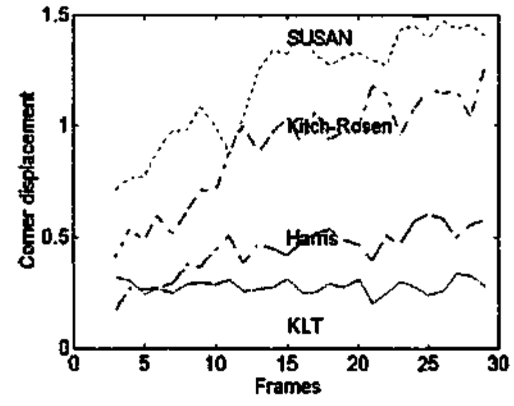
Building Sequence Results (Performance vs. Number of Frames)



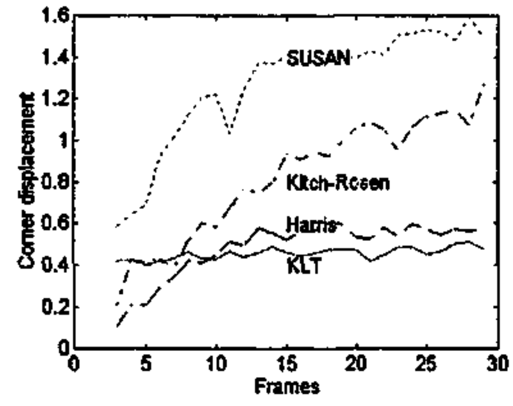
(a) Using GVM (Thresh=0.009)



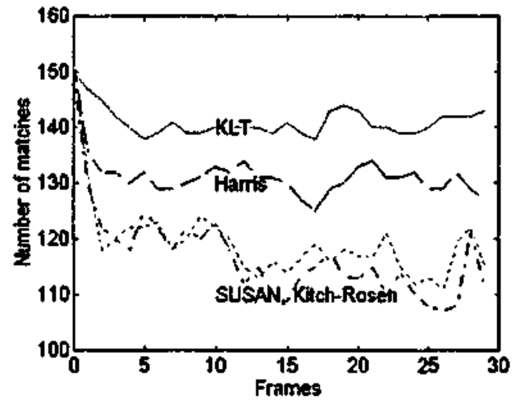
(b) Using PMCM (Thresh = 0.7)



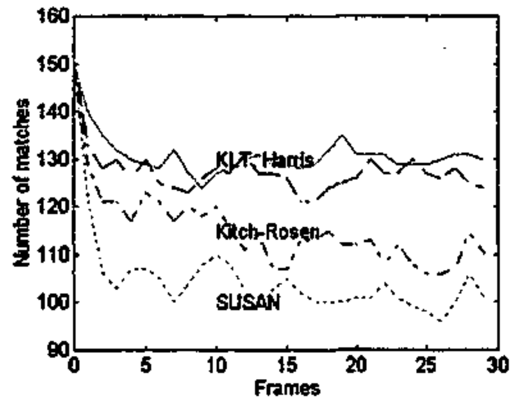
© Using GVM (Thresh=0.009)



(d) Using PMCM (Thresh = 0.7)



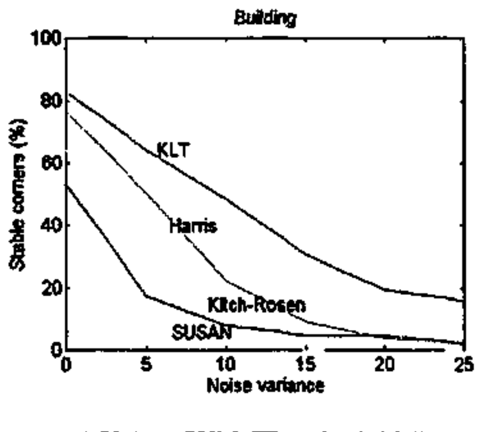
(e) Using GVM (Thresh=0.009)



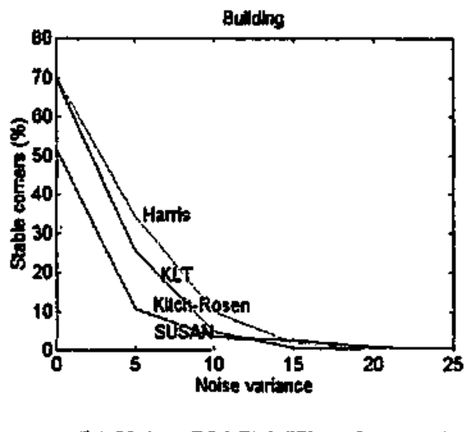
(f) Using PMCM (Thresh = 0.7)

*Figure 3.9: Corner detector performance-test for the '**static building**' sequence (the best 150 corners as seen by each detector are extracted from each frame). (a) The KLT, Harris, Kitchen-Rosenfeld and SUSAN corner detectors are assessed for stable corners (percentage) using GVM matcher. (b) Stable corners compared using the PMCM matcher. (c) Corner displacement test using GVM matcher. (d) Corner displacement test using PMCM matcher. (e) Number of successful first frame corner matches using GVM. (f) Number of successful first frame corner matches using PMCM. In all 3 tests the KLT corner detector gives the best performance (for both matchers) followed by Harris, Kitchen-Rosenfeld and SUSAN detectors.*

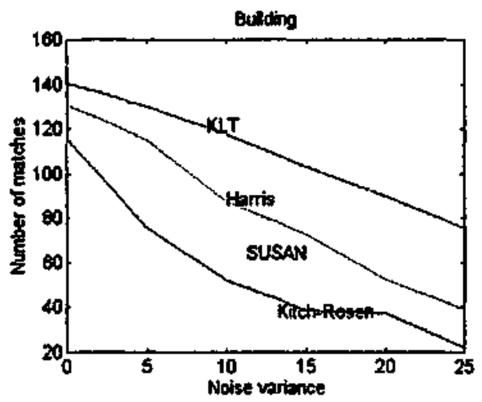
Building Sequence Results (variation with noise)



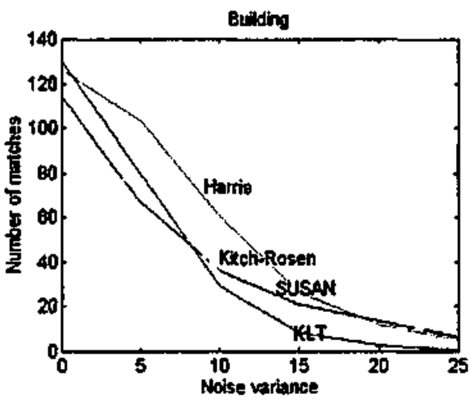
(a) Using GVM (Thresh=0.004)



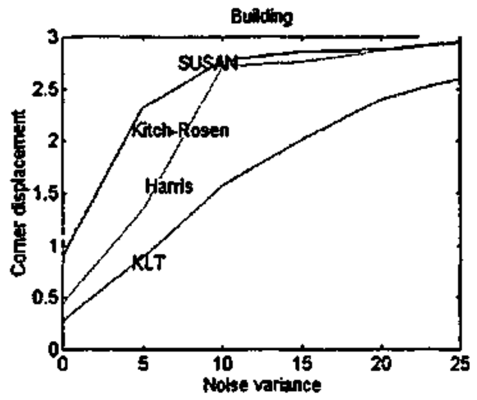
(b) Using PMCM (Thresh = 0.8)



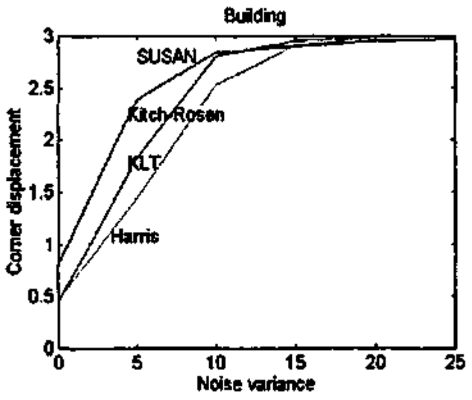
(c) Using GVM (Thresh=0.004)



(d) Using PMCM (Thresh = 0.8)



(e) Using GVM (Thresh=0.004)



(f) Using PMCM (Thresh = 0.8)

Figure 3.10: Performance of the corner detectors when applied to the ***static building*** *sequence at varied noise levels (noise variance ranging from 0 – 25). (a) The percentage stable corners using GVM matcher. (b) The percentage stable corners using PMCM matcher. (c) The number of first frame corner matches using GVM. (d) The number of first frame corner matches using PMCM. (e) Corner displacement using the GVM. (f) Corner displacement using the PMCM.*

Lab Sequence Results

Corner Detector		*KLT*		*Harris*		*Kitch-Rosen*		*SUSAN*	
Patch Size		5x5		5x5		5x5		5x5	
Threshold (*GVM/PMCM*)		**T=0.004**	**T=0.8**	**T=0.004**	**T=0.8**	**T=0.004**	**T=0.8**	**T=0.004**	**T=0.8**
GVM	μ (CD)	0.3386	-	0.2493	-	0.5047	-	0.7039	-
GVM	σ^2 (CD)	0.0005	-	0.0014	-	0.0053	-	0.0034	-
GVM	μ (matches)	184.89	-	191.86	-	179.13	-	176.10	-
GVM	σ^2 (matches)	4.36	-	3.56	-	11.42	-	15.47	-
GVM	γ (%)	82.5	-	87.0	-	78.9	-	67.0	-
PMCM	μ (CD)	-	0.4079	-	0.2071	-	0.4416	-	0.9449
PMCM	σ^2 (CD)	-	0.0003	-	0.0003	-	0.0023	-	0.0030
PMCM	μ (matches)	-	178.27	-	191.51	-	177.27	-	157.93
PMCM	σ^2 (matches)	-	5.71	-	4.87	-	11.44	-	26.13
PMCM	γ (%)	-	78.0	-	85.5	-	76.8	-	51.5

Table 3.11: Performance of the 4 corner detectors for the ***static lab sequence*** *(only a single threshold value is used for the 2 matchers employed).*

Computer Sequence Results

Corner Detector		*KLT*		*Harris*		*Kitch-Rosen*		*SUSAN*	
Patch Size		5x5		5x5		5x5		5x5	
Threshold (*GVM/PMCM*)		**T=0.004**	**T=0.8**	**T=0.004**	**T=0.8**	**T=0.004**	**T=0.8**	**T=0.004**	**T=0.8**
GVM	μ (CD)	0.6212	-	0.3720	-	0.8542	-	1.5328	-
GVM	σ^2 (CD)	0.0009	-	0.0015	-	0.0067	-	0.0064	-
GVM	μ (matches)	223.44	-	237.93	-	210.10	-	196.86	-
GVM	σ^2 (matches)	21.35	-	14.96	-	15.54	-	30.67	-
GVM	γ (%)	73.6	-	84.2	-	64.9	-	43.7	-
PMCM	μ (CD)	-	0.9669	-	0.3915	-	0.7040	-	1.8518
PMCM	σ^2 (CD)	-	0.0021	-	0.0002	-	0.0014	-	0.0204
PMCM	μ (matches)	-	192.31	-	229.20	-	205.72	-	157.41
PMCM	σ^2 (matches)	-	67.59	-	12.71	-	21.23	-	41.82
PMCM	γ (%)	-	53.6	-	77.0	-	60.5	-	27.7

Table 3.12: Performance of the 4 corner detectors for the ***static computer sequence*** *(only a single threshold value is used for the 2 matchers employed).*

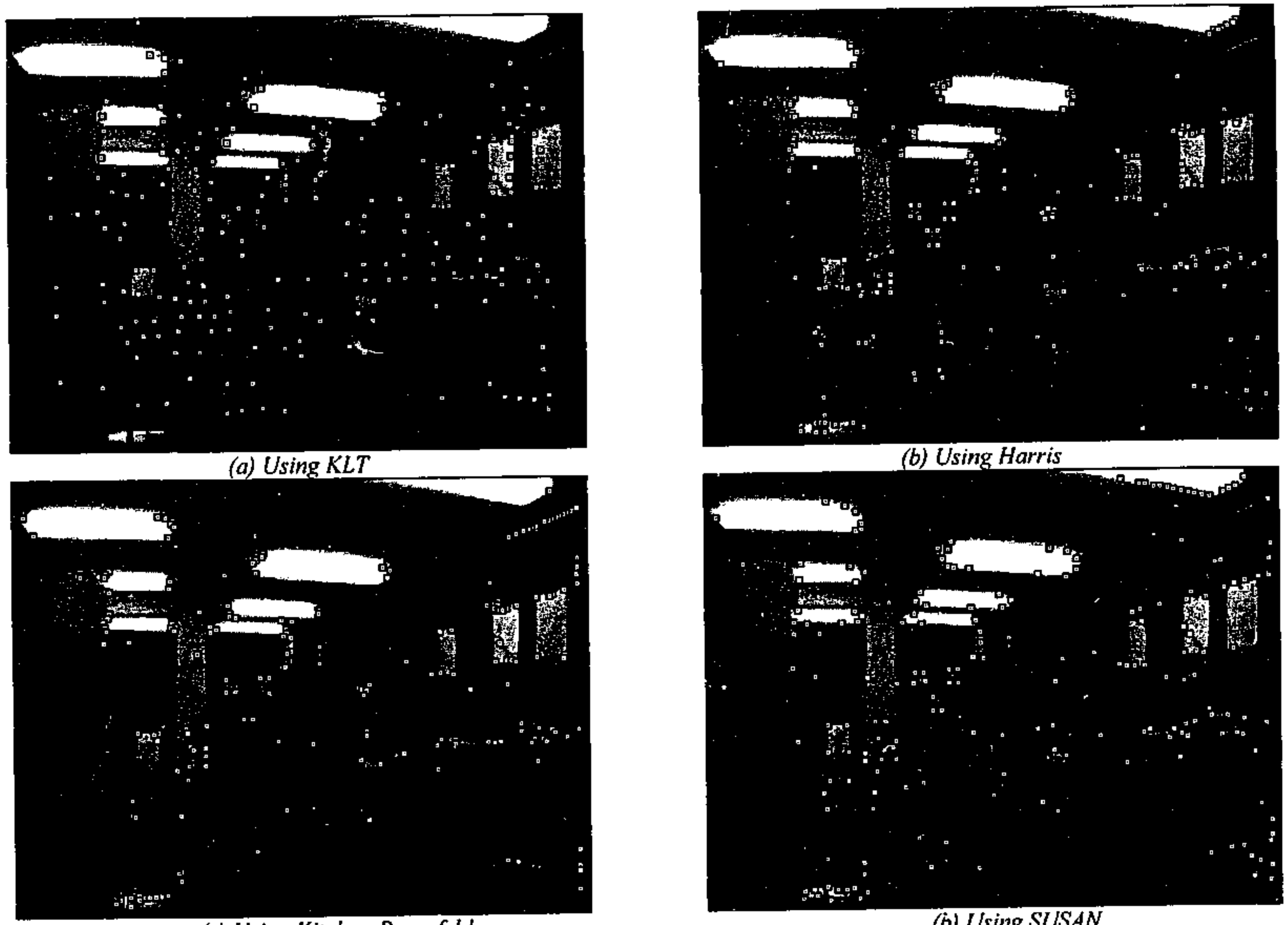
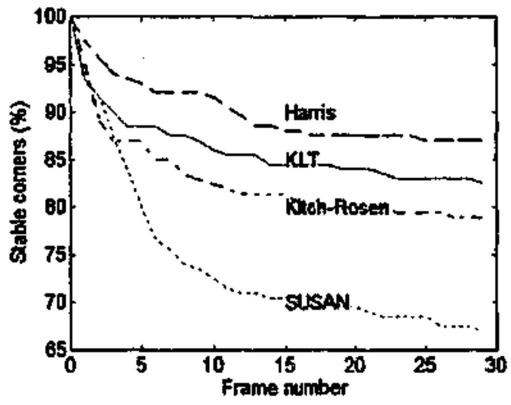
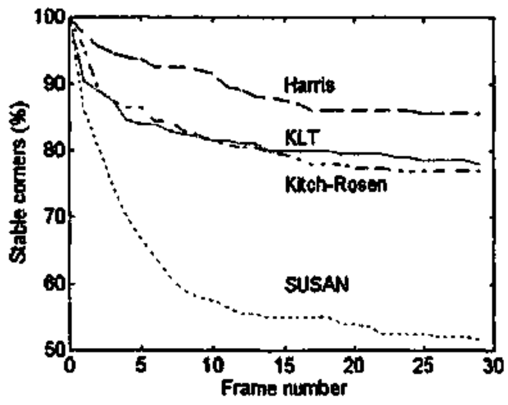


Figure 3.11: The best 200 corners extracted from the static lab sequence. (a) Using KLT corner detector. (b) Using Harris corner detector. (c) Using Kitchen-Rosenfeld corner detector. (d) Using SUSAN corner detector.

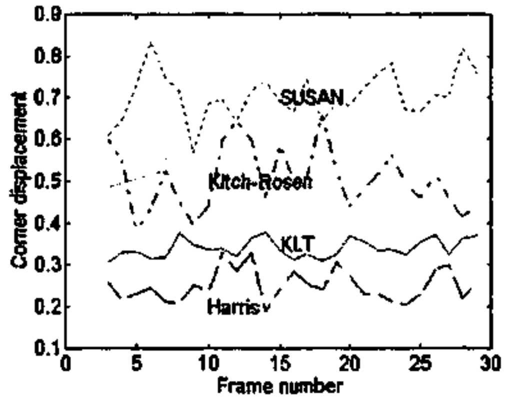
Lab Sequence Results (Performance vs. Number of Frames)



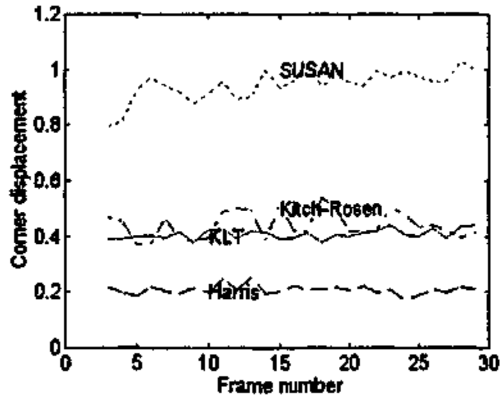
(a) Using GVM (Thresh=0.004)



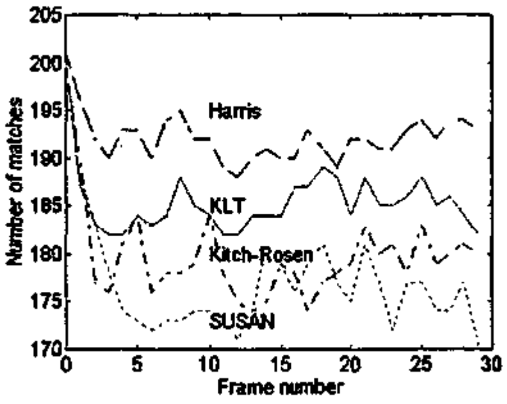
(b) Using PMCM (Thresh = 0.8)



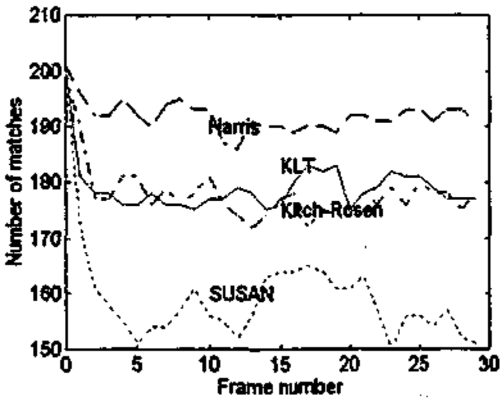
(c) Using GVM (Thresh=0.004)



(d) Using PMCM (Thresh = 0.8)

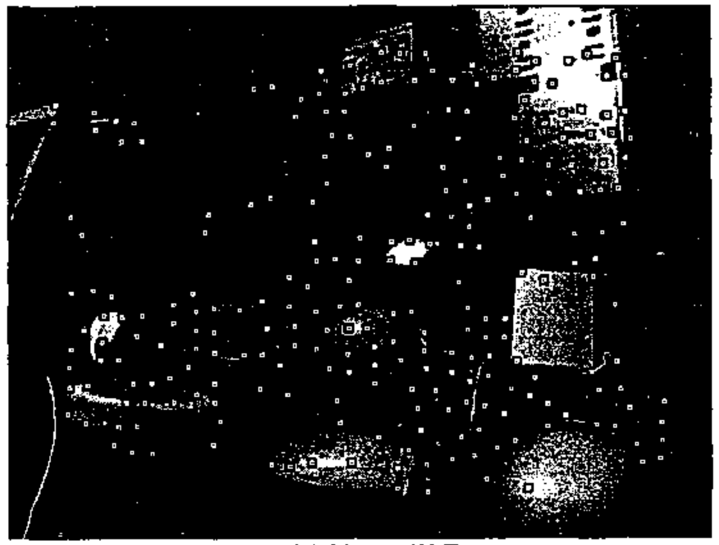


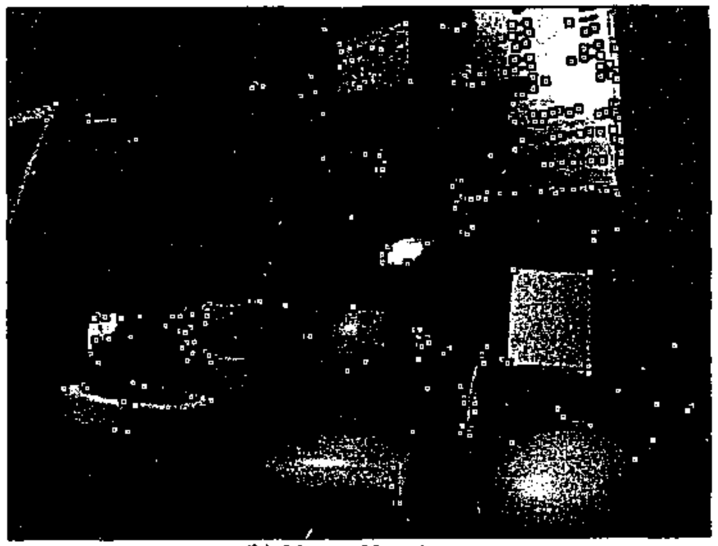
(e) Using GVM (Thresh=0.004)

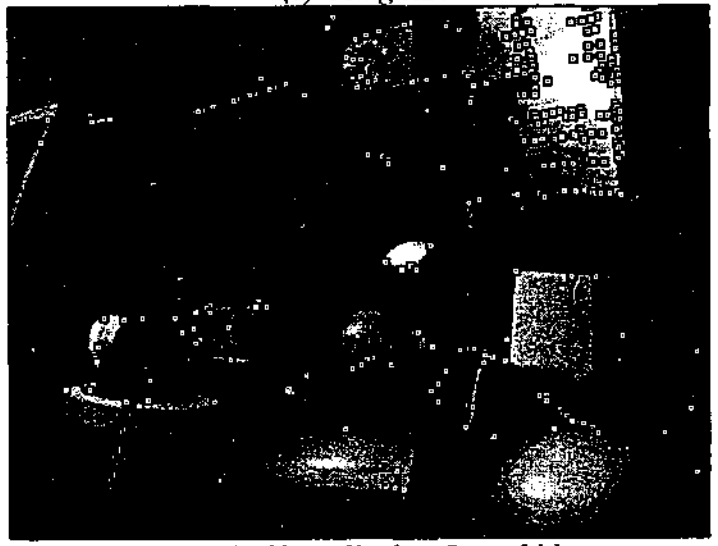


(f) Using PMCM (Thresh = 0.8)

Figure 3.12: Corner detector performance-test for the ***'static lab'*** *sequence (the best 200 corners as seen by each detector are extracted from each frame). (a) The KLT, Harris, Kitchen-Rosenfeld and SUSAN corner detectors are assessed for stable corners (percentage) using GVM matcher. (b) Stable corners compared using the PMCM matcher. (c) Corner displacement test using GVM matcher. (d) Corner displacement test using PMCM matcher. (e) Number of successful first frame corner matches using GVM. (f) Number of successful first frame corner matches using PMCM. In all 3 tests the Harris corner detector gives the best performance (for both matchers) followed by KLT, Kitchen-Rosenfeld and SUSAN detectors.*


(a) Using KLT


(b) Using Harris


(c) Using Kitchen-Rosenfeld

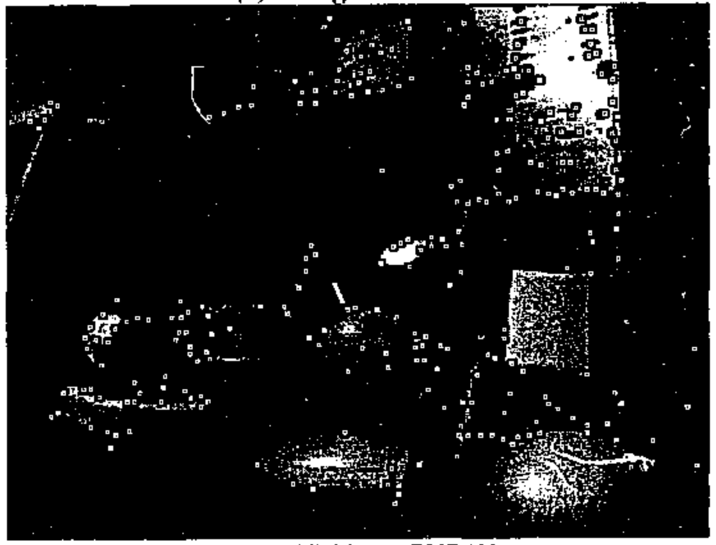
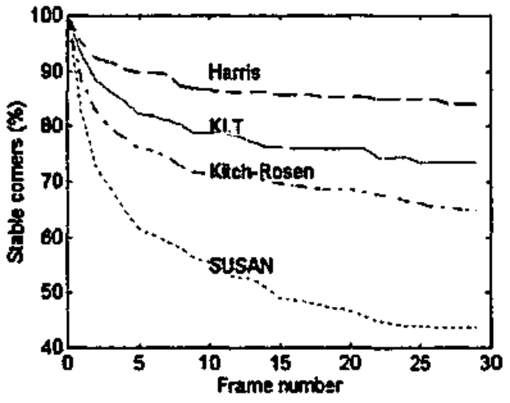
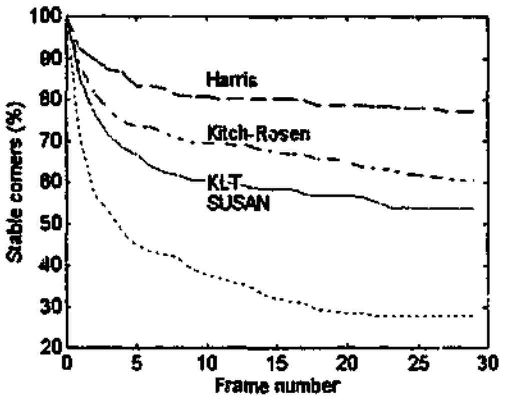

(d) Using SUSAN

Figure 3.13: The best 250 corners extracted from the static computer sequence. (a) Using KLT corner detector (b) Using Harris corner detector. (c) Using Kitchen-Rosenfeld corner detector. (d) Using SUSAN corner detector.

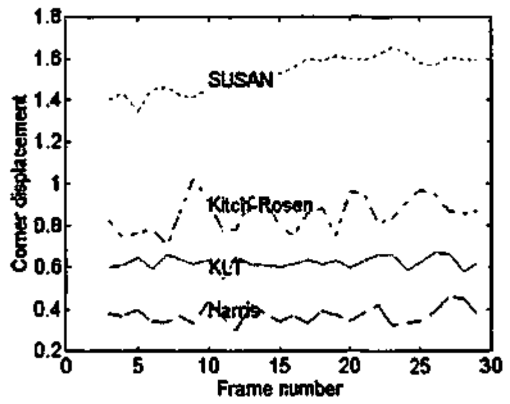
Computer Sequence Results (Performance vs. Number of Frames)



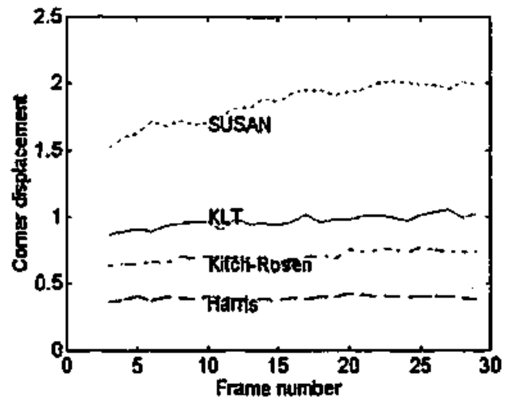
(a) Using GVM (Thresh=0.004)



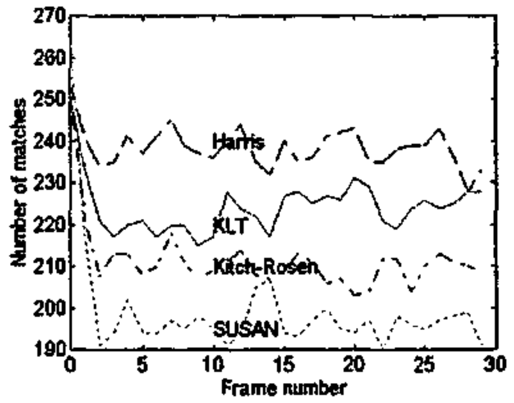
(b) Using PMCM (Thresh = 0.8)



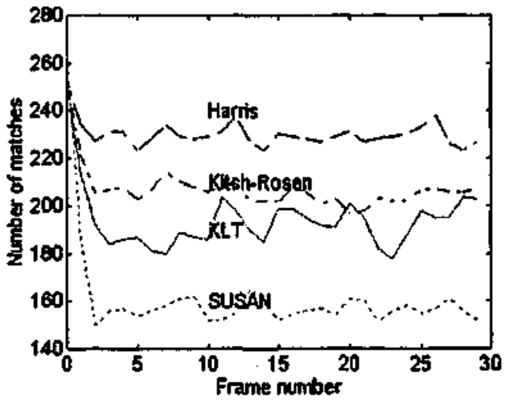
(c) Using GVM (Thresh=0.004)



(d) Using PMCM (Thresh = 0.8)



(e) Using GVM (Thresh=0.004)



(f) Using PMCM (Thresh = 0.8)

Figure 3.14: Corner detector performance-test for the 'static computer' sequence (the best 250 corners as seen by each detector are extracted from each frame). (a) The KLT, Harris, Kitchen-Rosenfeld and SUSAN corner detectors are assessed for stable corners (percentage) using GVM matcher. (b) Stable corners compared using the PMCM matcher. (c) Corner displacement test using GVM matcher. (d) Corner displacement test using PMCM matcher. (e) Number of successful first frame corner matches using GVM. (f) Number of successful first frame corner matches using PMCM. In all 3 tests the Harris corner detector gives the best performance (for both matchers) followed by KLT, Kitchen-Rosenfeld and SUSAN detectors.

3.7 Discussion

The fo!lowing sub-sections give detail results on the empirical evaluations carried on the four image sequences considered.

3.7.1 Test Results for Indoor Dog Sequence

A 30 frame static dog sequence was considered. The 100 best corners as seen by each of the corner detectors were extracted. Fig. (3.5) shows the qualitative results obtained by applying the 4 corner detectors with the 100 corners superimposed on the first frame.

3.7.1.1 General Performance

From the results reported in Fig. (3.6) and Tables 3.1, 3.3-3.6, it can be seen that the KLT detector provided the largest number of stable corners using the GVM matcher with a 0.009 threshold (93% stable corners) and using the PMCM matcher with a 0.7 threshold (88% stable corners). The number of first frame corner matches also indicate that KLT provided the best result (Fig. (3.6e,f), Table (3.3)). About 97% of matches are reported using GVM matcher and about 95% matches are reported employing the PMCM matcher. The mean corner displacement result indicates that the KLT provided about 0.3 pixel displacement using both matchers (Fig. (3.6c,d)). Both matchers were also tested with a more stringent threshold values (0.0007 for GVM and 0.9 for PMCM), and for complete details see Tables 3.3-3.6. The 2 matchers were tested with 2 different sizes of patches (5x5 and 7x7), but no significant differences were observed (Table 3.1).

The Harris corner detector also provided equally good results. Fig. (3.6) indicates that 86% stable corners are detected using GVM matcher (with 0.009 threshold) and about 87% stable corners are reported using the PMCM matcher (with 0.7 threshold). The first frame corner matches also shows that about 90% matches are found using GVM and PMCM matchers for the same threshold values. The mean corner displacement result suggest that the Harris detector is as good as the KLT detector in providing around 0.3 pixel displacement for both matchers, which indicate good corner localization property. Tests carried out using tighter threshold values (GVM wit 0.0007 and PMCM with 0.9) are reported in Table 3.4, which shows that Harris detector performed better than the KLT detector. As before, the difference in patch sizes did not result in significant difference for each of the tests carried out (Table 3.1).

Kitchen-Rosenfeld detector resulted with about 82% stable corners using GVM with a threshold of 0.009 and about 84% using PMCM with a 0.7 threshold. The number of first frame matches and mean corner displacement are as good as for the Harris detector (Fig. (3.6)). Again with tighter thresholds, the performance results are refined as one might expect (see Table (3.1)).

The SUSAN detector is the least impressive of the 4 detectors (Fig. 3.6). Only around 62% stable corners are reported using GVM (with 0.009 threshold) and about 50% using PMCM matcher (with 0.7 threshold). The mean first frame corner matches also indicate that on average only about 80% corners are matched throughout the sequence using GVM, and about 75% using PMCM matcher. The mean corner displacement result shows that about 1.2 pixel displacement is observed using GVM, and about 1.35 pixels using PMCM. This suggests poor localization of corners for the sequence considered. Tighter matching constraints resulted in 0% stable corners detected (for both matchers) and a poor mean corner displacement (see Table (3.6)). A value of 3 pixels is assigned (indicating poor localization property) for corner displacement if there was no valid match of a corner is reported.

3.7.1.2 Performance under Noise

We observed the results of the 4 corner detection algorithms on the same sequence, after adding Gaussian noise to each frame (apart from frame 1) at varied noise levels (with noise variance ranging from 0 - 25) prior to applying the detectors. The results are reported in Tables (3.3)-(3.6) and Fig. (3.7). As expected the quality of result decreases rapidly with added noise. The stable corners using KLT dropped from 90% (at $\sigma^2 = 0$) to about 40% (at $\sigma^2 = 25$) using GVM (at threshold 0.004), while Harris performance dropped from around 83% to 15%, Kitchen-Rosenfeld dropped from about 80% to 8%, and SUSAN dropped from 60% to 0 %. Added noise has more effect using the PMCM matcher (0.8 threshold), because it uses an image patch correlation technique to match corners in consecutive frames, thus the result can be somewhat inaccurate (See Fig. (3.7b,d,f)). The number of first frame corner matches also follows similar trend as the stable corners. The mean corner displacement also deteriorates rapidly with noise. KLT provided around 0.3 pixels at $\sigma^2 = 0$, which increased to nearly 2 pixels at $\sigma^2 =25$, while Harris result jumped from around 0.3 pixel to 2.5 pixels for the same range of noise variation. Kitchen-Rosenfeld detector reported similar corner displacement values to Harris method, while using SUSAN, the displacement increased from 1.2 pixels to nearly 3 pixels. Similar trends are also reported using the PMCM matcher (Fig (3.7)). The overall experiments suggest that KLT and Harris corner detectors still outperform the Kitchen-Rosenfeld and SUSAN detectors under noise.

3.7.2 Test Results for Outdoor Building Sequence

A 30 frame 'static outdoor building' sequence with only natural lighting was considered. The 150 best corners as seen by each of the corner detectors were extracted. Fig. (3.8) shows the qualitative results obtained by applying the 4 corner detectors.

3.7.2.1 General Performance

From results reported in Fig. (3.9) and Tables (3.7-3.10), it is clear, that in general the number of percentage stable corners tracked by all 4 detectors are less than for the dog sequence. The KLT provided the best result using GVM matcher (with 0.009 threshold) with about 85% stable corners, followed by Harris detector with around 80%, Kitchen-Rosenfeld with 55%, and SUSAN with only 50% stable corners. Using PMCM matcher (with 0.7 threshold), KLT and Harris provided around 70% stable corners, followed by Kitchen Rosenfeld (50%) and SUSAN (40%) detectors. The average number of first frame corner matches using GVM is significantly higher for KLT (with around 140 corner matches), followed by Harris (with 130), then Kitchen-Rosenfeld and SUSAN with each around 117 matches. Using PMCM matcher, the KLT and Harris provide around equal number of match (130), followed by Kitchen-Rosenfeld (110) and SUSAN (100) detectors. The corner displacement test again reveals that KLT provides the best localization property with around 0.25 pixel displacement, which is followed by Harris (0.4 pixels), Kitchen-Rosenfeld (1 pixel), and SUSAN (1.5 pixels) detectors. Corner displacement test using PMCM (with 0.7 threshold) revealed that, KLT and Harris provide around 0.4 pixel displacement, followed by Kitchen-Rosenfeld (0.8 pixel), and SUSAN (1.4 pixels) detectors.

3.7.2.2 Performance under Noise

All 4 corner detectors were subject to extract corners from noisy frames of the building sequence (Gaussian noise is added at different variances ranging from 0 – 25). The results observed are tabulated in Tables (3.7-3.10) and Figure 3.10. KLT still provided the best result for the most number of stable corners using GVM (at 0.004 threshold), providing around 20% stable corners at noise variance $\sigma^2 = 25$, while the other 3 detectors provided only around 5% at $\sigma^2 = 25$. With PMCM (at 0.8 threshold), all four corner detectors resulted with 0% stable corners at $\sigma^2 = 25$. This is expected, as outdoor sequences already have image plane noise, and by adding extra noise, causes the PMCM correlation matcher to result in very low match coefficient. The number of first frame corner matches using GVM resulted with KLT having around 80 matches at noise variance at $\sigma^2 = 25$, followed by Harris (40 matches), Kitchen-Rosenfeld (40 matches), and SUSAN (with 0 matches). Using PMCM (with 0.8 threshold) provided around 10 matches for all 4 detectors at $\sigma^2 = 25$. The mean corner displacement for KLT at $\sigma^2 = 25$ is around 2.1 pixels, which is followed by the other 3 detectors providing around 3 pixels displacement. The same experiment using PMCM results with all 4 detectors providing 3 pixels displacement, indicating poor quality corner localization under considerable noise.

3.7.3 Test Results for the Static Lab Sequence

A 30 frame lab sequence with direct light interference was considered. The 200 best corners as seen by each of the detectors are extracted, and are qualitatively displayed in Fig. (3.11).

3.7.3.1 General Performance

The results reported in Table 3.11 and Figure (3.12) indicate that Harris detector provided the best result for this sequence. Harris provide nearly 90% stable corners while KLT provided about 82% stable corners. Kitchen-Rosenfeld and SUSAN detectors resulted with 80% and 67% stable corners respectively using the GVM matcher (with 0.004 threshold). The number of first-frame corner matches also indicate that Harris gives the highest number of matches than the other detectors (a mean of 192 for Harris, 185 for KLT, 177 for Kitchen-Rosenfeld and 173 for SUSAN) using the GVM matcher. The mean corner displacement result also shows that Harris provides the best result with around 0.25 pixel displacement, followed by KLT with 0.3 pixel, Kitchen-Rosenfeld with 0.5 pixel, and SUSAN with 0.7 pixel displacement using GVM matcher. The trends in observations are consistent when employing the PMCM matcher with a 0.8 threshold.

3.7.4 Test Results for the Static Computer Sequence

A 30 frame computer sequence with light reflections was considered. This sequence had plenty of curved objects, with less easily definable corners, thus presenting a challenge for each of the corner detectors. The 250 best corners extracted using each corner detector are displayed in Fig. (3.13).

3.7.4.1 General Performance

Figure (3.14) and Table 3.12 shows that Harris detector again provided the best result for the most number of stable corners (85%), followed by KLT (75%), which is followed by Kitchen-Rosenfeld (65%) and SUSAN (45%) detectors using the GVM matcher (0.004 threshold). The observations are similar for PMCM (0.8 threshold) except Kitchen-Rosenfeld performed better than KLT detector (see Fig. (3.17b,d,f)). The number of first frame matches also show that Harris with around 240 matches on average outperformed the KLT (220 matches), the Kitchen-Rosenfeld (210) and the SUSAN (195) detectors for both matchers. The mean corner displacement result also suggest that Harris provides the lowest displacement with 0.4 pixel, followed by KLT with 0.6 pixel, followed by Kitchen-Rosenfeld (0.9 pixel) and SUSAN (1.4 pixel) detectors for the GVM matcher. The results are consistent when using the PMCM matcher.

3.7.5 Overall Observation of Results

The overall observation of the results suggest that the KLT and Harris corner detectors are more suitable for tracking features in long sequences, while Kitchen-Rosenfeld and SUSAN are less reliable for long term corner tracking.

Further tests carried out (not reported in this chapter due to space limitations) also suggest that for tracking large number of corner points, Harris provides slightly better result than KLT, while for tracking small number of corners, KLT provides better quality results. It is also observed that for sequences with varying light sources, Harris detector provides better quality results than KLT. The qualitative results also show (Figs. 3.5, 3.8, 3.11, 3.13) that KLT picks the best *N* corners from all parts of the image frame (which is highly desirable for multiple object tracking) while the other detectors tend to pick corners from objects where there is significant difference in contrast. Kitchen-Rosenfeld and SUSAN detectors also tend to pick several corners from edges (despite 'edge suppression' applied to the detectors), which is undesirable for point feature tracking applications.

The empirical evaluations also shed some insight into the matcher's ability to correctly associate corners. The overall results suggest that for an indoor sequence, GVM or PMCM give equally good results, but for an outdoor sequence the GVM provides better quality result. This is expected, because the PMCM matcher compares a patch of image information surrounding the corner in two frames. With image plane noise (generally the case for outdoor images) one would expect a reduced correlation coefficient, which in turn leads to less reliable results. The two patch sizes examined did not make a significant difference, which indicates that a 5 x 5 image patch size is adequate for most applications. Setting matcher threshold is more of a design issue. It is important that threshold chosen should impose restraints for dis-allowing false corners being accepted as correct match.

These results pose the question; which is best, a matcher that produces a few good matches per frame ?, or a slightly less accurate matcher that produces more matches, but also produces a few bad matches ?. It is the opinion of the author that a matcher that produces a high number of matches (also with high percentage of stable corners) is preferable, even if the data generated contained bad matches. A good tracking algorithm will be able to discard bad matches over a period of time.

From the results reported in this chapter, it is reasonably clear that the KLT and Harris detectors are appropriate corner extractors to track point features in long image sequences, with GVM as the preferred matcher particularly for outdoor sequences, and PMCM for indoor sequences (easier to implement than GVM). Further feature tracking examples with considerable motion in image sequences are discussed in the next chapter in detail. In such cases we employ tracking algorithms to estimate the likely position of features in subsequent frames.

3.8 Conclusion

The results are interesting because unlike the *Harris* and *Kitchen-Rosenfeld* detector, the *Smith* detector uses no spatial derivatives. Spatial derivatives are normally associated with poor performance in the presence of noise since they magnify its effect, and hence it would be expected that the *Smith* detector would perform better than the *Harris* and the *Kitchen-Rosenfeld* detectors in the presence of noise (assuming that both detectors perform equally as well when there is no noise present). This is clearly not the case. A possible explanation for these results is that the *Harris* detector has a built-in smoothing function as part of its formulation. The products of the intensity gradients used to calculate the cornerness *C* are first Gaussian smoothed over a 3x3 pixel image patch. It is this smoothing that makes the *Harris* detector more robust to noise than the *Smith* detector even though it uses noise sensitive first derivatives.

Because of its poor performance in the presence of image-plane noise and hence its very temporally unstable corners, the *Smith* corner detector is not an appropriate feature detector for tracking in long sequences. It performed well in the ASSET project because there, segmentation was performed using 2 frame matching, which was good enough to produce good number of matches in consecutive frames.

The *Kitchen-Rosenfeld* method does not use a built-in smoothing function and also has second order derivatives, and as a result its' performance is poorer than the *Harris* method (*K-R* was used only as a bench-mark). The *KLT* detector on the other hand is aided by its tracking framework (the tracking process cannot be easily decoupled from the corner detection process). The *KLT* tracker and corner detectors work together to provide high quality corners as indicated by the results. The overall empirical results revealed that the *KLT* and *Harris* detectors provided the best quality corners (qualitatively and quantitatively). The corners extracted by the *Kitchen-Rosenfeld* and the *Smith (SUSAN)* detectors are less desirable for point feature tracking in long image sequences.

Chapter 4

Point Feature Tracking with Automatic Motion Model Switching

Abstract

This chapter provides a novel technique of efficiently and reliably tracking features in a sequence of images. The method we provide for tracking features is based on theBayesian Multiple Hypothesis Tracking (MHT) technique coupled [58] with a Multiple Model Filtering algorithm. We show the results of our work comparing it with some of the existing single model based trackers using a variety of video sequences. Initially we demonstrate the ability of the MHT-MMF tracker developed (Fixed form of the filter), and later in the chapter we extend the MMF based tracker to the Interacting Multiple Model (IMM) tracker [5, 28] and show the superiority of the latter in handling motion switching of features efficiently. The primary purpose of this chapter is to show how the IMM algorithm combined with an extension of the classical MHT framework can be used in a visual tracking scenario. The study shows that the method proposed can distinguish between different motions depicted in an image sequence with good feature tracking results.

4.1 Introduction

In this chapter we propose a feature tracking algorithm based on the combination of Multiple Hypothesis Tracking (MHT) and Multiple Model filtering technique. The combination of these two methods provides an attractive feature tracker that has the capability of switching motion models according to the object's motion.

The surveillance tracking community has studied target tracking techniques for a number of years, mainly in the context of finding efficient methods to track missiles, aircraft etc. and tracking targets of unknown motion. Their work has been used for a variety of applications [54-56, 5, 6, 41-43, 152, 125,126]. In the recent years there has been an interest in using surveillance tracking techniques for visual tracking applications such as tracking features in a video sequence. One such proposal is outlined in [58] by Cox *et al.* Cox *et al* use a modified version of the MHT tracker, based on the assumption of a single motion model. We extend the modified MHT tracker to track features that move with multiple motions. The proposed "multiple model based" MHT tracker is shown to outperform visual trackers based on a single motion model as demonstrated in this chapter.

Multiple hypothesis based tracking methods have been shown to provide reliable results for a variety of tracking applications [55, 58, 180, 184, 185]. An important reason for considering the MHT technique for point feature tracking is because the MHT technique is a statistical data association algorithms that integrates all the capabilities such as track initiation, track termination, track continuation, explicit modelling of spurious measurements, and explicit modelling of uniqueness constraints. Unfortunately, the classical MHT technique [152] by itself is computationally exponential both in time and memory. Therefore, we use an algorithm which is an efficient approximation to the MHT algorithm [58] (originally developed by Murty [137]) to generate directly the '*K*-best' hypotheses in approximately polynomial time without explicitly enumerating all possible hypotheses. This is a significant contribution to the practical application for the MHT methodology and has recently been shown to be approximately three orders of magnitude faster than previous hypothesis generation strategies [56].

The advantage of using a multiple-model based tracking is that the varied motion of features captured in an image sequence can be tracked reliably. The motion model switching capability of a multiple model framework has been demonstrated to track features much more accurately than a tracker based on a single motion model [5, 77, 206]. Amongst the many type of multiple model filters available [5], we consider two types of filters in this chapter. First, we demonstrate the performance of the more generally used *fixed* form of the filter, which we refer to as the MMF in this chapter, and later we introduce an extended version of MMF, which is referred to as the Interacting Multiple Model (IMM) algorithm. The *fixed* form of the filter assumes that there is no model switching during the estimation process. It operates in one of many modes (models) available from a bank of filters. A further improvement of the MMF filter is the IMM algorithm [5, 28]. The IMM is a sub-optimal tracking algorithm, which can automatically switch motion models according to a Markov chain process. The IMM by itself was originally developed for surveillance applications such as radar tracking, air craft tracking etc. and has been shown to track a target of varying motion efficiently. In this chapter we refer to MHT coupled with MMF as the *MHT-MMF* tracker, and MHT coupled with IMM as the *MHT-IMM* tracker. Details of these trackers are discussed in sections 4.5 and 4.6 respectively.

The tracking technique we propose relies on features (corners) extracted from each of the frames of a given image sequence. The tracking process is virtually independent of the feature extraction procedure. Therefore, as a first step we extract corner-point features from every frame of a sequence (details discussed in section 4.3) prior to tracking the features of interest. It is important that the corner features extracted are well localised and stable for reliable tracking as discussed in chapter 3. The extracted corners (in each frame) are later used as measurements for the tracking filter, to guide the tracker to follow the correct feature trajectory.

We demonstrate our results based on a variety of image sequences. The sequences considered are the PUMA sequence (30 frames), the Toy car sequence (9 frames), the Walking Man sequence (50 frames), The Rubic sequence (20 frames), the Road sequence (48 frames), and a Waving Hand sequence (75 frames).

This chapter is organised as follows: Section 4.2 briefly details the MHT methodology. Section 4.3 provides details of the corner extraction procedure. Section 4.4 details the problems in relation to single model based tracking and the necessity for a multiple model method for reliable tracking. Section 4.5 outlines the general *fixed* form of the multiple model algorithm (MMF), and Section 4.6 details the basic operation of the IMM algorithm. In Section 4.7 we provide the motion models used for MMF and IMM filters. In section 4.8 we provide our results. In section 4.9 we give a direct comparison of the 2 types of trackers considered. Section 4.10 gives a general discussion, and finally Section 4.11 provides the conclusion.

4.2 Multiple Hypothesis Algorithm

The Multiple Hypothesis Tracking (MHT) algorithm was originally developed by Reid [152] in the context of multi-target tracking. Fig. (4.1) outlines the basic operation of the MHT algorithm. The overall procedure is an iterative process that forms a feedback loop as shown by Fig. (4.1). Iteration k begins with the set of current hypotheses from iteration (k-1). Each hypothesis represents a different set of assignments of measurements to features. The extracted measurements are matched to predictions based on some distance metric such as the *Mahalanobis* distance (see [58] for details). After matching, each global hypothesis (from iteration (k-1)) has an associated ambiguity matrix Ω, which is generated as shown by Fig. (4.2b) for a simple example with 2 known geometric features and four new measurements. From Ω it is necessary to generate a set of legal assignments (In Fig. (4.2a), the black elliptical blobs are the current position of the features T_1 and T_2, and the white elliptical regions correspond to the validation gate within which possible measurements are searched for the respective features). Each subsequent child hypothesis represents one possible interpretation of the new set of measurements and, together with its parent hypothesis, represents one possible interpretation of all past measurements (the reader is referred to [58, 152, 187] for complete details of MHT). The hypothesis generation procedure and a brief mathematical framework for MHT underlying our work are given in Appendix B.

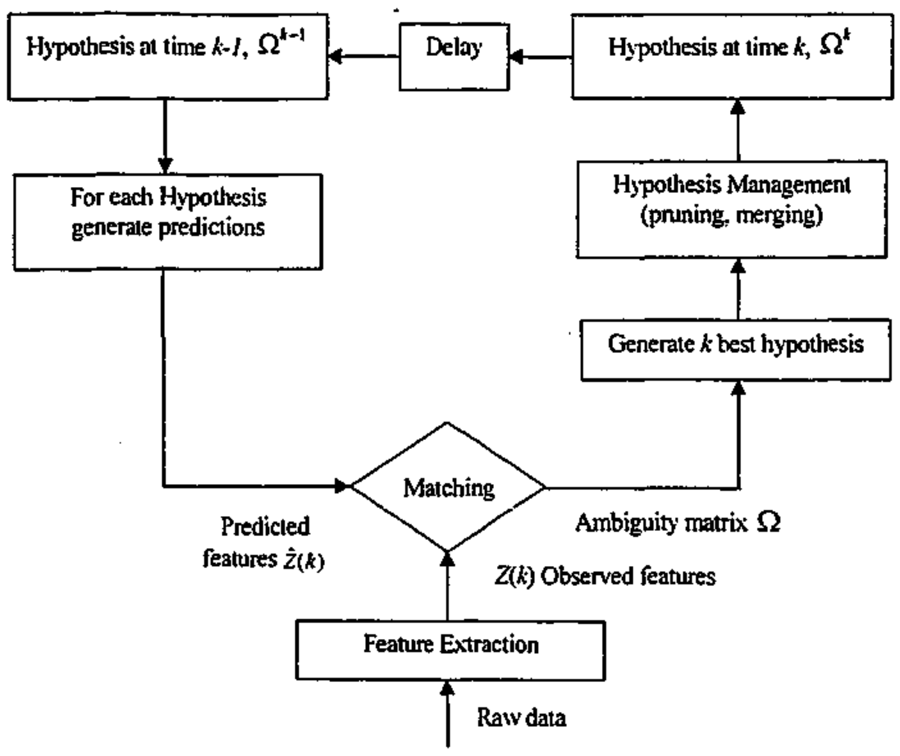
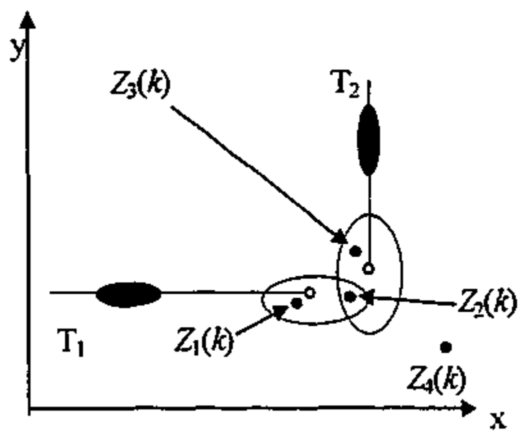

Hypothesis at time k-1, Ω^{k-1}
Delay
Hypothesis at time k, Ω^{k}
For each Hypothesis generate predictions
Hypothesis Management (pruning, merging)
Generate k best hypothesis
Matching
Predicted features $\hat{z}(k)$
Ambiguity matrix Ω
$Z(k)$ Observed features
Feature Extraction
Raw data

Figure 4.1: Outline of the multiple hypothesis algorithm for one cycle.



$$\begin{array}{cc} & \begin{array}{cccc} T_F & T_1 & T_2 & T_N \end{array} \\ \Omega = & \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \left\{ \begin{matrix} z_1(k) \\ z_2(k) \\ z_3(k) \end{matrix} \right. \end{array}$$

(a) *(b)*

Figure 4.2: (a) Predicted target locations and elliptical validation regions for a situation with two known geometric features (T_1 and T_2) and four new measurements ($z_1(k)$, $z_2(k)$, $z_3(k)$, $z_4(k)$) which are shown with black dots. The white dots are the estimated positions of T_1 and T_2. (b) Hypothesis matrix for the situation depicted in Fig. 4.2(a). Note that, as summarised by Ω, T_1 has valid measurements $z_1(k)$, $z_2(k)$, and T_2 has valid measurements $z_2(k)$, $z_3(k)$. $z_4(k)$ is out of the validation region (calculated by the Mahalanobis distance) and not considered by T_1 and T_2. In 4.2(b) the 1st and last columns entries are always 1, that is, a false alarm (T_F) or a new feature appearing (T_N) is possible at any given time k. See Appendix B for further details.

Because of the exponential complexity of the multiple hypothesis approach, only an approximation to the MHT algorithm can be practically implemented. In particular, it is simply not feasible to search the entire space of hypotheses in order to determine the most likely set of assignments. In order to contain the growth of the tree, it is necessary to prune unlikely branches. To do this intelligently, the probability of each hypothesis can be used to guide a pruning strategy efficiently (see [54, 55] for details on pruning mechanism). An approximation of the classical MHT, using Murty's algorithm [137] (we call it the modified MHT algorithm) has been shown to limit the combinatorial explosion of the classical MHT algorithm [152] and also shown to resolve data association uncertainty reliably. The successful application of such an algorithm has been reported in [58, 180]. For the remainder of this paper all references to MHT refers to the modified version of the MHT algorithm as provided by Cox *et al.* [58].

4.3 Feature Extraction

Extraction of accurate, well localised, and stable corner features are essential for reliable tracking. While this is a current research issue, our earlier study on comparison on corner detectors for tracking (reported in chapter 3 [179]), revealed that Harris [85] and Lucas-Kanade corner detectors [190] provide stable enough features for tracking in long sequences. Therefore, we have used these 2 corner detectors for the tracking applications reported in this chapter. In our application, the position of features that appear in the first frame of a sequence are predicted in the subsequent frames (matched /discarded) by the MHT framework. The implementation of the MHT uses the *Mahalanobis* distance as the main validation gate, and further, to supplement the search area, a correlation matching strategy (based on a 5x5 patch size, as explained in chapter 3.4.2) is employed.

For the PUMA, the Walking man and the Road sequences, we used the corner detector proposed by Harris [85]. For the Toy car, the Waving Hand, and the Rubic sequences we used the KLT corner detector [190] (the reason for 2 different corner detectors is that these provide the best corners representing all parts of a given frame). We maintained the number of corners extracted per frame to around 40-60. This is achieved by adjusting the appropriate parameters of the corner detection algorithm. Limiting the number of corners to a reasonable quantity is preferable purely for clarity and computational purposes.

4.4 Tracking Features

Prior to the work presented in this chapter, we used a single model Kalman Filter (KF) [199, 74, 5, 6] based on a known motion model to track the features of interest (employing the MHT framework within the tracker). But one of the drawbacks of this approach is that, the correct motion model needs to be presented to the KF in order for the tracking to be successful. Often, one does not know the motion described by the image sequence in advance. To avoid such a problem it is advisable to use a more robust and flexible approach to identify the correct motion model in order for the KF to track the features correctly and reliably.

It is also quite well known that a potential weakness of an estimator based on a single motion model is that it can lead to under-modelling and over-modelling [157]. Under-modelling, occurs when the Kalman filter state model does not describe the actual state adequately. It is possible to construct a Kalman filter that adequately describes any type of motion likely to be observed in feature tracking scenario, thus removing the under-modelling problem. Such a Kalman filter would have a large number of states and have a complex state model. A Kalman filter of this type, although possible to construct, would not be practical in a noisy environment. High order Kalman filters containing states that are derivatives of derivatives tend to be very sensitive to noise. Kalman filters consisting of large numbers of states may also be over complex for the majority of the time when the observed signal may be filtered using simpler Kalman filters.

To overcome this limitation, one solution is to use a number of filters based on different motion models, (sub-models) covering the range of possible expected observed motions, and to some how combine the estimates from these filters based on the expectation of each model being the correct descriptors of the features' motion. Such a system can be achieved by using a multiple model filtering based algorithms [5, 130]. As well as improving estimation accuracy, such systems could also help in segmenting a scene into independently moving objects. It has been proposed [157] that the segmentation process may be performed by utilising the confidence/belief measures generated by the individual filters that make up the multiple motion model system (for example segmenting/grouping cars that move with similar velocities). If all objects in a scene are assumed to be rigid, all points on a single object will move in an identical fashion, i.e, with the same motion model. Further description of the multiple model filtering algorithm is given in the next section.

4.5 The Multiple Model Filtering Algorithm

In the multiple model approach, it is assumed that the system obeys one of a finite number of models. Such systems are called hybrid because they have both continuous (noise) uncertainties as well as discrete uncertainties [5]. The system uses a Bayesian framework to calculate the probability of each model in operation. That is, starting with prior probabilities of each model being correct (ie., the system is in a particular mode), the corresponding posterior probabilities are obtained.

Initially we consider the case where the model the system obeys is fixed, that is no switching from one mode to another occurs during the estimation process. In this chapter we name such a system as the Multiple Model Fixed (MMF) algorithm. A typical MMF is shown in Fig. 4.3 (for further description refer to [5, 187]). The MMF consists of r separate Kalman filters, each based on a particular state model. The model (M), assumed to be in effect throughout the process, is one of r possible models (the system is in one of r modes):

$$M \in \{M_j\}_{j=1}^{r}$$

The prior probability that M_j is correct (the system is in mode j) is

$$P\{M_j | Z^0\} = \mu_j(0) \qquad j = 1,\ldots,r$$

where Z^0 is the prior information and

$$\sum_{j=1}^{r} \mu_j(0) = 1$$

since the correct model (or a model closest to the correct model) is among the assumed r possible models ($\mu_j(0)$ corresponds to the probability that the j-th model being correct at time step 0). It will be assumed that all models are linear Gaussian.

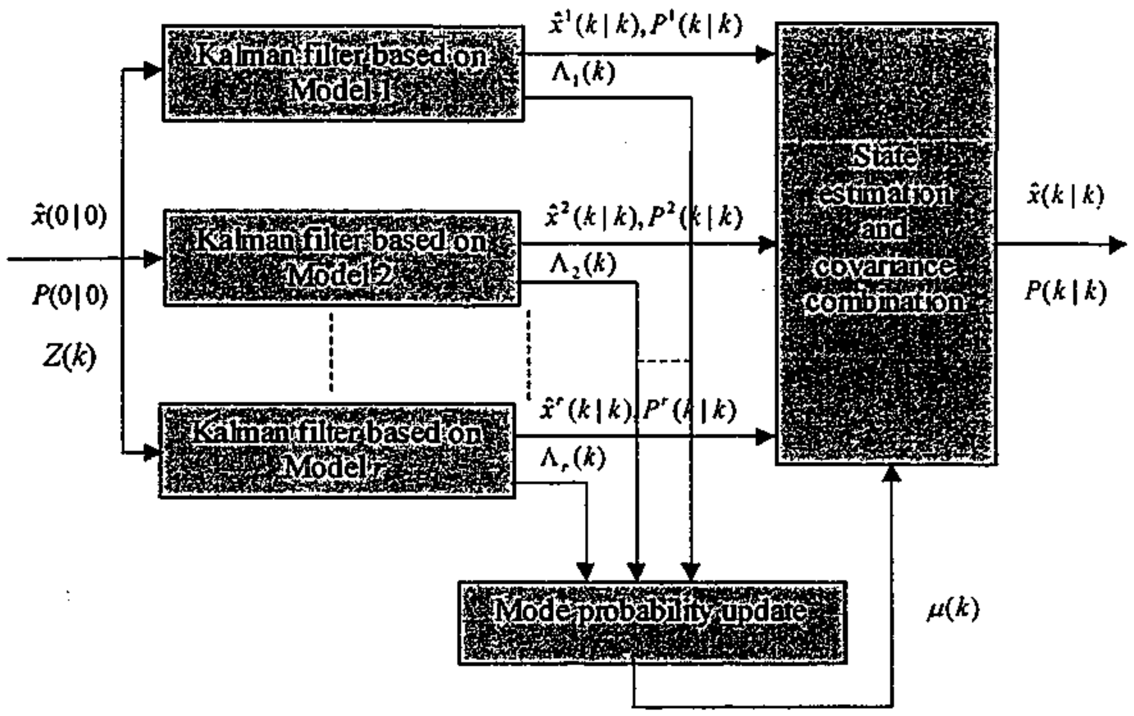
The overall state estimate is the linear combination of the state estimates generated by the individual Kalman filters, and is calculated using the following equation (for the state and covariance updates).

$$\hat{x}(k|k) = \sum_{j=1}^{r} \mu_j(k).\hat{x}^j(k|k) \tag{4.1}$$

$$P(k|k) = \sum_{j=1}^{r} \mu_j(k)\left\{P^j(k|k) + [\hat{x}^j(k|k) - \hat{x}(k|k)][\hat{x}^j(k|k) - \hat{x}(k|k)]^T\right\}$$

where $\hat{x}^j$ is the state vector of the j-th Kalman filter, $\mu_j(k)$ is the probability that the actual system model, M, equals the j-th model M_j at time k given the past observations, Z^{k-1}. r is the total number of filters considered. The weighting factors $\mu_j(k)$ are recursively updated using:

$$\mu_j(k) = \frac{p[z(k)|Z^{k-1}, M_j]\mu_j(k-1)}{\sum_{i=1}^{r} p[z(k)|Z^{k-1}, M_i]\mu_i(k-1)} \qquad j = 1,\ldots,r \tag{4.2}$$



Kalman filter based on Model 1
Kalman filter based on Model 2
Kalman filter based on Model r
$\hat{x}(0|0)$
$P(0|0)$
$Z(k)$
$\hat{x}^1(k|k), P^1(k|k)$
$\Lambda_1(k)$
$\hat{x}^2(k|k), P^2(k|k)$
$\Lambda_2(k)$
$\hat{x}^r(k|k), P^r(k|k)$
$\Lambda_r(k)$
State estimation and covariance combination
$\hat{x}(k|k)$
$P(k|k)$
Mode probability update
$\mu(k)$

Figure 4.3: A typical MMF for r filters.

To prevent $\mu_j(k)$ from becoming too small, effectively "turning off" the j-th filter, values of $\mu_j(k)$ should be limited to a minimum value, $\mu_{\min}$ [5,157]. The residuals, v_j and estimated covariance of the residual's, S_j are given by (seeAppendix B for the full KF recursion that was employed),

$$v_j(k) = z(k) - H_j\hat{x}^j(k) \tag{4.3}$$

$$S_j(k) = H_jP(k)(H_j)^T + R_j \tag{4.4}$$

where H, P & R are the observation, state covariance, and the measurement noise matrices respectively. These quantities are used to compute the likelihood function of the r-filters. The probability density is assumed Gaussian if a model is linear [5], and with this assumption, the likelihood function for mode j at time k is given by:

$$\begin{aligned}\Lambda_j(k) &\equiv p[z(k) \mid Z^{k-1}, M_j] = p[v_j(k)] \\ &= \mathrm{N}[v_j(k);0;S_j(k)] \\ &= \frac{1}{(2\pi)^{m/2}\left|S_j(k)\right|^{1/2}} \exp\left\{-\tfrac{1}{2} v_j^T(k) S_j^{-1}(k) v_j(k)\right\}\end{aligned} \tag{4.5}$$

where m is the number of measurements at time k. The extra computation in updating the weighting factors compared to the normal Kalman filter is therefore negligible. It should be noted that the separate Kalman filters may be run simultaneously and in parallel.

Equation (4.5) assumes that the residuals v_j, are Gaussian and zero mean. Hence, the MMF algorithm effectively chooses between filters based on the size of the mean of their residuals, with the one having the smallest mean (i.e. the one nearest to zero) being the *correct* filter (in other words, MMF chooses the most appropriate filter based on the likelihood functions). This is because, a filter which has been modelled correctly will produce residuals with near-zero mean (see [130, 157] for more details). From a mathematical point of view, the probability of each model being correct is obtained according to Eq. (4.2) based on its likelihood function (Eq. 4.5) relative to the other filters' likelihood functions.

An alternative version of the MMF algorithm was proposed by Mealy and Tang [131]. They applied the multiple model estimation technique to a terrain height correlation system using a bank of identical Extended Kalman Filters (EKFs) each initialised with different state estimates. The difference between this implementation and the MMF described above, is that after the initial transients settle down, the filter with the highest probability, $\mu_j(k)$ was used to estimate the feature position. i.e. there was no combination of the individual filter state estimates (usually performed using equation 4.1). Instead, the filter with the highest probability was allowed to track the subsequent object (features) alone, and all other filters were turned off. This *decision* to switch between a multiple model tracking mode and a single model tracking mode was performed based on a comparison of the residuals between filters. The advantage of such a system is that the computational overhead loads required by r filters was eliminated. However, the danger of this strategy is that the adaptive properties of the MMF algorithm are lost when switching to a single model mode. Tobin and Maybeck [189] also proposed using the filters with the highest probability alone as the state estimator, except that in their implementation all the filters remained active ensuring that the MMF's adaptive capabilities remain.

4.5.1 Multiple Model Algorithm to Cope with Multi-Order States

The MMF algorithm assumes that each separate KF has identical states and is of the same order. This can be seen from equation 4.1, where combination of the individual estimates requires all the states to be present in each filter. However, this restriction is not imposed when calculating the hypothesis conditional probability (equation 4.2). This equation requires the separate KFs to have common measurement state variables only; the conditional probability (equation 4.5) is composed entirely from measurement states.

Since it is only the state estimate combination equation that requires common state variables among all the KFs, the standard MMF algorithm may be extended to cope with filters having different structures and different orders (but with common measurement states). Such a multi-order/differing state multiple model adaptive estimator uses the same probability equations (Eqs. 4.2, 4.5) as the standard MMF algorithm, but requires the state estimate combination equation to be re-written to account for any *missing* states. This is done as follows:

$$\hat{x}_i(k) = \frac{\sum_{j=1}^{r} \mu_j(k) D_{ji} \hat{x}_{ji}(k)}{D_i^T \mu(k)} \tag{4.6}$$

where, i is the state considered,

r	- total number of Kalman filters,
$\hat{x}_{ji}(k)$	- the i-th state of the j-th Kalman filter,
$\mu_j(k)$	- the weighting factor of the j-th filter,
$\mu(k)$	- the vector of weighting factors for all r filters,
D_{ji}	- the membership value (one or zero) of the j-th filter for the i-th state,
D_i	- the i-th column of the membership matrix D [187].

4.5.2 Limitations of MMF

A problem with the MMF based systems is that, their fundamental design does not cater for motion model switching automatically, which can cause problems in tracking a target whose motion is varying. In cases where a target switches to a different motion model during the course of tracking, the MMF based methods can fail or even converge to the wrong motion model [5, 185] (see section 4.8.3 for an example). However, there are ad-hoc modifications which are available (mentioned in section 4.8) for MMF algorithms to cope with model switching [5] as we have demonstrated in the results section. In spite of these modifications, the mismatched filter's errors can still grow to unacceptable levels. Thus, re-initialization of the filters that are mismatched is in general needed (see also the discussion in section 4.10 for further explanation).

To overcome the drawbacks of MMF based algorithms, an enhanced algorithm that could cope with automatic model switching called the Interacting Multiple Model (IMM) was proposed by Bar-Shalom *et al.* [5, 28]. The IMM algorithm is able to cope with mode changes during motion transition and is capable of switching from one mode of motion to another efficiently (including tracking of manoeuvring feature targets) during the course tracking. The operation of the IMM is discussed further in the next section.

4.6 The IMM Algorithm

The Interacting Multiple Model (IMM) approach is a sub-optimal technique for switching motion models (mode (model) jumping process) during the estimation process [5]. The system model at time k is assumed to be among the possible r modes

$$M(k) \in \{M_j\}_{j=1}^{r}.$$

The motion mode switching process is assumed to be a Markov process [6, 5] with known mode transition probabilities (these are design parameters).

In the IMM approach, at time k the state estimate is computed under each possible current model using r filters, with each filter using a different combination of the previous model-conditioned estimates (*mixed initial condition*).

A typical IMM is shown in Fig. 4.4. For further description refer to [5]. The r separate filters are each based on a different motion model (each model can be of a different order).

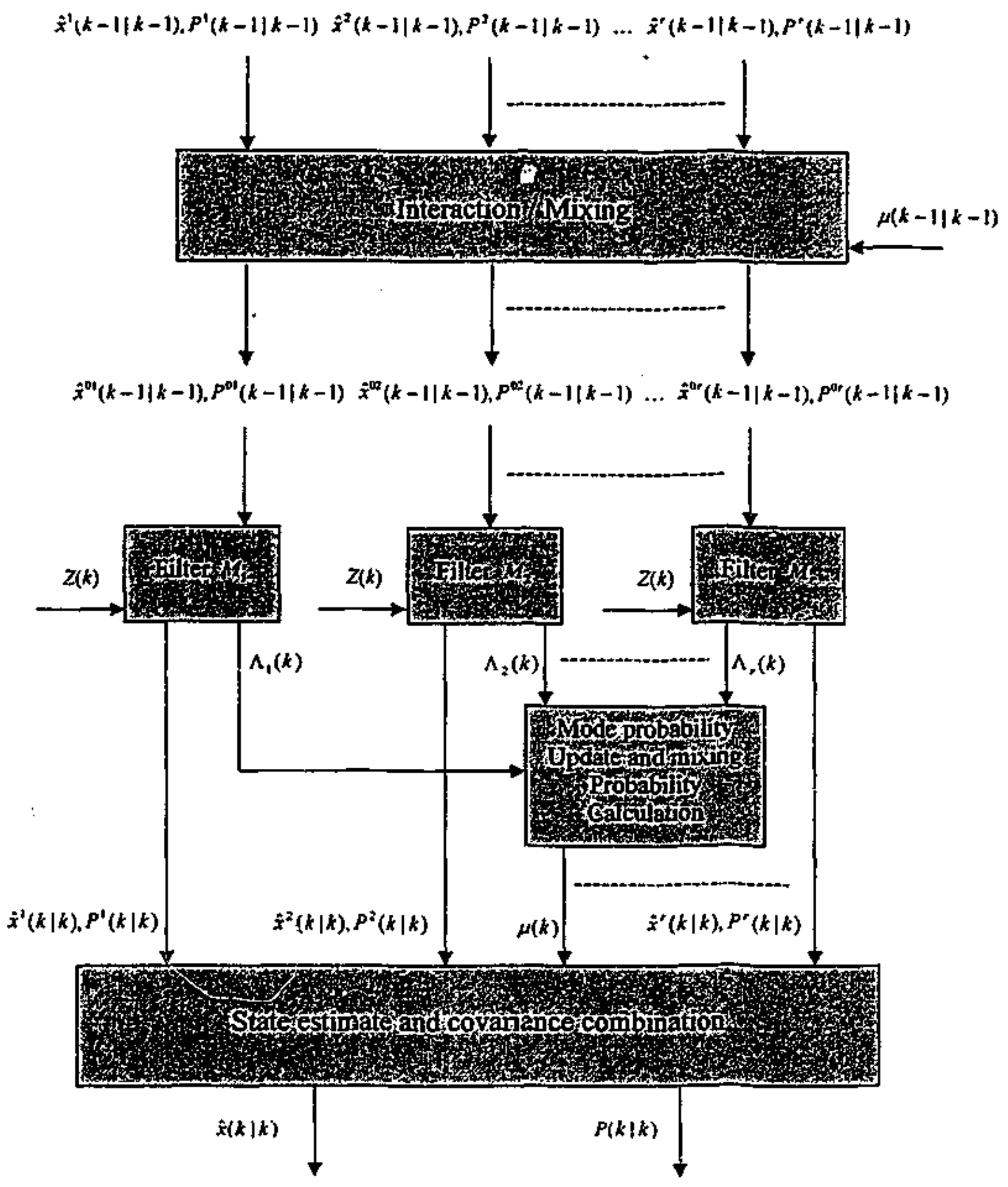

$\hat{x}^1(k-1|k-1), P^1(k-1|k-1)$ $\hat{x}^2(k-1|k-1), P^2(k-1|k-1)$ ... $\hat{x}^r(k-1|k-1), P^r(k-1|k-1)$
Interaction / Mixing
$\mu(k-1|k-1)$
$\hat{x}^{01}(k-1|k-1), P^{01}(k-1|k-1)$ $\hat{x}^{02}(k-1|k-1), P^{02}(k-1|k-1)$ ... $\hat{x}^{0r}(k-1|k-1), P^{0r}(k-1|k-1)$
$Z(k)$
Filter M_1
Filter M_2
Filter M_r
$\Lambda_1(k)$
$\Lambda_2(k)$
$\Lambda_r(k)$
Mode probability Update and mixing Probability Calculation
$\hat{x}^1(k|k), P^1(k|k)$
$\hat{x}^2(k|k), P^2(k|k)$
$\mu(k)$
$\hat{x}^r(k|k), P^r(k|k)$
State estimate and covariance combination
$\hat{x}(k|k)$
$P(k|k)$

Figure 4.4: IMM Algorithm with r filters (one cycle).

One cycle of the IMM algorithm consists of the following:

1) Calculation of the mixing probabilities (i, j=1,...,r). The probability that mode M_i is in effect at time k conditioned on Z^{k-1} (measurements) is given by,

$$\mu_{i|j}(k-1|k-1) = \frac{1}{\bar{c}_j} p_{ij} \mu_i(k-1) \qquad i, j = 1, \ldots, r \tag{4.7}$$

where the mode transition probability is p_{ij} (assumed known), and the normalising constants are:

$$\bar{c}_j = \sum_{i=1}^{r} p_{ij}\mu_i(k-1) \qquad j = 1,\ldots,r \tag{4.8}$$

2) Mixing (j=1,...,r). Starting with $\hat{x}^i(k-1|k-1)$ one computes the mixed initial condition for the filter matched to $M_j(k)$ is,

$$\hat{x}^{oj}(k-1|k-1) = \sum_{i=1}^{r} \hat{x}^i(k-1|k-1)\mu_{i|j}(k-1|k-1) \qquad j=1,\ldots,r \tag{4.9}$$

The covariance corresponding to the above is,

$$P^{oj}(k-1|k-1) = \sum_{i=1}^{r} \mu_{i|j}(k-1|k-1)\{\, P^i(k-1|k-1) + [\hat{x}^i(k-1|k-1) - \hat{x}^{oj}(k-1|k-1)]. \\ [\hat{x}^i(k-1|k-1) - \hat{x}^{oj}(k-1|k-1)]^T \,\} \\ j = 1,\ldots,r \tag{4.10}$$

3) Mode-matched filtering (j=1,...,r). The estimate Equation (4.9) & (4.10) are used as input to the filter matched to $M_j(k)$, which uses $z(k)$ to yield $\hat{x}^j(k|k)$ and $P^j(k|k)$. The likelihood functions corresponding to the r filters are given by,

$$\Lambda_j(k) = N\left[x(k); \hat{z}^j[k|k-1; \hat{x}^{oj}(k-1|k-1), S^j[k; P^{oj}(k-1|k-1)]\right], \\ j = 1,\ldots,r \tag{4.11}$$

4) Mode probability update (j=1,...,r). This is given by,

$$\mu_j(k) = \frac{1}{c}\Lambda_j(k)\bar{c}_j \quad j=1,\ldots,r \tag{4.12}$$

where $\bar{c}_j$ is same as Eq. (4.8) and, $c = \sum_{j=1}^{r} \Lambda_j(k)\bar{c}_j$ is the normalisation constant for equation (4.12).

5) Estimate and covariance combination. Combination of the model-conditioned estimates and covariances is done according to the mixture equations:

$$\hat{x}(k|k) = \sum_{j=1}^{r} \hat{x}^j(k|k)\mu_j(k) \tag{4.13}$$

$$P(k|k)=\sum_{j=1}^{r}\mu_j(k)\left\{P^j(k|k)+[\hat{x}^j(k|k)-\hat{x}(k|k)][\hat{x}^j(k|k)-\hat{x}(k|k)]^T\right\} \quad (4.14)$$

Equations (4.13) & (4.14) are only for output purposes (they are the same as in Eq. (4.1)). It is not part of the algorithm recursions.

In our tracking implementation, each point feature of interest is applied with a coupled tracking algorithm (MHT & IMM). The resulting tracker is referred to as the *MHT-IMM* algorithm.

4.7 Motion Models

To test our complete tracking system we employed motion models from the following:

(i) A first order constant position model (M5)

(ii) A second order constant acceleration model (M1)

(iii) A second order constant velocity model (M2)

(iv) A second order constant turn model (M3)

(v) A third order acceleration model (M4)

The description of each motion model is briefly discussed in the following sections.

4.7.1 First Order Motion Model (M5)

The state vector **x**(k) at time k, the measurement matrix H, and the state transition matrix ($\mathbf{F}_{M5}$) for a first order model is given as follows:

$$\mathbf{x}(k)=\begin{bmatrix}x & y\end{bmatrix}^T, \qquad \mathrm{H}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}, \qquad \mathbf{F}_{M5}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

The only measurement required is the feature position (x, y). The initial values for **x**(k) are set to the feature position in the first frame.

The process noise matrix **Q** is set to, $\mathrm{Q}=\begin{bmatrix}T & 0\\ 0 & T\end{bmatrix}\tilde{q}$, where T is the sampling time and $\tilde{q}$ is a small power spectral density for the process noise (see Appendix B.5 for a Kalman filter diagram).

4.7.2 Second Order Motion Models (M1 – M3)

The state vector $\mathbf{x}(k)$ at time k, and the measurement matrix H for the tracking filter is set as follows for the second order models.

$$\mathbf{x}(k) = \begin{bmatrix} x & \dot{x} & y & \dot{y} \end{bmatrix}^T, \qquad \mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Velocity (x and y direction) estimates were initialised to zero. Alternatively if the initial velocity is known, one can set the filter initial velocity to the known value.

The state transition matrices (**F**) chosen for the Models M1-M3 are given below. Note that M1, M2 employ the same **F**. A near acceleration model (M1) is obtained by choosing a larger value for the power spectral density $\tilde{q}$ (~10) of the process noise [5]. A small $\tilde{q}$ (~0.1) results in a constant velocity model - M2 (that is, the changes in the velocity have to be small compared to the actual velocity). A second order constant turn model (M3) is obtained by using a state transition matrix ($\mathbf{F}_{M3}$) as shown below, where T is the sampling time and ω is a constant turn rate for M3.

$$\mathbf{F}_{M1,M2} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{F}_{M3} = \begin{bmatrix} 1 & \frac{\sin\omega T}{\omega} & 0 & \frac{\cos\omega T - 1}{\omega} \\ 0 & \cos\omega T & 0 & -\sin\omega T \\ 0 & \frac{1-\cos\omega T}{\omega} & 1 & \frac{\sin\omega T}{\omega} \\ 0 & \sin\omega T & 0 & \cos\omega T \end{bmatrix}$$

The process noise matrix (Q) is set as follows for all second order models (but different values of $\tilde{q}$ are used):

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{3}T^3 & \frac{1}{2}T^2 & 0 & 0 \\ \frac{1}{2}T^2 & T & 0 & 0 \\ 0 & 0 & \frac{1}{3}T^3 & \frac{1}{2}T^2 \\ 0 & 0 & \frac{1}{2}T^2 & T \end{bmatrix} \tilde{q}$$

4.7.3 Third Order Motion Models (M4)

For the third order motion model, an acceleration component $(\ddot{x}, \ddot{y})$ is included as part of the state vector $\mathbf{x}(k)$, and initialised to zero. The state transition and process noise matrices were set as follows, where $\tilde{q}$ is set to a small value (typically ~0.01). See [5] for further details on specific motion models.

$$\mathbf{F}_{M4} = \begin{bmatrix} 1 & T & \frac{1}{2}T^2 & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & \frac{1}{2}T^2 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathrm{Q} = \begin{bmatrix} \frac{1}{20}T^5 & \frac{1}{8}T^4 & \frac{1}{6}T^3 & 0 & 0 & 0 \\ \frac{1}{8}T^4 & \frac{1}{3}T^3 & \frac{1}{2}T^2 & 0 & 0 & 0 \\ \frac{1}{6}T^3 & \frac{1}{2}T^2 & T & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{20}T^5 & \frac{1}{8}T^4 & \frac{1}{6}T^3 \\ 0 & 0 & 0 & \frac{1}{8}T^4 & \frac{1}{3}T^3 & \frac{1}{2}T^2 \\ 0 & 0 & 0 & \frac{1}{6}T^3 & \frac{1}{2}T^2 & T \end{bmatrix} \tilde{q}$$

4.8 Results

This section provides the results obtained for the trackers mentioned in sections 4.5 and 4.6. We initially provide the results using a tracker based on a single motion model and show instances where such a tracker can provide poor quality trajectories or even fail. We then provide results for the MHT-MMF and MHT-IMM trackers. Finally we provide a direct comparison (with regards to model switching capability) of the two trackers based on a simulated feature trajectory. We complete the experimental evaluation by applying the *MHT-IMM* tracker to a real sequence of a waving hand moving with multiple motion. The result for the latter is given in Section 4.9.

4.8.1 MHT with a Single Motion Model

The first stage of our tracking process was to extract corner features reliably, which was essential for good tracking performance as discussed before. Figure 4.5 shows the corner features extracted for selected frames from each of the image sequence considered.

Figures (4.6)-(4.10) show tracking results obtained by using the MHT algorithm based on a single motion model Kalman filter (for each of the motion models in turn, as described by the figure captions). In Figures (4.6)-(4.10), on the right side are 3 tracks picked out from the many tracks displayed on the left side. These are shown to illustrate that the inappropriate motion model selected for the tracker can result with a shorter feature trajectory or even a wrong trajectory, thus emphasising the need for a correct motion model for accurate and reliable tracking.

From these figures, it is quite clear to the naked eye that a constant acceleration model gives the best tracking performance for the PUMA and the Toy car sequences, and a constant velocity model gives the best tracking performance for the Walking man and the Road sequences (in terms of the quality and length of the trajectories obtained). When the inter-frame motion is relatively small, a constant velocity model is adequate, but for larger inter-frame motion (like the PUMA sequence) a constant acceleration model provides better results. For the Rubic sequence, the inter-frame motion is very small (as seen from Fig. 4.5e), and as a result, a constant position model (M5) tracker is able to provide feature trajectories as good as M1 or M2 (see Fig. 4.10).

An advantage of the MHT based trackers is the ability to track objects which are temporarily occluded. For example, it can be seen for the Toy car sequence, the MHT framework provides good tracking results in spite of occlusion. Fig. 4.5(b) shows the jeep and the van are occluded in part in frames 5,6 (see Appendix G), but despite occlusion the tracker retains the trajectory (Figures 4.7c,d) of the van and jeep until the final frame (assuming a correct motion model is in operation).

4.8.2 MHT with Multiple Motion Models

For experiments with *r* motion models in the filter bank, we initialised the probability of selecting a model to *(1/r)*. That is, at the start all models have an equal chance of getting selected. All motion models were also initialised with the same state to eliminate any bias. The following sub-sections provide tracking results for the 2 trackers presented in this chapter (*MHT-MMF* and *MHT-IMM*).

4.8.2.1 MHT-MMF Tracker Performance

The quantitative results obtained by using the MHT-MMF tracker are shown in Figs. (4.11)-(4.14). In these graphs we plot the model selection probability against the frame number. The probability of the correct model (most appropriate model) getting selected confirms our observation of Figs. (4.6)-(4.10). It is clear that an acceleration model is indeed the most suitable model for the PUMA and the Toy car sequences, and a velocity model is more suitable for the walking man and road sequences, and a constant position model is adequate for the Rubic sequence. We experimented with multi-order multi-type motion models in the MMF filter bank, but the final observation (that is the correct model being selected) did not depend on the combinations of models employed in the filter bank. Provided the correct model (model closest to the object's motion) is one of the models considered, the result is consistent. When only 2 models (which are similar) are used in a filter bank, care must be taken to tune the filters, as false tracking results might occur (details in section 4.10). Conditions under which the MMF algorithm might fail and the precautions to be taken are discussed in Section 4.10.

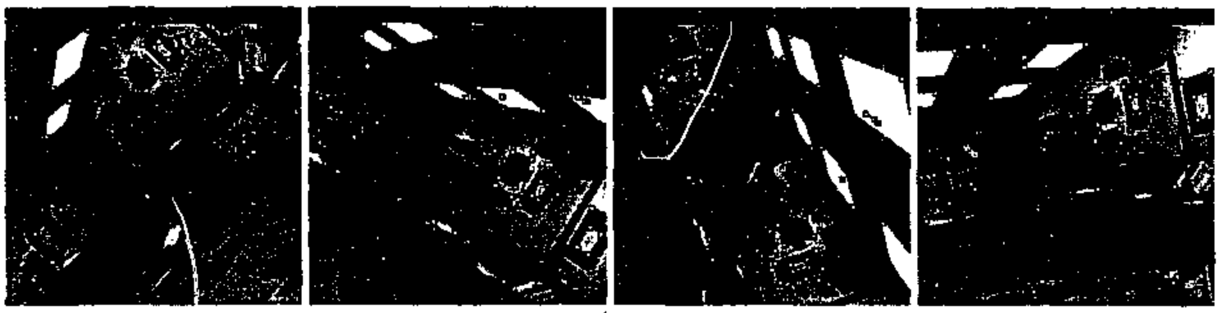
MMF framework assumes that the correct motion model is among the models in the filter bank. If the correct motion model is absent, then the tracker will converge to the model closest to the correct motion model in the filter bank. It should be noted that in the implementation of the MMF based tracker, a lower bound for $\mu_j (= \mu_{min})$ was set in order to keep the incorrect motion models alive. Otherwise the MMF framework does not have a mechanism to "revive" a model once it approaches a near zero selection probability. See the Discussion section (Section 4.10) and [157] for further explanation. We also examined the state covariance matrix associated with each motion model after every cycle to check for possible divergence, if a divergence is detected, the filter concerned is reinitialised. These undesirable ad-hoc modifications are necessary for the MHT-MMF tracker to be able to switch motion models. In spite of these modifications, the MHT-MMF tracker fails where multiple model switching is required, such an example is illustrated in Figs. (4.20)-(4.24) and discussed further in sub-section 4.8.3.

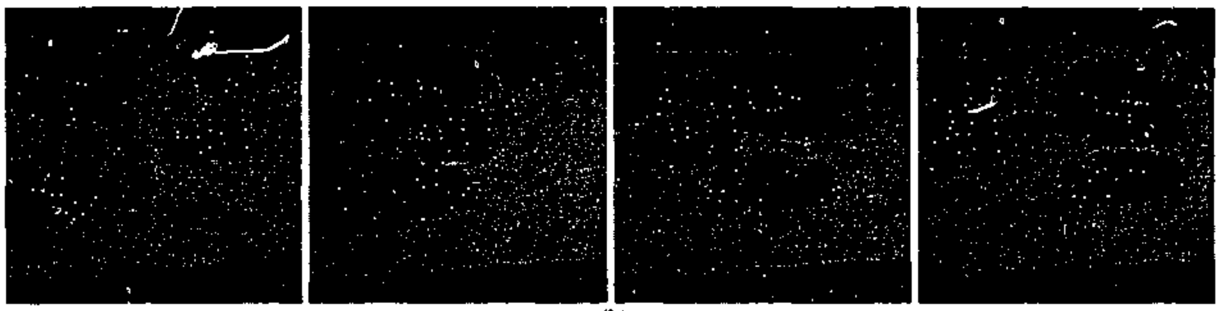
4.8.2.2 MHT-IMM Tracker Performance

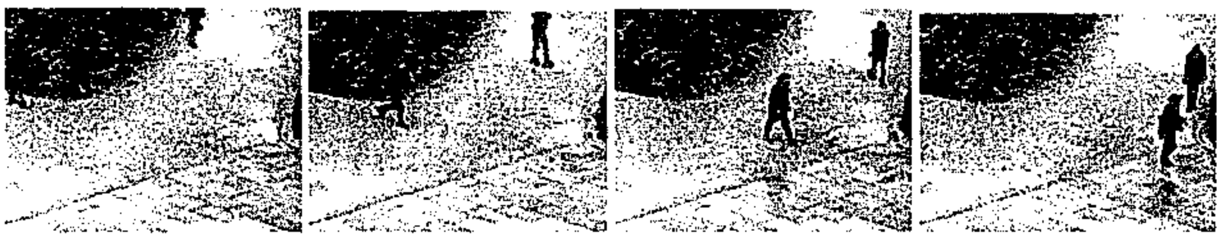
The MHT-IMM tracker overcomes most of the limitations that MHT-MMF tracker present. For the MHT-IMM tracker, there are no requirements for any modifications or assumptions for the basic IMM algorithm to cope with multiple model switching. This is one of the main advantages over using MMF trackers. The other advantage of this tracker is that the switching from one model to another is possible during the estimation process (using a Markov chain procedure), which provides faster responses to model changes.

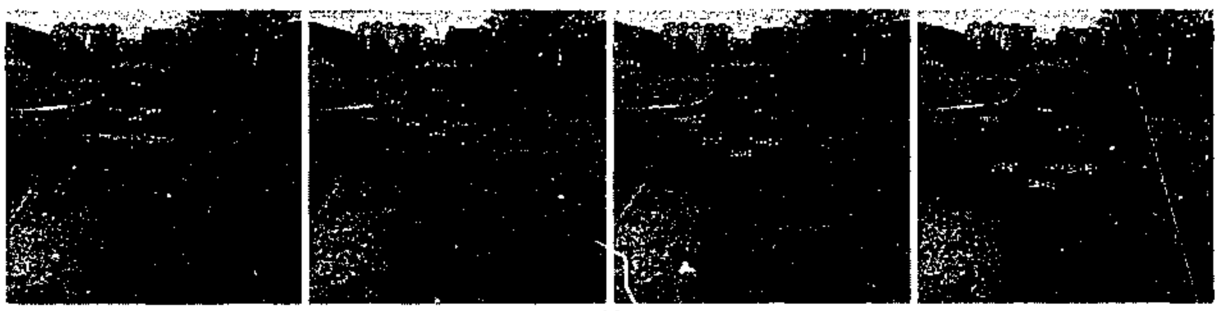
The results for MHT-IMM tracker are given in Figures (4.15)-(4.19) (model selection probability vs. frame number). As expected the MHT-IMM tracker converges to the "*most correct*" model presented. It is clearly seen from the results that after the initial transient dies, the probability for the *correct* model approaches one while the probability for the *incorrect* models decreases to zero. It can be observed that the convergence to the correct motion model is fast and smoother for the MHT-IMM tracker than the MHT-MMF tracker in almost all the examples considered. It is worth noting the results for the walking man sequence (Fig. 4.17) where the person's motion is not always constant, and there appear 'dips' in the model selection probability curve, which indicates temporary model changes (or as a weighted combination of motion models in operation). The tracker performance in terms of position (error between true and estimated) and velocity are shown in Figs. (4.15c,d)-(4.18c,d) and Fig. (4.19a,b) for one feature trajectory, and the track statistics are tabulated in Tables 4.1-4.5 (for 3 arbitrarily chosen feature tracks) for the image sequences tested. The evidence of faster model switching of MHT-IMM over MHT-MMF can be observed in Figs. 4.11 – 4.19 (compare the MHT-MMF results with MHT-IMM results). In all experimental cases considered promising tracking results are obtained (qualitatively and quantitatively).

Examples of Corners Extracted


(a)


(b)


(c)


(d)

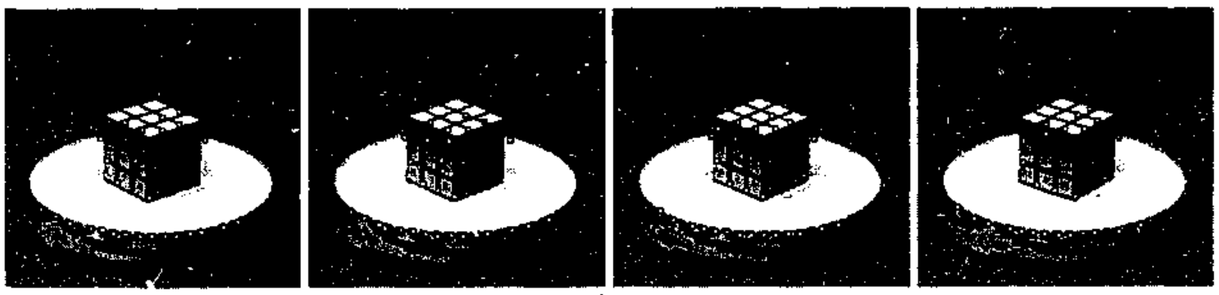

(e)

Figure 4.5: (a) Frames 1, 10, 20 and 30 of the PUMA sequence with corners extracted superimposed on the respective frames. (b) Frames 1,4,7 and 9 of the Toy car sequence. (c) Frames 1, 17, 35 and 50 of the Walking man sequence. (d) Frames 1, 16, 32 and 48 of the Road sequence. (e) Frames 1, 5, 9, 15 of the Rubic sequence. Only the best 25-50 corners were extracted from each frame of a sequence for clarity.

Tracks Obtained Using Different Motion Models (PUMA)

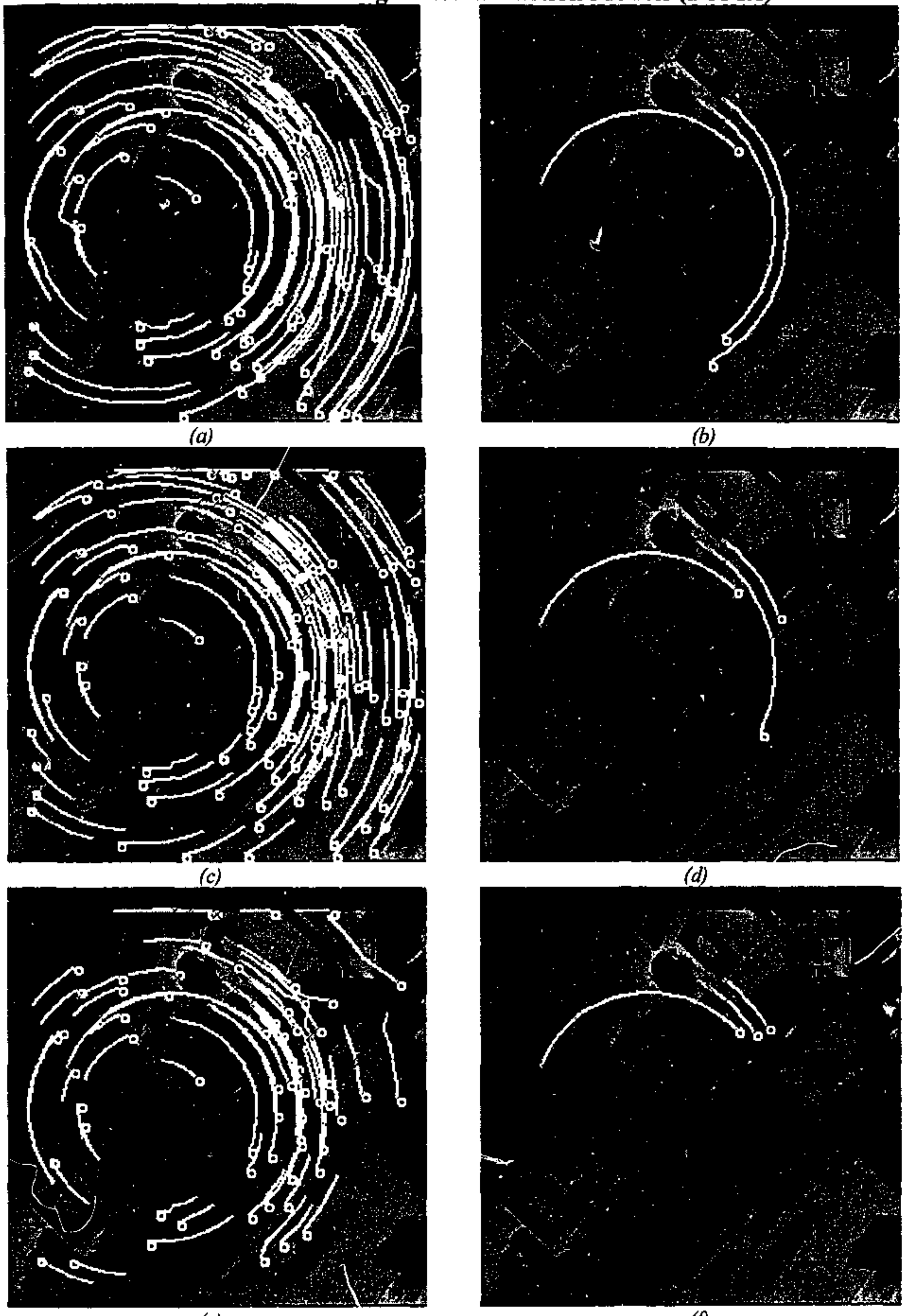


Figure 4.6: PUMA sequence track results (displayed on frame-1 for tracks which survived for length more than 6). The circle indicates the end of track: (a) Tracking performance obtained by using a constant acceleration model (M1). (b) Selected 3 tracks from (a) for illustration. (c) Tracking performance using a constant velocity model (M2). (d) The 3 tracks from (c), for the same features as shown in (b). (e) Tracking performance obtained by using a constant turn model (M3). (f) Selected 3 tracks from (e).

Tracks Obtained Using Different Motion Models (Toy Car)

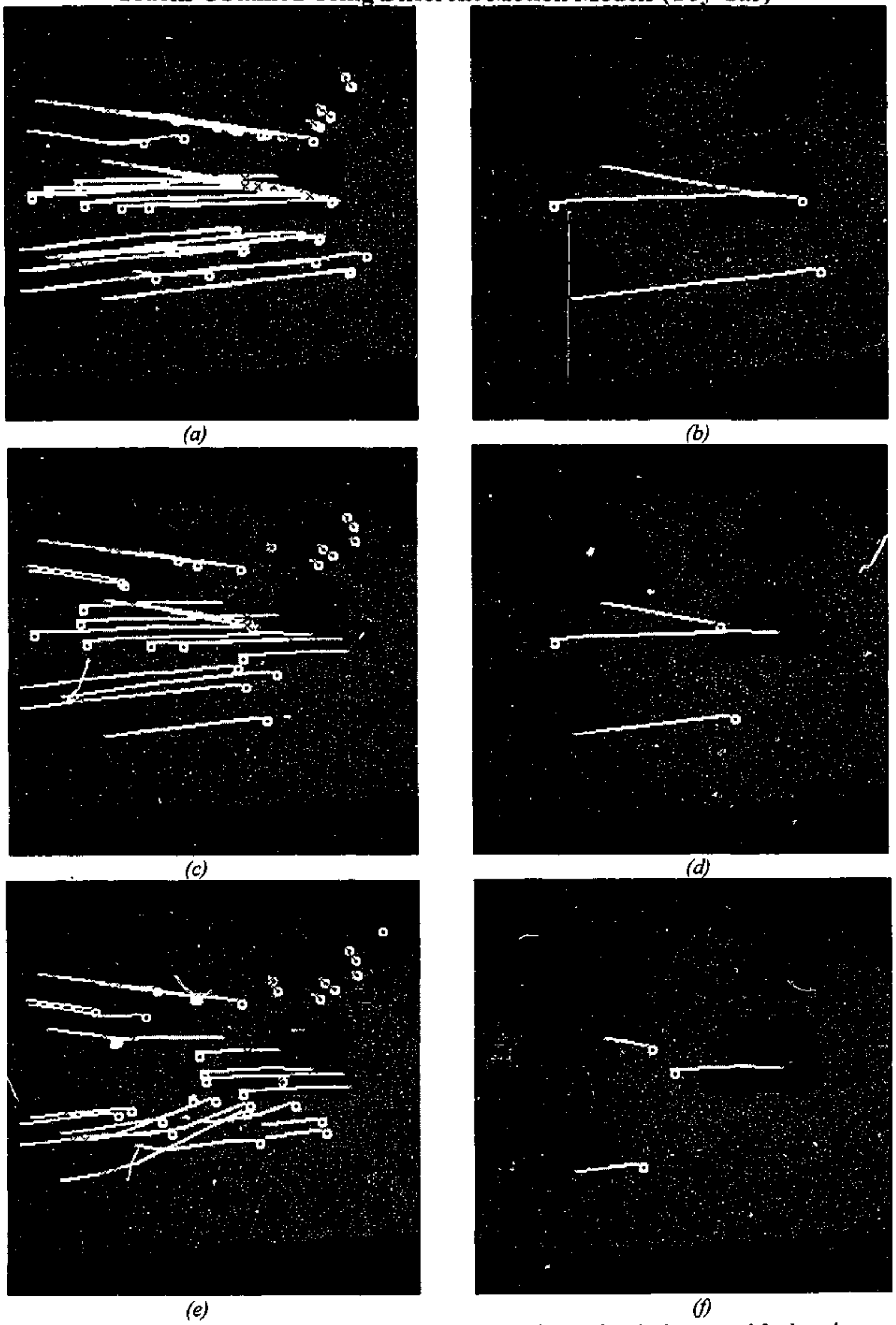


Figure 4.7: Toy car sequence track results (displayed on frame-1 for tracks which survived for length more than 5). The circle indicates the end of track: (a) Tracking performance obtained by using a constant acceleration model (M1). (b) Selected 3 tracks from (a) for illustration. (c) Tracking performance using a constant velocity model (M2). (d) The 3 tracks from (c), for the same features as shown in (b). (e) Tracking performance obtained by using a constant turn model (M3). (f) Selected 3 tracks from (e).

Tracks Obtained Using Different Motion Models (Walking Man)

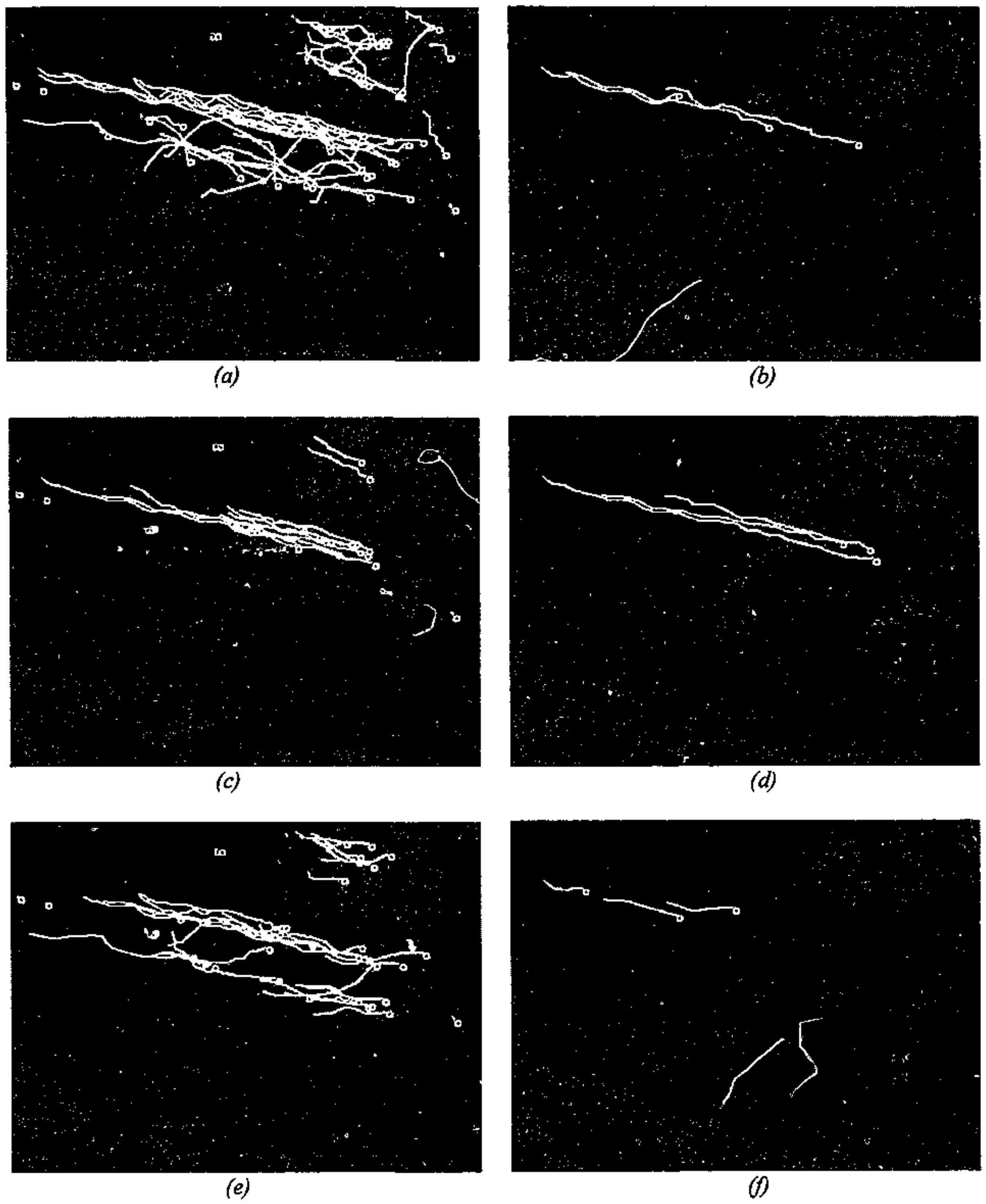


Figure 4.8: Walking man sequence track results (displayed on frame-1 for tracks which survived for length more than 12). The circle indicates the end of track: (a) Tracking performance obtained by using a constant acceleration model (M1). (b) Selected 3 tracks from (a) for illustration. (c) Tracking performance using a constant velocity model (M2). (d) The 3 tracks from (c), for the same features shown in (b). (e) Tracking performance obtained by using a constant turn model (M3). (f) Selected 3 tracks from (e).

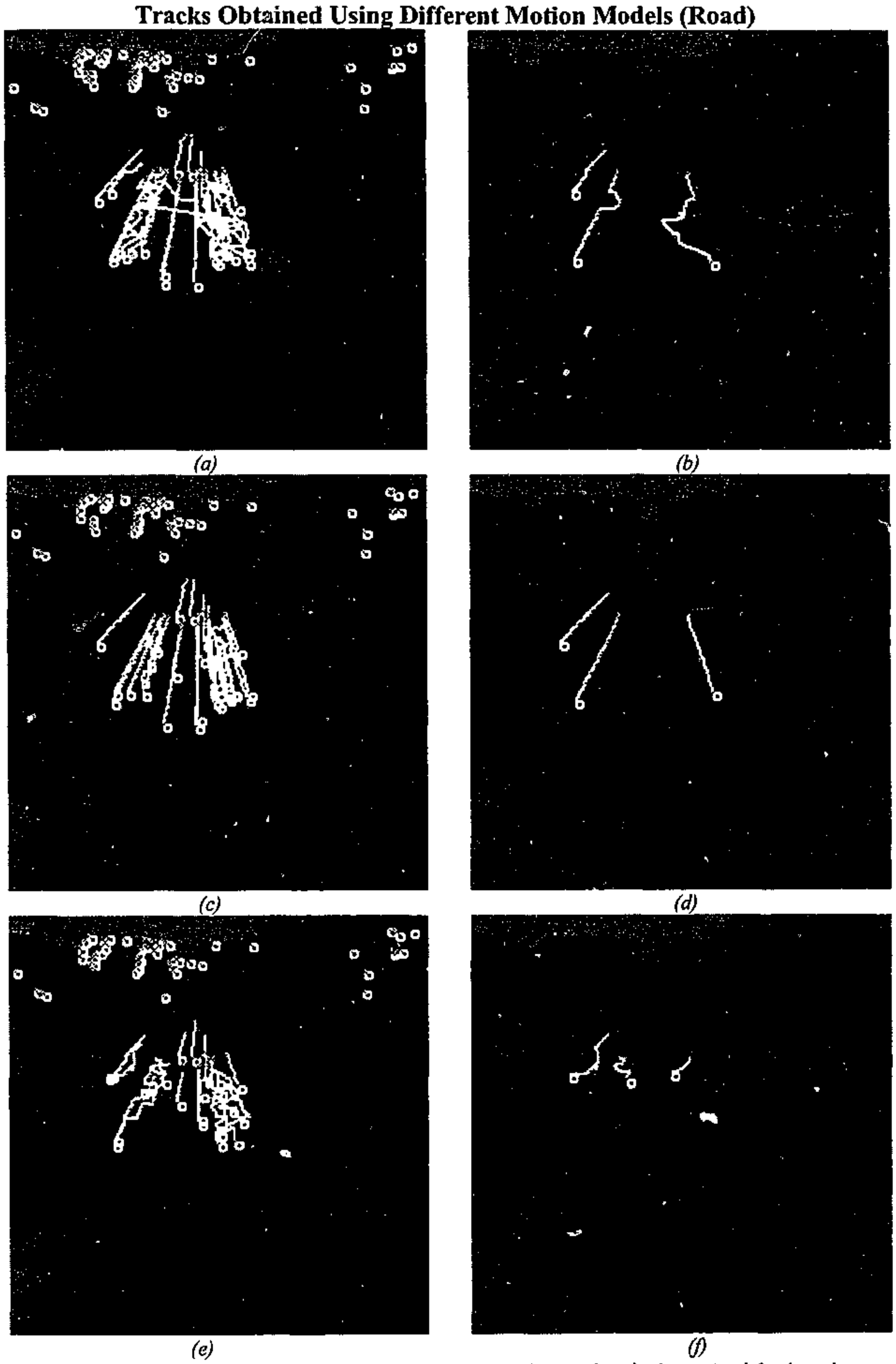
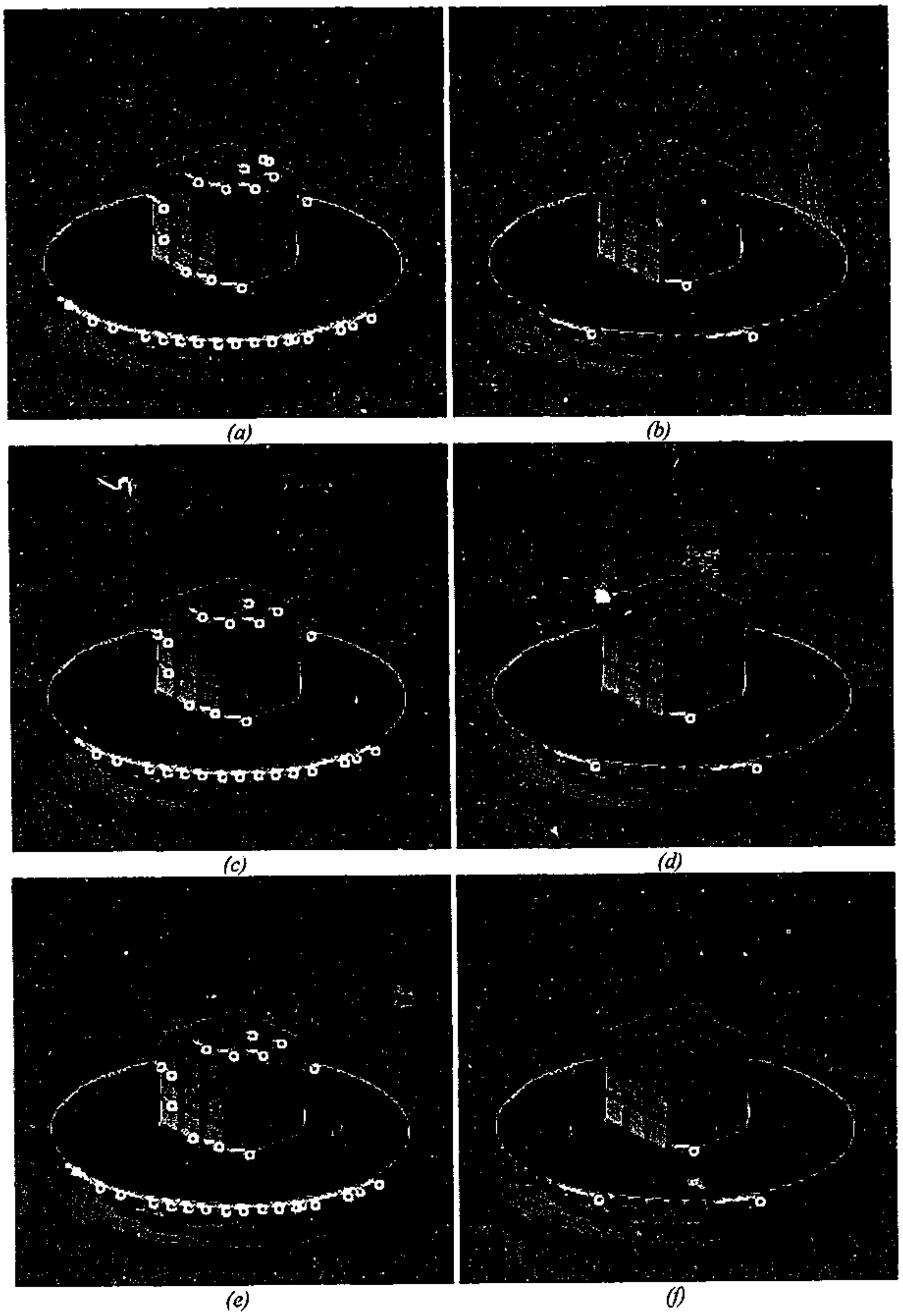


Figure 4.9: Road sequence track results (displayed on frame-1 for tracks which survived for length more than 20). The circle indicates the end of track: (a) Tracking performance obtained by using a constant acceleration model (M1). (b) Selected 3 tracks from (a) for illustration. (c) Tracking performance using a constant velocity model (M2). (d) The 3 tracks from (c), for the same features as shown in (b). (e) Tracking performance obtained by using a constant turn model (M3). (f) Selected 3 tracks from (e).

Tracks Obtained Using Different Motion Models (Rubic)



(a) (b) (c) (d) (e) (f)

Figure 4.10. Rubic sequence track results (displayed on frame-1 for tracks which survived for length more than 5). The circle indicates the end of track: (a) Tracking performance obtained by using a constant acceleration model (M1). (b) Selected 3 tracks from (a) for illustration. (c) Tracking performance using a constant velocity model (M2). (d) The 3 tracks from (c), for the same features as shown in (b). (e) Tracking performance obtained by using a constant position model (M5). (f) The best 3 tracks from (e).

MHT-MMF Tracker Result

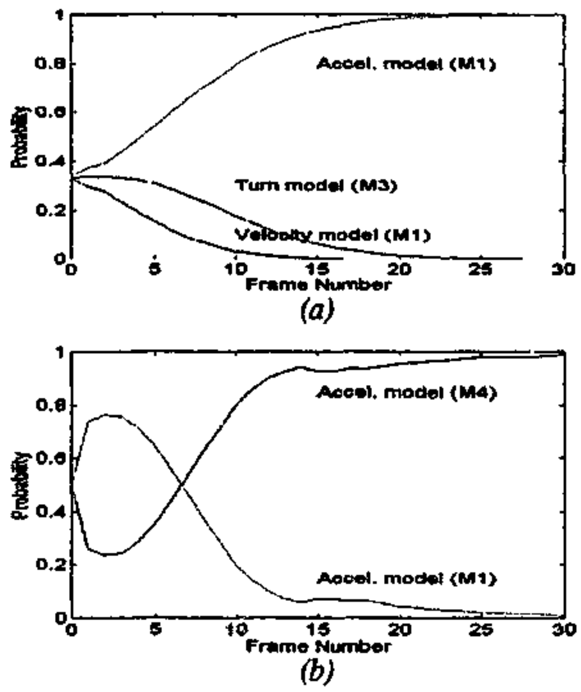


Figure 4.11: PUMA sequence using the MHT-MMF tracker. (a) M1,M2 and M3 (all 2nd order models) competing for the correct motion model. The constant acceleration model (M1) is chosen as the most appropriate model, as confirmed by Fig. 4.6a. (b) M4 (3rd order acceleration model) preferred over M1.

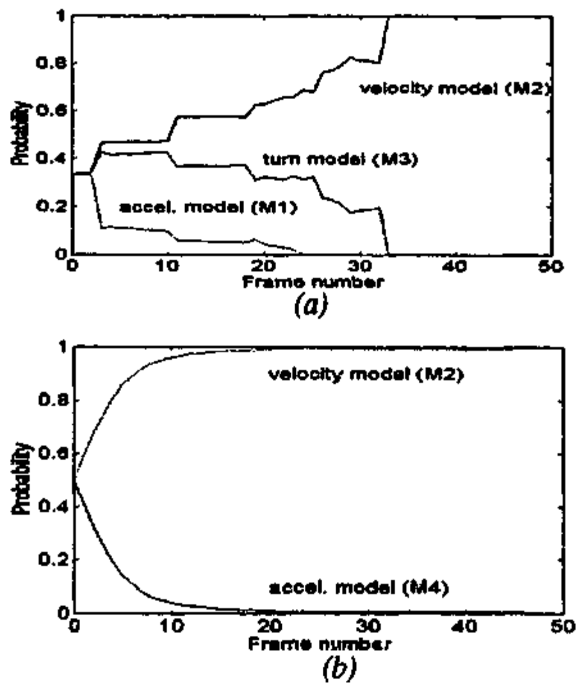


Figure 4.13: Walking man sequence using the MHT-MMF tracker. (a) M1,M2 and M3 (all 2nd order models) competing for the correct motion model. The constant velocity model (M2) is chosen as the most appropriate model, as confirmed by Fig. 4.8c. (b) M2 preferred over M4 (3rd order acceleration model).

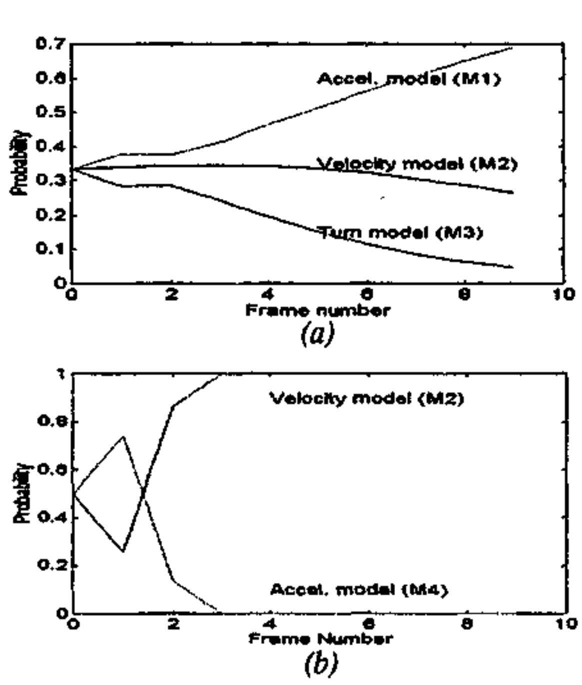


Figure 4.12: Toy car sequence using the MHT-MMF tracker. (a) M1, M2 and M3 competing for the correct motion model. The constant acceleration model (M1) is chosen as the most appropriate model, as confirmed by Fig. 4.7a. (b) M2 preferred over M4 (3rd order acceleration model).

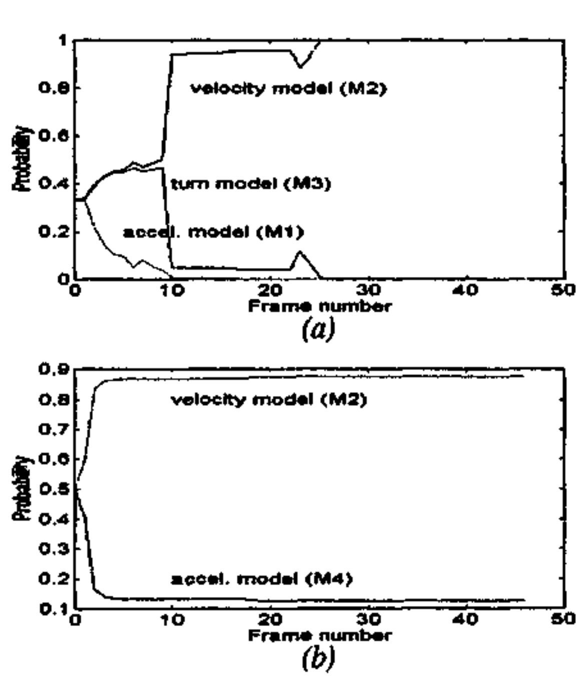


Figure 4.14: Road sequence using the MHT-MMF tracker (a) M1, M2 and M3 (all 2nd order models) competing for the correct motion model. The constant velocity model (M2) is chosen as the most appropriate model, as confirmed by Fig. 4.9c. (b) M2 preferred over M4 (3rd order acceleration model).

MHT-IMM Tracker Results

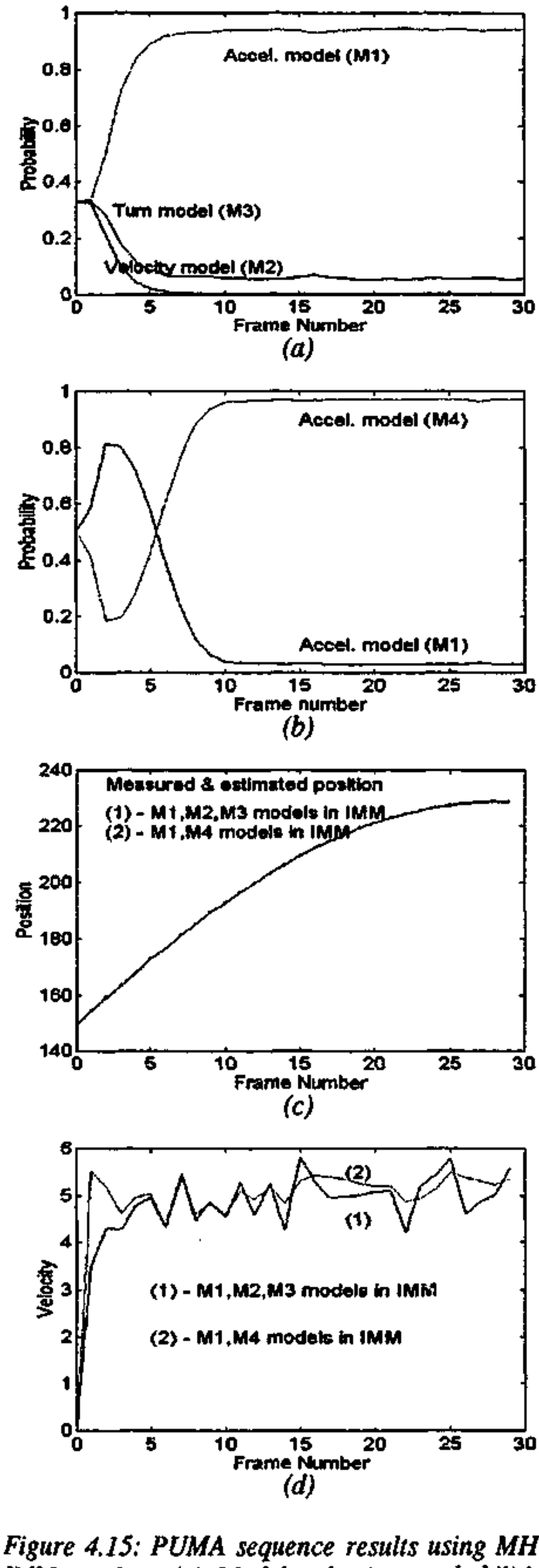


Figure 4.15: PUMA sequence results using MHT-IMM tracker. (a) Model selection probabilities: constant acceleration model (M1) selected as the correct motion model (three 2nd order models compared). (b) 3rd and a 2nd order model compared. M4 preferred over M1 as the correct motion model. (c) Measured and estimated position (d) Estimated velocity.

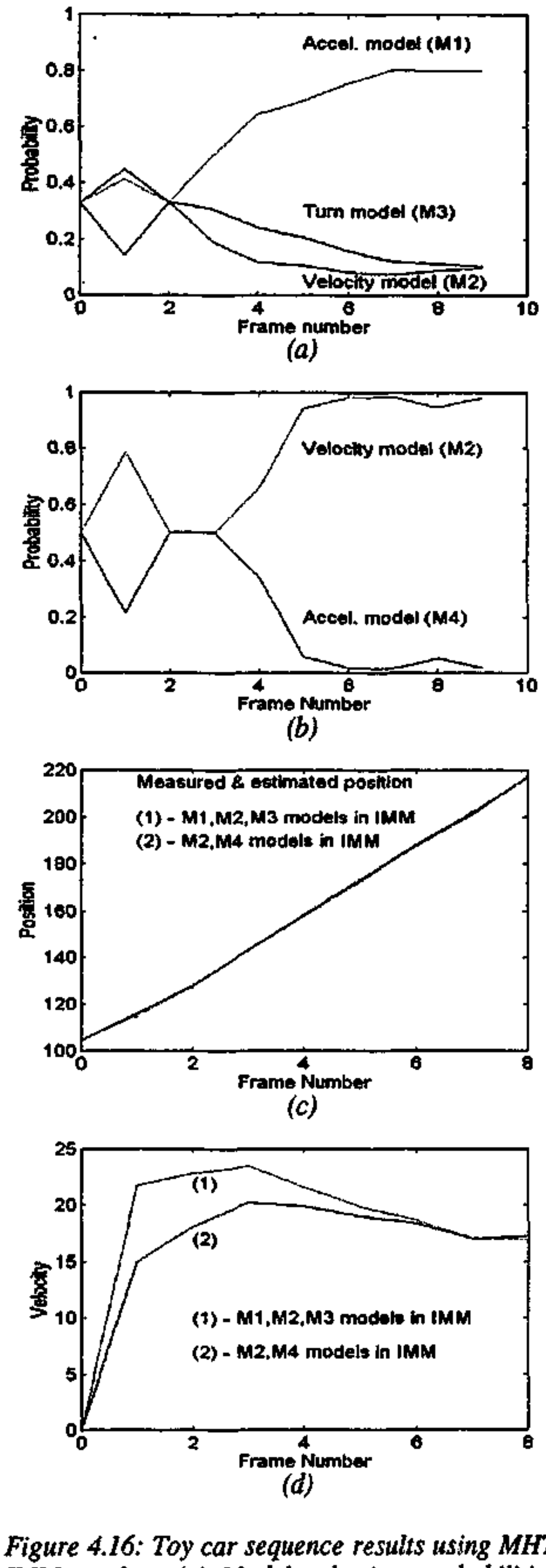
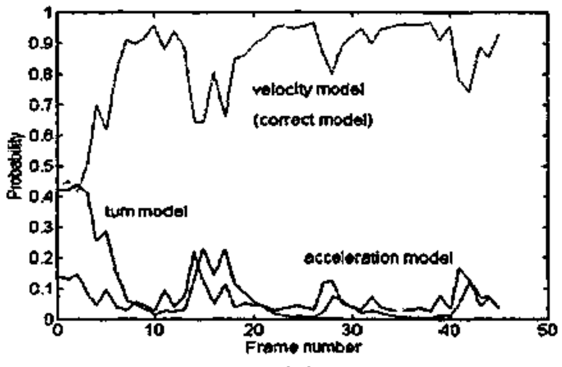
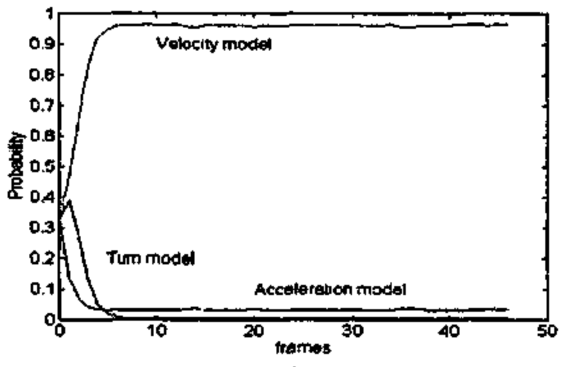


Figure 4.16: Toy car sequence results using MHT-IMM tracker. (a) Model selection probabilities: constant acceleration model (M1) selected as the correct motion model (all 2nd order models compared). (b) 3rd and a 2nd order model compared. M2 preferred over M4 as the correct motion model. (c) Measured and estimated position (d) Estimated velocity.

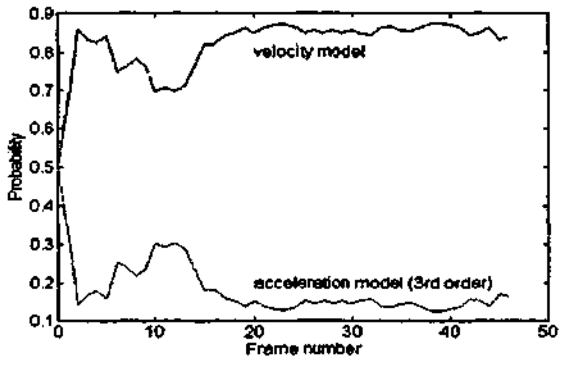
MHT-IMM Tracker Results (Cont...)



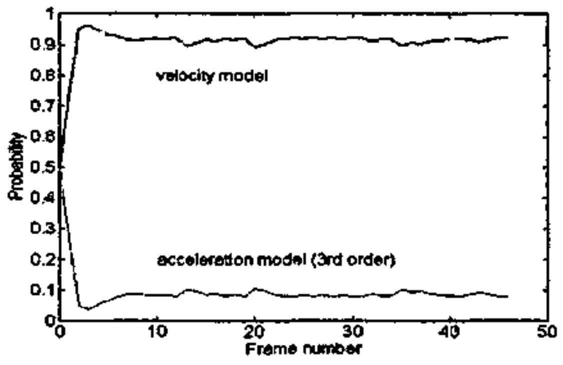
(a)



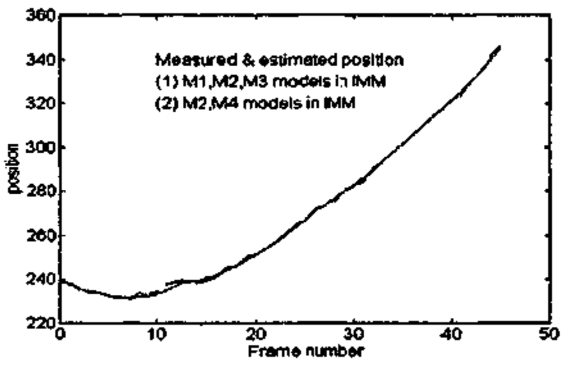
(a)



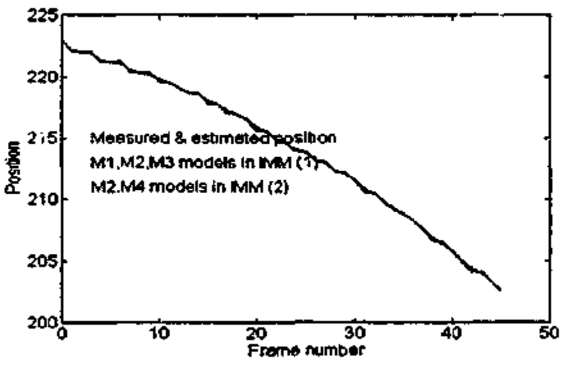
(b)



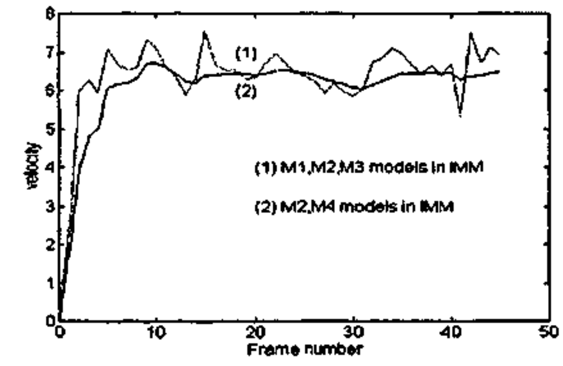
(b)



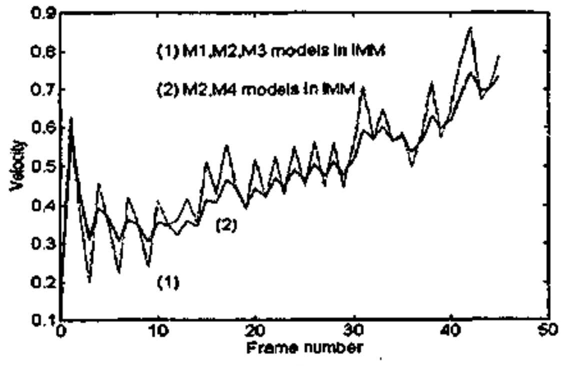
(c)



(c)



(d)



(d)

*Figure 4.17: Walking man results using MHT-IMM tracker. (a) Model selection probabilities: Constant velocity model (M2) selected as the **correct** motion model (M1, M2, M3 compared). (b) M2 preferred over M4. (c) Measured and estimated position of the feature considered. (d) Velocity estimates for the feature considered.*

*Figure 4.18: Road sequence results using MHT-IMM tracker. (a) Model selection probabilities: Constant velocity model (M2) selected as the **correct** motion model (M1, M2, M3 compared). (b) M2 preferred over M4. (c) Measured and estimated position of the feature considered. (d) Velocity estimates for the feature considered*

MHT-IMM Tracker Results (Cont...)

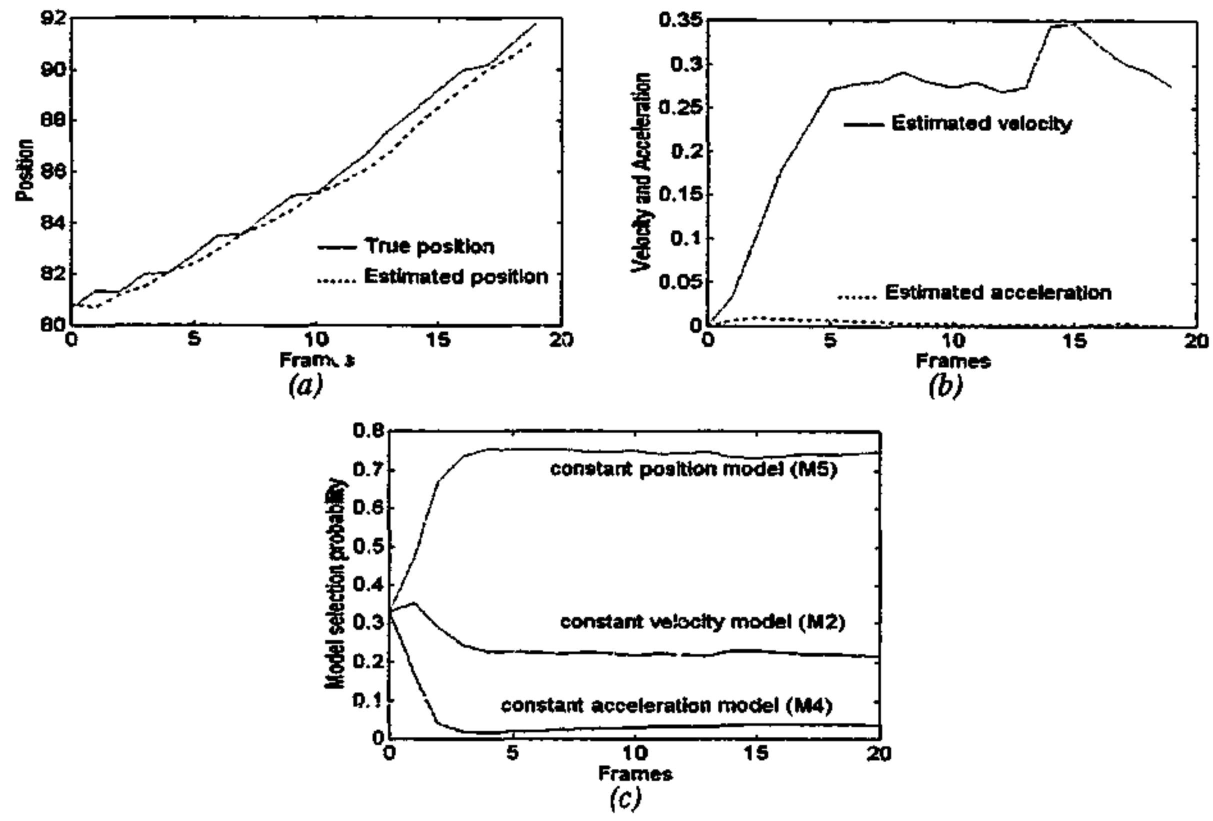


Figure 4.19: Rubic sequence results using MHT-IMM. (a) True and estimated position of a feature trajectory. (b) Estimated velocity and acceleration of the feature. (c) Model selection probability of the tracker. The constant position model is selected as the most appropriate motion model. Since the inter-frame displacement is very small, the results confirmed our expectation (see Fig. 4.10).

Note: The following tables provide the MHT-IMM tracking performance statistics. The error is measured between the measured (true) and the estimated position (x and y direction) and the velocity is obtained from the tracker filter estimate. The mean value (in pixels) is calculated by taking the average over the given image sequence length. It should be noted that the MHT-MMF tracker also gave good tracking results, provided the tracker was tuned properly.

Filters in bank for the IMM algorithm	mean absolute error (x)	mean absolute error (y)	MSE (Position, in pixels)	mean velocity (x-dir.)	mean velocity (y-dir.)	mean velocity (mag.)
Track 1						
with models M1, M2, M3	0.1016	0.1247	0.0419	3.7236	2.2390	4.6532
with models M1, M4	0.2932	0.1111	0.1741	4.2015	2.2791	5.0553
Track 2						
with models M1, M2, M3	0.2668	0.1940	0.1783	4.0444	9.3946	10.5599
with models M1, M4	0.2933	0.2060	0.2074	4.5252	10.0700	11.3737
Track 3						
with models M1, M2, M3	0.1527	0.1691	0.0785	5.9186	2.5892	6.7550
with models M1, M4	0.2938	0.1172	0.1834	6.5815	2.4579	7.2827

Table 4.1: Track performance statistics for the PUMA sequence for 3 selected tracks

Filters in bank for the IMM algorithm	mean absolute error (x)	mean absolute error (y)	MSE (Position, in pixels)	mean velocity (x-dir.)	mean velocity (y-dir.)	mean velocity (mag.)
Track 1						
with models M1, M2, M3	0.5552	0.7289	2.1916	14.5615	5.3467	15.5272
with models M2, M4	0.6916	0.2902	0.9942	18.2450	2.3869	18.4034
Track 2						
with models M1, M2, M3	0.7816	0.6824	1.8746	13.6980	3.6053	14.1712
with models M2, M4	0.3815	0.2470	0.3658	16.9096	0.5008	16.9192
Track 3						
with models M1, M2, M3	0.6539	0.7693	1.6643	11.7574	0.9493	11.8101
with models M2, M4	0.3347	0.1145	0.2214	13.9348	2.0419	14.0841

Table 4.2: Track performance statistics for the Toy car sequence for 3 selected tracks.

Filters in bank for the IMM algorithm	mean absolute error (x)	mean absolute error (y)	MSE (Position, in pixels)	mean velocity (x-dir.)	mean velocity (y-dir.)	mean velocity (mag.)
Track 1						
with models M1, M2, M3	0.3086	0.3965	0.5478	6.1275	1.4169	6.3280
with models M2, M4	0.8379	0.8034	1.2948	5.8074	1.5451	6.3272
Track 2						
with models M1, M2, M3	0.9425	1.1464	1.6424	5.6698	1.0798	5.8016
with models M2, M4	0.9556	1.2767	1.7500	5.7341	1.2152	5.9075
Track 3						
with models M1, M2, M3	1.2888	1.0865	1.9419	5.4369	1.2369	5.6206
with models M2, M4	1.3531	1.1846	2.0859	5.5535	1.3905	5.8081

Table 4.3: Track performance statistics for the Walking man sequence for 3 selected tracks.

Filters in bank for the IMM algorithm	mean absolute error (x)	mean absolute error (y)	MSE (Position, in pixels)	mean velocity (x-dir.)	mean velocity (y-dir.)	mean velocity (mag.)
Track 1						
with models M1, M2, M3	0.0764	0.1586	0.1952	0.0620	0.4856	0.4994
with models M2, M4	0.1229	0.1967	0.2559	0.0535	0.4729	0.4791
Track 2						
with models M1, M2, M3	0.0112	0.4994	0.5011	0.0076	0.9189	0.9201
with models M2, M4	0.0000	0.5159	0.5159	0.0000	0.9665	0.9665
Track 3						
with models M1, M2, M3	0.4671	0.7326	0.9127	0.5535	0.6720	0.8906
with models M2, M4	0.4952	0.8103	0.9906	0.5638	0.6902	0.9229

Table 4.4: Track performance statistics for the Road sequence for 3 selected tracks.

Filters in bank for the IMM algorithm	mean absolute error (x)	mean absolute error (y)	MSE (Position, in pixels)	mean velocity (x-dir.)	mean velocity (y-dir.)	mean velocity (mag.)
Track 1						
with models M2, M4, M5	1.16	0.71	1.36	0.23	-0.07	0.24
Track 2						
with models M2, M4, M5	0.55	0.01	0.5517	-0.0025	0.0816	0.0817
Track 3						
with models M2, M4, M5	1.40	0.33	1.43	0.27	0.03	0.2775

Table 4.5: Track performance statistics for the Rubic sequence for 3 selected tracks.

4.9 MHT-IMM versus MHT-MMF Tracker

4.9.1 A Simulated Example of a Feature Moving with Multiple Motions

In this section we provide a crucial test of the two trackers discussed. To do so, we simulated a trajectory of a feature as shown by Fig. (4.20) (the feature path is along A -> B -> C -> D -> E) consisting of 30 frames. The feature moves with a constant velocity for the first 10 frames (A -> B) then a manoeuvre occurs with a constant turn from frames 11 to 18 (B -> C), then the feature travels with a constant acceleration from frames 19 - 25 (C -> D), and finally travels with a constant velocity for the last 5 frames (D -> E). Both the trackers were applied to track the feature motion, and the results for the model selection probabilities are shown in Figs. (4.21)-(4.22). Fig. (4.21) shows the model switching ability of the MHT-MMF tracker. It can be observed that the constant velocity model (M2) operates until frame 11 (when the first model switching occurs), then we can see a dip in the model selection probability for M2, and the constant turn model (M3) takes over for a short period (about 3 frames), thereafter the *incorrect* model M2 is selected giving the *incorrect* feature motion. The reason for incorrect model selection is because M1 never 'revives' despite modifications made to the algorithm as discussed in section 4.5 and section 4.8.2.1. The model switching problem is rectified by using the MHT-IMM tracker. Fig. 4.22 shows the MHT-IMM model selection probability result. It can be clearly seen that the model switching occurs correctly almost at the appropriate time giving the desired result.

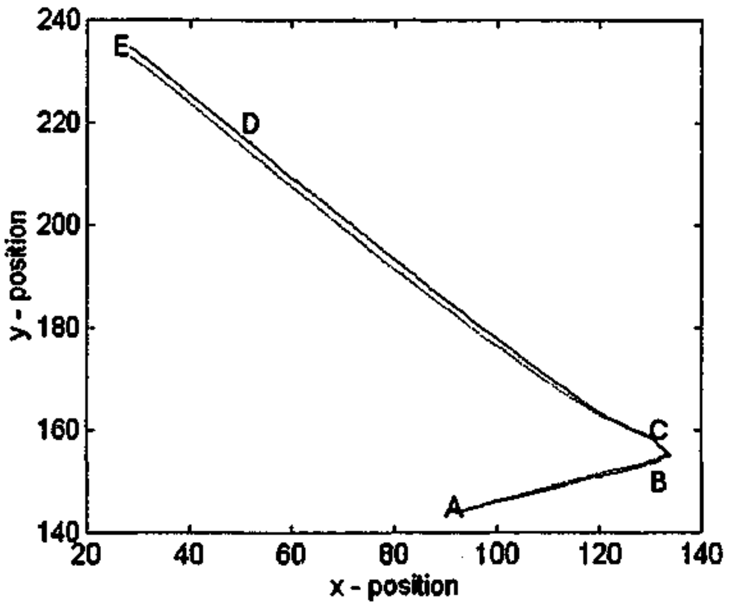


Figure 4.20: A simulated trajectory of a feature which travels with varied motion. The feature travels with constant velocity from A -> B, with a constant turn from B -> C, with constant acceleration from C -> D, and finally with constant velocity from D -> E. The continuous line (red) shows the true trajectory while the dashed line (green) is the estimated trajectory obtained by the MHT-IMM tracker.

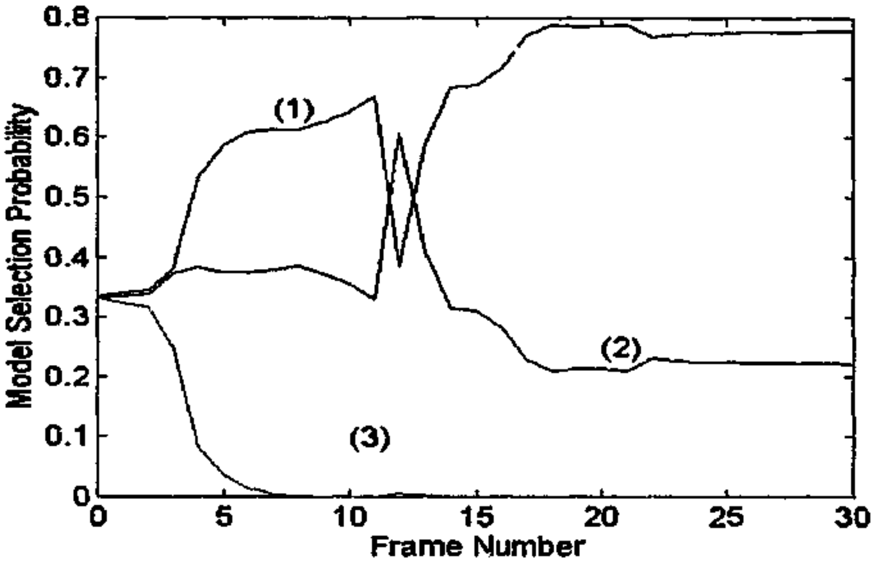


Figure 4.21: The motion model selection probability versus frame number for the MHT-MMF tracker when tracking the feature in Fig. 4.20. The models used are: (1) - M2 (constant velocity model), (2) - M3 (constant turn model), (3) - M1 (constant acceleration model). It can be observed that after the first model switching (around frame 12 at position B) the MMF frame work fails to switch motion models at positions C and D.

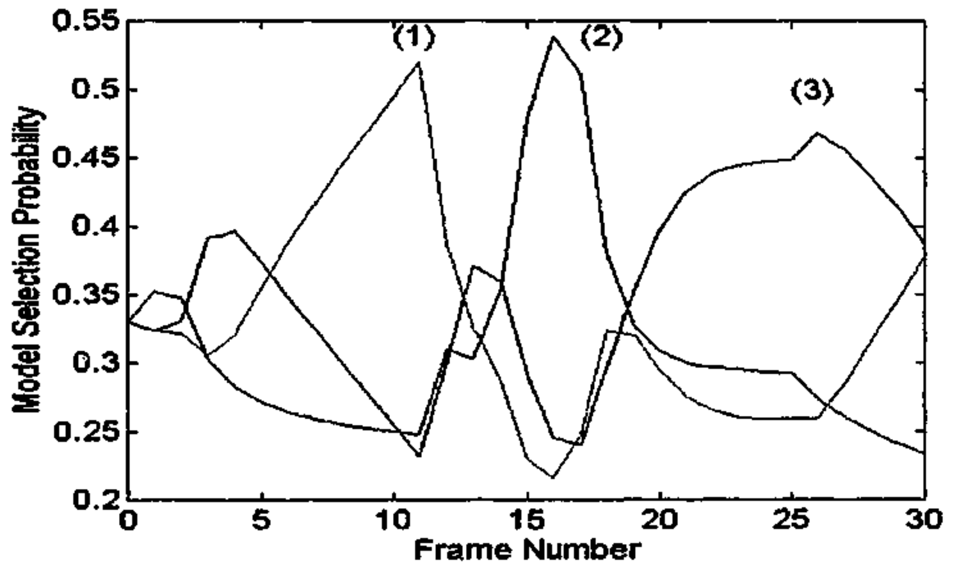
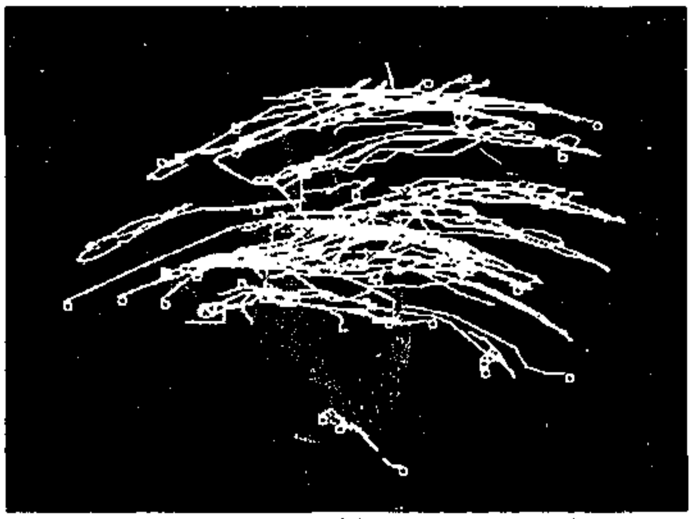


Figure 4.22: The model selection probability versus frame number for the MHT-IMM tracker. The same motion models are used as for the MHT-MMF case. The model switching occurs as expected at around frame 13 (position B), around 18 (position C) and around 26 (position D). The tracker correctly switches motion model, and tracks the feature as shown by the trajectory (green lines) in Figure 4.20.

4.9.2 A Real Example of a Feature Moving with Multiple Motions

A waving hand sequence was captured at 15 frames/sec. The motion of the hand was variable to test the agility of the MHT-IMM tracker. The hand starts to move from a stationary position towards the right. It picks up speed rapidly and then comes to a halt. It then moves again towards the left gaining speed, then slows down and comes to a rest. The process is repeated 2 times. The motion models that were employed for the MHT-IMM algorithm included a constant acceleration model (M1), a constant velocity model (M2) and a constant position model (M5). The last model assumes that the current position of the feature is the best estimate for the next time step, such a model is useful to describe a feature at rest (More of this model is discussed in the next chapter).

Figures 4.23(a) shows the qualitative results of the trajectories obtained. A total of 25 corners were tracked to keep the tracking complexity to a minimum. The corners were extracted using the KLT detector [190]. Figure 4.23(b) shows 3 of the trajectories to clearly show the path of the hand movement. A feature that was successfully tracked for a considerable length of time was further examined to verify the model selection changes. Figure 4.24(a) shows the trajectory of the feature of interest (true and estimated positions of the *x* and *y* coordinates). Figs. 4.24(b),(c) show the estimated velocity and acceleration respectively. From Fig. 4.24(d) it can be seen that model switching occurs around the 13th, 20th, 24th, 30th, 40th, 45th, 53rd, 60th and 70th frames. This result closely corresponds to the actual motion changes of the hand (actual motion was manually assessed).


(a)

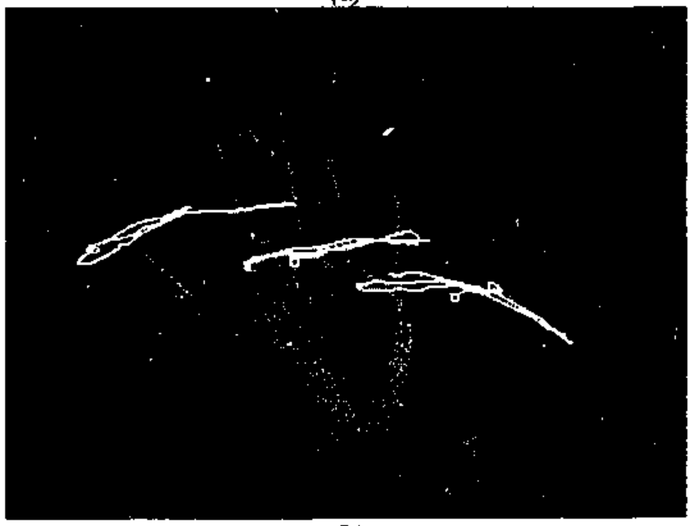
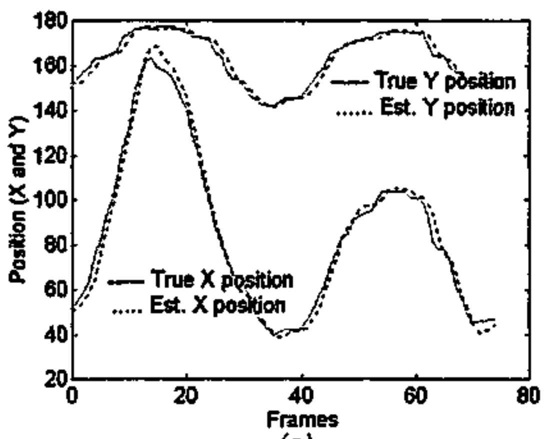
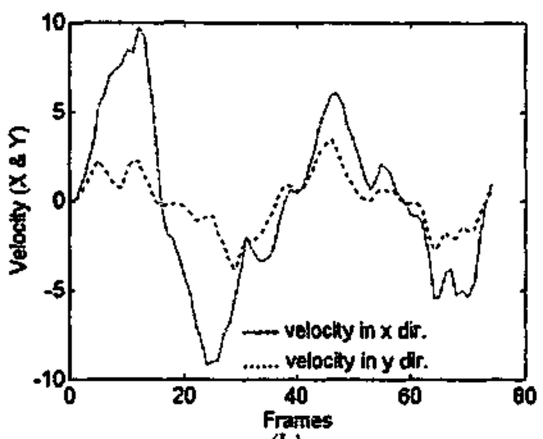

(b)

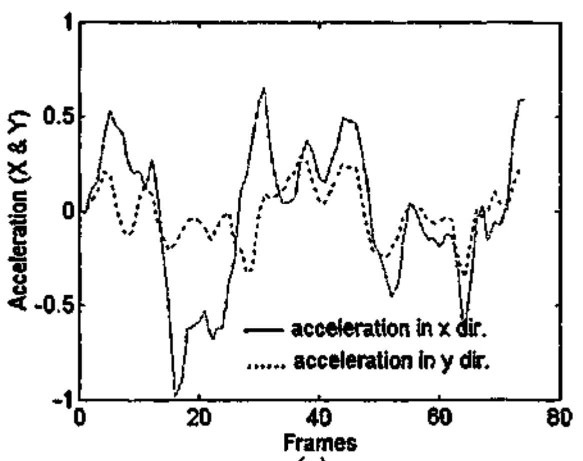
Figure 4.23: The qualitative results of the trajectories obtained for the waving hand sequence. (a) Displays trajectories which survived more than 10 frames. The small circle indicates the end of track. (b) 3 tracks displayed to illustrate the movement of the hand clearly (these 3 tracks survived for more than 50 frames).



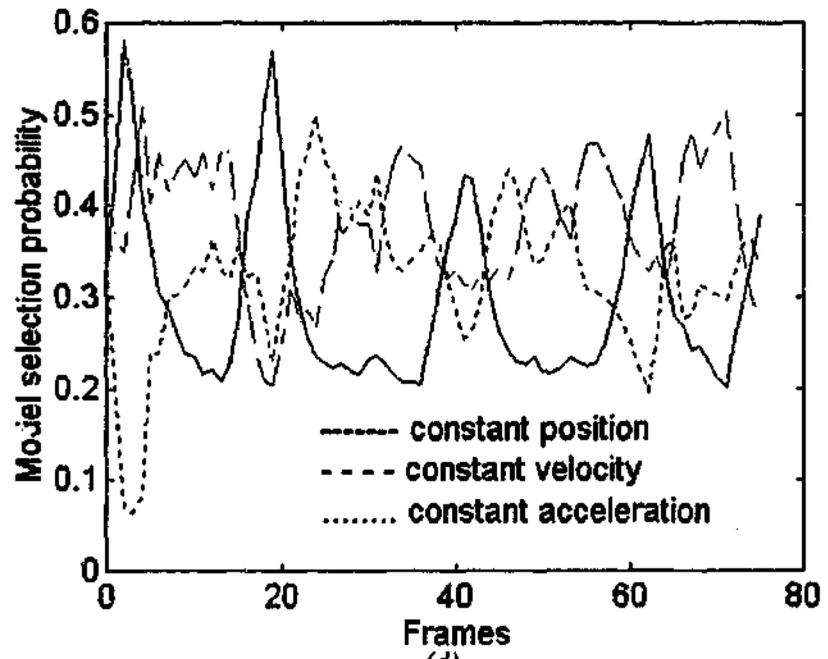
(a)



(b)



(c)



(d)

Figure 4.24: Tracking result for the waving hand sequence. (a) The true and estimated position of the corner feature of interest. (b) The estimated velocity of the feature. (c) Estimated acceleration of the feature. (d) The probabilty of the motion model/s in operation during the course of tracking. As expected the MHT-IMM switches motion models automatically according to the featur's motion. Note the constant postion model (M5) in operation periodically, which corresponds to the change of hand direction (20th, 40th and 60th frames).

4.10 Discussion

In this section we discuss some issues that need to be taken care of when implementing the multiple motion model filtering algorithm. Some limitations of the multiple model filtering framework are also discussed. Most of the issues discussed are applicable to the *MHT-MMF* algorithm, but some aspects mentioned need to be considered when implementing the *MHT-IMM* algorithm.

4.10.1 Model Selection for Two Models of Similar Type

When there are *only 2* models in a bank of filters, and if they are of '*similar type*', that is they have very comparable residuals (ν), then there is a possibility that the multiple model filtering (the *MHT-MMF* tracker) method might fail [130, 157]. In this particular case, we can't effectively use the multiple model algorithms in the conventional way (as discussed in section 4.5). A modified version of the tracker is required to discriminate between the most appropriate (*correct*) and the unlikely (*incorrect*) models.

For example, consider the case where only M1 (2nd order near constant acceleration model) and M2 (2nd order constant velocity model) are the 2 models to exist in a bank of filters. If the multiple model tracker is applied to track a feature that move with a motion described by model M1, then there is a chance that M2 will be selected as the correct motion model despite the actual motion being constant acceleration (M1). Why ?

Maybeck [130] states that one would expect that the residuals, ν_j (for the *j*-th model), of the Kalman filter based upon the *correct* model (most appropriate model) will be consistently smaller than the residuals of the other mismatched (*incorrect*) filters, which, will cause the *correct* probability to increase, while causing the others to decrease. The performance of the multiple model algorithm is dependant upon a <u>significant</u> difference between the residual characteristics [130] of the *correct* and the *incorrect* filter models, and that if the residuals instead are consistently of the same magnitude, then the algorithm results in the growth of the probability, $\mu_j(k)$ associated with the filter with the smallest value of $|S_j|$. Lund et al. [124] explain the effect as follows:

When the system model, *M*, equals the *i*-th model M_i (i.e. $M = M_i$), one would expect that the exponential term in Eq. (4.5) would be lower for the *correct* filter (the *i*-th filter in this case). i.e.

$$r_i(k) << r_j(k), \qquad \forall \quad j \neq i \tag{4.15}$$

where $r_j(k)$ is defined as:

$$r_j(k) = \frac{1}{2} v_j^T(k) S_j^{-1}(k) v_j(k) \tag{4.16}$$

Hence, as $\mu_i(k)$ increases towards unity, the probabilities of the mismatched filters will decrease towards zero if the condition described by equation (4.15) persists over several measurements. If however, the system model does not equal any of the r models that form the multiple model algorithm and /or the filters are tuned improperly it is possible that:

$$r_1(k) \approx r_2(k) \approx \cdots \approx r_N(k) \tag{4.17}$$

Then $\mu_j(k)$ is now governed by $|S_j(k)|$, $j=1,\ldots,r$ and $\mu_j(k)$ increases if $|S_i(k)|$ is less than $|S_j(k)|$, $i \neq j$, while $|S_j(k)|$, $j \neq i$ decreases. For Kalman filters and Extended Kalman filters, $|S_j(k)|$ is not dependent on which model is correct, and erroneous decisions upon the correct model may result. The situation described by equation (4.17) is undesirable.

Focussing on the example of having M1 and M2 motion models in a bank (both have identical state vectors, and state transition matrices, but have different process noise matrices), one would expect the magnitude of residuals for both models to be identical. In this case M2 will always give a smaller value of $|S_j(k)|$, because it employs a lower value of process noise (Q). The effect of this is the same as a lower order filter being selected while the other is a higher order filter (i.e., when both their residuals are comparable). When the magnitude of the residuals are similar, the lowest order filter will always have the smallest value of $|S_j(k)|$ as discussed in [157]. Hence, the *MMF* algorithm will *choose* M2 as the most correctly modelled filter irrespective of the true motion. The probability that a model is correct is therefore determined solely by the covariance S_j when the magnitudes of the residuals are similar.

Robert et al. [157] in their study of multiple model filtering (using *MMF* type algorithms) for vehicle tracking have shown that the value of residual, v, has very little effect on the likelihood function value Λ (in equation (4.5)). In contrast, the value of the estimated residual covariance, S has a large effect on Λ. It must be concluded that the *MMF* algorithm does not discriminate effectively when presented with motion models of residual values that are small, or of similar magnitude, and separate filters of different dimension, and (or) have different states. In the next 2 sub-sections we consider methods which will aid the multiple model algorithm (*MMF*) to resolve the correct motion model when presented with 2 similar types of motion models.

4.10.2 Modelling the System Noise (Q)

Lund et al. [124] addressed the problem in Section 4.10.1 by using a method called Inter-Residual Distance Feedback (IRDF), where the filters are modified on-line in such a way as to de-tune them through the modulation of one of the filter parameters. The modulation is governed by a scalar quantity calculated from a distance measure between residuals. The main principle of the method is to keep an inter-residual distance measure above a specified limit by adjusting the filter gains. This is achieved by varying the system noise covariance, Q_j (for the j-th model). In the filter equation, Q_j is simply replaced by a modulated system noise covariance matrix Q'_j, where Q'_j is defined as:

$$Q'_j(k) = \eta(k) Q_j, \qquad j = 1,2 \tag{4.18}$$

where $\eta(k)$ is the modulating variable [124, 157]. It must be noted that the number of filters is restricted to two when using the IRDF method as stated in [124]. A method allowing more filters to be used is also outlined which consists of considering only the inter-residual distance between the 'two most probable models'. However, it must be remembered that the motivation behind the IRDF algorithm is to overcome the problem of unreliable probability values, $\mu_j(k)$. A method relying on $\mu_j(k)$ values to discriminate between the most probable models therefore seems unwise.

For the vehicle tracking project, Robert et al. [157] had used the above method along with the empirically generated covariance of the residuals $\hat{S}(k)$ (proposed my Mayback [130] for tuning Q) which is defined as,

$$\hat{S}(k) = \frac{1}{N} \sum_{n=k-N+1}^{k} v_n v_n^T \tag{4.19}$$

where N is the most recent time step. Robert et al. [157] describe the empirically generated covariance as more reliable than using the theoretical $S(k)$. Their simulations suggest that using this method selected the correct motion model, which the traditional *MMF* method failed to do.

In the example discussed in section 4.10.1 (where M1, M2 are used in a filter bank), we cannot tune Q, since Q is used to distinguish between a velocity and a near acceleration model, and is kept constant. Therefore, we need another method to overcome this problem. A possible method is to tune the measurement noise matrix R, which is discussed in the following section.

4.10.3 Modelling the Measurement Noise (R)

In the following section we introduce a *tunable* measurement noise in the Kalman filter which can discriminate between similar models provided the initial condition for R is a reasonable estimate to the actual measurement noise.

Maybeck [130] states that if R or Q is to be estimated separately, a reasonable solution is usually achievable. It is generally true, for all algorithms and not just those based upon the maximum likelihood concept, that the R parameter estimates are more precise than the Q parameter estimates. Since Q is fixed in our case, we have to tune R, which needs to be updated adaptively. Maybeck [130] proposed a possible tuning method for R as given below:

$$\underbrace{R(k) = \frac{1}{N}\left[\sum_{j=k-N+1}^{k} v_j v_j^T\right] - H(k)P(k)H^T(k)}_{\textit{Maybeck's tuning method}}, \quad \text{for } k \geq N \tag{4.20}$$

The estimation process is essentially time-invariant over the most recent N steps, ie, $S(k)^{-1}$ remains almost a constant over these steps, provided Q is known completely (as in the case of the example).

For the experiments given in this chapter, where there were only 2 motion models, the above modelling methods have been followed, so that the final result reflected the true motion of the feature considered. Further examples of 2 model filtering process can be found in [187].

4.10.4 Limitations of the Multiple Model Filter

A Kalman filter's performance is extremely sensitive to initialisation. Good initial state estimates will ensure fast convergence, while poor estimates give rise to slow convergence, sometimes even filter divergence. Both Kalman filters and EKFs are prone to divergence. That is, although it is an optimal filter, there are practical limitations to Kalman filters that may lead to its divergence [130]. Three types of divergence exist:

1) True divergence - due to unbounded system modelling errors, which lead to some elements of the state covariance matrix, P, increasing without limit. This is the most severe divergence since errors become unbounded very quickly.
2) Apparent divergence - due to the mis-modelling of plant excitation, measurement noise variances, and the effects of system biases. Here, a steady state is reached but the associated errors are too large to allow the estimates to be useful.
3) Numerical divergence - due to filter computation round-off errors and finite precision arithmetic.

Both true and apparent divergences were observed during our simulations. Apparent divergence tends to manifest itself as a constant bias in the estimates (see [187] for examples). In this case the state covariance matrix has to be initialised (when a divergence is detected) in order for correct tracking to occur (particularly true for the *MHT-MMF* tracker). The explanation for this phenomenon is that the calculated covariance matrix becomes unrealistically small, so that undue confidence is placed in the estimates and subsequent measurements are effectively ignored. The phenomenon of apparent divergence is critical to the operation of the multiple model algorithm. It is this effect that provides *MMF* the mechanism with which to choose the most appropriate filter from its component filters. Apparent divergence shows itself when the residuals have a non-zero mean (a correctly converged filter will have near zero-mean residuals). It is therefore true divergence that adversely effects multiple model filtering performance (for MMF type filters).

True divergence leads to error magnitudes that become unbounded very quickly. It is therefore important to detect the occurrence of true divergence as quickly as possible. In this algorithm implementation, true divergence is detected by analysing the values of the leading diagonal of the state covariance matrix P. When the value of any one of these elements exceed a pre-set threshold the filter is said to have truly diverged and it is re-started and re-initialised at that time. The estimate produced at the point of detected true divergence is therefore the observed estimate. The effects of re-starting the diverged filter is to set its probability, $\mu_j(k)$, to zero (this is achieved automatically using the standard *MMF* equations because the covariance of the residuals for that filter will be zero). If the divergent filter is the *correct* filter, there will be a significant time delay before its estimates converge again and hence a lag before it is recognised as the correct filter. The problem of poor initialisation is hence an issue after true divergence.

However most of these limitations caused by *MMF* based algorithms are alleviated by using the *IMM* algorithm. This is because of the interaction and mixing between each filter at the start of the algorithm recursion (see Fig. 4.4 and Eq. 4.9 and 4.10), which is not the case with the *MMF* algorithm. It also worth noting that the likelihood functions for both these algorithms are quite different (compare Eq. 4.5 and Eq. 4.11). Despite IMM's slight computational cost over MMF, the advantages of IMM is far more than the MMF algorithm (see Table 4.6).

Multiple Model Filter Type	*MMF*	*IMM*
Number of filters	r	r
Number of combinations of r estimates and covariances	1	$r + 1$
Number of probability calculations	r	$r^2 + r$

Table 4.6: Comparison of complexities of the multiple model algorithms

4.11 Conclusion

Our study has shown how the Multiple Hypothesis Tracking (MHT) technique combined with an Interacting Multiple Model (IMM) algorithm can discriminate between different motions described by an image sequence. The results have provided evidence of our method being able to identify different motions while maintaining good tracking results. With the increasing power and availability of parallel machines, the parallel nature of the MHT-IMM algorithm provides an attractive solution for many real time visual processing applications. The tracking technique presented can also be used to segment objects moving with varied motion into separate groups.

A difficulty with the MHT-MMF tracker is to tune the Kalman filters at the initialisation stage. Since KFs (and hence the MMFs) are sensitive to initial conditions, reasonable initial parameters need to be provided in order for the tracker to perform well. Bad or improper initial condition can lead to filter divergence, or even converge to the wrong motion model [130]. Another drawback of the MHT-MMF algorithm is the ad-hoc modifications required for the base algorithm in order to cope with model switching tracking applications, which demands extra computational cost. However, by employing the IMM algorithm most of the limitations caused by MMF are alleviated to a great extent.

All tracking systems have limitations. This system is no exception. In any image sequence, if one wants to track several objects, each feature on each object needs to have a separate tracking algorithm in order to identify the motion correctly. This will be computationally expensive. The other drawback is that the features need to be extracted independently of the MHT. A coupled feature detection and tracking mechanism, perhaps along the lines of Zheng and Chellappa [207], Shapiro *et al* [169] or Kang *et al* [112] is worth investigating. In such a coupled system one could use information as a feedback between tracking and feature extraction to improve the performance of the latter.

Chapter 5

Performance Prediction Analysis of a Point Feature Tracker Based on Different Motion Models

Abstract

This chapter provides performance prediction analysis techniques for a linear point feature tracking algorithm based on different motion models. We provide closed form expressions to evaluate the probability of correct data association of a tracker (analysed with different motion models), when tracking under clutter. We also extend our analysis for the prediction of correct data association when a tracker recovers from a false match to regain correct tracking. The simple mathematical expressions provided here, can be used to implement performance analysis procedures that are fast, easy, and are reasonably accurate (compared with conventional computationally expensive Monte-Carlo tracking experiments employed to predict the performance of a tracker). We have also demonstrated the importance of using a correct motion model for a visual tracker to get optimum tracking performance, based on empirical evaluation techniques. The performance of a tracker's robustness under varied noise has also been investigated.

5.1 Introduction

For the last two decades the target tracking community has been focussing on the performance of various target tracking algorithms [5, 6, 41, 42, 43, 125, 126, 135, 136, 138, 139, 158]. In most cases the applications of these algorithms are for specific purposes (mostly defence oriented), such as tracking missiles and satellites, to analyse aircraft manoeuvres, space-craft trajectory analysis etc. Our survey shows that in the area of image processing and pattern recognition there are very few published papers which provide performance analysis techniques for tracking algorithms for computer vision related applications. The relatively small amount of performance analysis work reported in the literature, relating to visual tracking, are (in most cases) for a narrow band of applications. For example, analysing the tracking performance of a walking person [11], tracking of the left ventricle [27], evaluation of vehicle tracking [118], [3], tracking of faces [24], [52] and body motions [25], medical diagnostics [24] etc. Most of these evaluation techniques presented are for non-point feature tracking algorithms.

Work reported in [196] uses more generalised comparison techniques to compare 4 point feature trackers. The performance of the trackers are compared only on the basis of the speed of the tracking algorithm in relation to the number of points tracked using a cost function strategy. In [138], [139] Ngan et. al. presented a more versatile tracker performance prediction measure called the Probability of Correct Association (PCA) for 2 trackers (a zero velocity and a constant acceleration tracker) when tracking in clutter. They also introduced the PCA concept for a tracker when recovering from a False Match (referred to as PCA-FM in this chapter), but they only considered the case for the zero velocity tracker (also referred to as the constant position tracker). In this paper we review their work (for completeness) and we then extend the performance prediction measures PCA and PCA-FM to a tracker based on the 3 different motion models (a constant acceleration, a constant velocity and a constant position (zero velocity) model). Thus we provide the extra 3 important performance measures *missing* in Ngan et. al.'s work (ie. PCA for a constant velocity model, and PCA-FM for a constant velocity and constant acceleration model) to complete the performance prediction study. In addition, we demonstrate that the theoretical closed form performance measures are a credible representation for track results obtained by independent Monte-Carlo simulations, using real dynamic image sequences. We also empirically evaluate the performance of a complete feature tracker, the Multiple Hypothesis Tracker - MHT [58] (as discussed in chapter 4) using the different motion models considered to emphasise the importance of choosing the correct motion model for optimal tracker performance (with supporting results). We have also compared MHT's performance with a non-prediction based tracker (the KLT tracker discussed in Chapter 3) to assess the effectiveness of the prediction scheme (based on the different motion models) in the presence of varied noise.

Since our primary task is to compare the performance of a point feature tracker with different motion models, the corner features (each occupying 1 pixel in the image plane) that we track are extracted independently in each frame of a given image sequence, by using the KLT corner detector [190] (It is also possible to manually label corner points of interest in each frame for this analysis). By doing so we totally isolate the tracking procedure from the corner extraction procedure purely to focus on the performance of the prediction and tracking process.

The performance of a tracker is evaluated at different clutter density levels. This is achieved by artificially inserting clutter points at different densities around the actual corner features extracted (within a specified area centred at each feature). This process is employed to see whether a tracker based on a particular motion model is robust enough to associate the predicted feature point with the actual feature. Each experiment is carried out at a different clutter density level to evaluate the tracker performance.

5.1.1 Tracking Performance

An important property of any type of tracker is its performance in the presence of clutter. A tracker would ideally always choose the actual target point feature over a clutter point feature at the data association step. In practice differences between the modelled target motion and the actual target motion compromise the effectiveness of the estimation step, and the random distribution of clutter points leads to a non-zero probability of false data associations occurring. The issue of false data association is of particular importance in tracking systems which choose only one data point to continue the object trajectory (such as the nearest neighbour method), because the unselected true data point is discarded from contention and is never considered in future data associations. Therefore, the analysis presented in this chapter will also consider the probability of a tracker to regain track of the moving object (corner feature) at the next step if at the current step a false association has occurred.

The second important factor emphasised in this chapter is motion model selection for optimum tracking performance. Most tracking algorithms use a single motion model in its framework mainly because of computational advantages [58, 11]. Such an assumption is valid provided the tracked feature of interest moves with a similar motion to that of the motion model. If the motion of the feature is different, or if the feature changes motion during the course of tracking; then the tracker fails to provide the best quality trajectories. We therefore, have included an empirical evaluation of a real point feature tracker (the Multiple Hypothesis Tracker - MHT [58]), tracking with varied motion models (the motion models as studied in chapter 4). The results presented emphasise the importance of motion model selection for optimal feature trajectories.

This chapter is organised as follows: Section 5.2 provides the assumptions that are required for the analysis. Section 5.3 provides the performance measures used. In section 5.4 and 5.5 the derivation of expressions for the probability of correct data association (PCA and PCA-FM) for the three trackers are considered. Section 5.6 outlines the experimental evaluation procedure employed. Section 5.7 provides the results and discussion, and finally Section 5.8 gives the conclusion.

5.1.2 Nomenclature

$\mathbf{p}_k$	True position of object at time *k*
$\mathbf{v}_k$	Velocity of object at time *k*
$\mathbf{a}_k$	Acceleration of object at time *k*
η_k	Gaussian distributed random variable at time *k*
$p(\eta_k)$	Probability density function of η_k
P_1	Probability that a pixel is a clutter point
P_0	Probability that a pixel is not a clutter point
$P_{ZVT}\{k+1\}$	Probability of correct association at time *k+1* for a Zero Velocity Tracker (*ZVT*)
$P_{CVT}\{k+1\}$	Probability of correct association at time *k+1* for a Constant Velocity Tracker (*CVT*)
$P_{CAT}\{k+1\}$	Probability of correct association at time *k+1* for a Constant Acceleration Tracker (*CAT*)
$P'_{ZVT}\{k+1\}$	Probability of correct association at time *k+1* for a ZVT, when a false match occurred at time *k*
$P'_{CVT}\{k+1\}$	Probability of correct association at time *k+1* for a CVT, when a false match occurred at time *k*
$P'_{CAT}\{k+1\}$	Probability of correct association at time *k+1* for a CAT, when a false match occurred at time *k*

5.2 Assumptions for Tracking Analysis

The initial objective of this chapter is to develop closed form performance prediction techniques for a tracker (based on 3 linear motion models: a Zero Velocity Tracker (ZVT), a Constant Velocity Tracker (CVT) and a Constant Acceleration Tracker (CAT)).

(1) **Zero Velocity Tracker (ZVT)**: This tracker assumes that the object position at any point in time originated from a Wiener process and so it's positional increment from one time instant to the next is independent of all preceding position increments (based on a constant position motion model). Under such circumstances, since the next positional increment could be equally in any direction, the best estimate of the object at time step *k* is by default its last observed position (given by Eq. 5.1). Such a prediction scheme would yield perfect predictions if the object remained in its last observed position, and therefore it can also be considered a zero velocity predictor.

(2) **Constant Velocity Tracker (CVT)**: This tracker assumes that the target object moves with constant velocity. Velocity is defined as the vector difference between two successive position vectors, as described by Eq. 5.2.

(3) **Constant Acceleration Tracker (CAT)**: This tracker assumes that the target object moves with constant acceleration. Acceleration is defined as the vector difference (a finite difference approximation) between two successive velocity vectors (Eq. 5.3).

The formulation of closed form expressions for the probability of correct association (PCA) for each tracker requires a number of assumptions to be made, which are listed as follows.

Assumption 1: Only a single moving corner point is considered at a time, and the selected corner is assumed present in every frame (this is verified by a manual check following corner detection). The motion of the point feature behaves according to the following dynamic motion equations (for ZVT, CVT, and CAT respectively)

$$\mathbf{p}_k = \mathbf{p}_{k-1} + \eta_k \tag{5.1}$$

$$\begin{aligned} \mathbf{p}_k &= \mathbf{p}_{k-1} + \mathbf{v}_{k-1} + \eta_k \\ &= 2\mathbf{p}_{k-1} - \mathbf{p}_{k-2} + \eta_k \end{aligned} \tag{5.2}$$

$$\begin{aligned} \mathbf{p}_k &= \mathbf{p}_{k-1} + \mathbf{v}_{k-1} + \mathbf{a}_{k-1} + \eta_k \\ &= 3\mathbf{p}_{k-1} - 3\mathbf{p}_{k-2} + \mathbf{p}_{k-3} + \eta_k \end{aligned} \tag{5.3}$$

where $\mathbf{p}_k, \mathbf{v}_k (= \mathbf{p}_k - \mathbf{p}_{k-1}), \mathbf{a}_k (= \mathbf{v}_k - \mathbf{v}_{k-1})$ are the position, velocity and acceleration at time k respectively (*per unit time*) and η_k is a Gaussian distributed noise.

Assumption 2: Data association is performed by the nearest neighbour method. No drift is assumed in the prediction phase when estimating the position of a feature.

Assumption 3: Clutter is present in every frame. A new set of clutter points is generated for each frame, and the clutter points are uniformly distributed in two-dimensional space for performance evaluation.

5.3 Performance Measures for Tracking

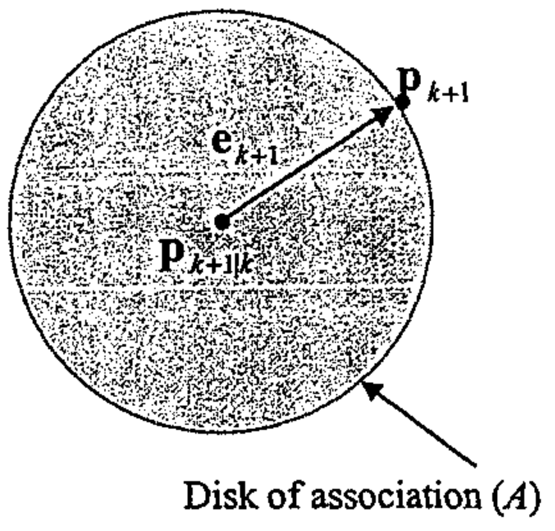


Figure 5.1: Definition of prediction error.

A natural choice for a measurement of tracking performance is the *track purity* of the output trajectory generated by the tracking system. This is the average percentage of correctly associated measurements in each track, see Chang et. al. [41]-[43]. However, an analytical expression for *track purity* is very difficult to derive. An alternative measurement is called the *Probability of Correct Association (PCA)* [138], which as its name suggests is the probability, at any given step, that the tracking system will make a correct data association in the presence of clutter.
The following *PCA*' s are presented (as performance prediction measures) in this chapter.

1. The probability of obtaining the correct association at $t = k+1$ for all three trackers given that a *correct* association has been made at the *previous* time steps (referred as PCA).
2. The probability of making a correct association at $t = k+1$ for all three trackers given that an *incorrect* association has been made at $t = k$, but that a *correct* association has been made at *previous* time steps (referred as PCA-FM).

5.3.1 Nearest Neighbour Data Association

Denote the probability that a given pixel is not a clutter point by P_0 (and its complement by P_1), the true position of the object at $t = k+1$ by $\mathbf{p}_{k+1}$, and the predicted position of the object at $t = k+1'$ given track positions up to and including $t = k$ by $\mathbf{p}_{k+1|k}$. Then the prediction error is defined by [138]. See Fig. (5.1).

$$\mathbf{e}_{k+1} = \mathbf{p}_{k+1} - \mathbf{p}_{k+1|k} \qquad (5.4)$$

An overhead tilde is used to denote values derived from an incorrectly associated measurement made at $t = k$, thus giving rise to, $\tilde{\mathbf{p}}_{k+1|k}$, and $\tilde{\mathbf{e}}_{k+1}$.

A correct association is attained when no clutter point occurs within the region of association, which is defined to be a disk with radius $\mathbf{e}_{k+1}$ centred at $\mathbf{p}_{k+1|k}$, as shown in Fig. (5.1). The probability of the event of correct association is equal to the probability that no clutter point exists within a radius of $\mathbf{e}_{k+1}$ of $\mathbf{p}_{k+1|k}$. This probability is given by the probability P_0 to the power of the number of pixels in a disk of radius $\mathbf{e}_{k+1}$ [138], namely,

$$P\{correct \quad association \quad at \quad time \quad k+1\} = P_0^{\pi|\mathbf{e}_{k+1}|^2} \qquad (5.5)$$

Thus the derivation for the probability of correct association for ZVT, CVT, CAT begins with the determination of an appropriate expression for $\mathbf{e}_{k+1}$ for each case.

5.4 Probability of Correct Data Association (PCA)

This section describes the method to obtain an expression for the probability of correct association (at time step *k+1*) for each of the trackers (ZVT, CVT and CAT) assuming the tracker has not made a false match, up to the current time step *k*.

5.4.1 Derivation of PCA for the Zero Velocity Tracker (ZVT)

This tracker assumes that the best prediction for the point feature in the next frame is the current point in the trajectory. Therefore, the following relationship holds for the ZVT (can also be termed the constant position tracker).

$$\mathbf{e}_{k+1} = \mathbf{v}_{k+1} \qquad (5.6)$$

Substituting Eq. (5.6) into Eq. (5.5) yields the expression for the probability of correct association at time step *k*+1 (denoted as $P_{ZVT}\{k+1\}$) in terms of the velocity from $\mathbf{p}_k$ to $\mathbf{p}_{k+1}$. In this case the error is caused by a small velocity $\mathbf{v}_{k+1}$ (For the ZVT, it is assumed that $\mathbf{v}_{k+1} >> \eta_k$. Therefore, the random noise component is neglected). If this assumption is taken into account, then the PCA for ZVT is given by

$$P_{ZVT}\{k+1 \mid \mathbf{v}_{k+1}\} = P_0^{\pi\|\mathbf{v}_{k+1}\|^2} \tag{5.7}$$

5.4.2 Derivation of PCA for the Constant Velocity Tracker (CVT)

The constant velocity tracker assumes correct data associations have been made at $t = k$, and $t = k$-1. The reason is that the CVT requires past positions of the feature at time k and k-1 to predict the position at k+1. If any one of these past 2 positions represent a clutter point, the calculated velocity value will not reflect the true velocity of the feature at $t = k+1$.

Using the error definition and the dynamic equations for constant velocity (section 5.2), we have,

$$\begin{aligned} \mathbf{e}_{k+1} &= \mathbf{p}_{k+1} - \mathbf{p}_{k+1|k} \\ &= \mathbf{p}_{k+1} - (\mathbf{p}_k + \mathbf{v}_k + \eta_k) \\ &= \mathbf{v}_{k+1} - \mathbf{v}_k + \eta_k = \mathbf{a}_{k+1} + \eta_k \end{aligned} \tag{5.8}$$

Assume that for a non ideal case, $\mathbf{a}_{k+1}$ is non zero (say a small constant acceleration error component $\mathbf{a}_{k+1}$=a is present) and η_k is a sample from a Gaussian distributed noise. Therefore, using Equations (5.5) and (5.8) the probability of correct association for CVT (denoted as $P_{CVT}\{k+1\}$) can be given as follows:

$$P_{CVT}\{k+1 \mid \eta_k, \mathbf{a}_k\} = P_0^{\pi\|\mathbf{e}_{k+1}\|^2} = P_0^{\pi\|\eta_k + \mathbf{a}_k\|^2} \tag{5.9}$$

Since CVT is more accurate than ZVT in terms of prediction, η_k is assumed non-negligible compared with the acceleration error term. Now using the total probability theorem [5], for the CVT, we obtain,

$$P_{CVT}\{k+1 \mid \mathbf{a}\} = \int_{-\infty}^{+\infty} p_{CVT}\{k+1 \mid \eta_k, \mathbf{a}\}.p(\eta_k).d\eta_k \tag{5.10}$$

where $p(\eta_k)$ is the pdf of the Gaussian distributed random variable η_k.

Expanding equation (5.10) with suitable substitution, and integrating out, produces the following expression (see Appendix C for proof):

$$P_{CIT}\{k+1 \mid \mathbf{a}\} = I_x . I_y \tag{5.11}$$

where $I_x = \frac{e^{\gamma_x}}{4\sqrt{2\pi}\sigma_x \alpha_x^{3/2}}\left[4\alpha_x\sqrt{\pi} + 4\sqrt{\alpha_x}\beta_x + \beta_x^2\sqrt{\pi}\right]$

with $\alpha_x = -\left(\ln P_0\pi - \frac{1}{2\sigma_x^2}\right)$, $\beta_x = 2a_x \ln P_0\pi$, $\gamma_x = a_x^2 \ln P_0\pi$. σ_x, a_x are the Gaussian noise variance and acceleration (a) component in the x direction respectively. A similar expression can also be obtained for I_y.

5.4.3 Derivation of PCA for the Constant Acceleration Tracker (CAT)

Recall from section 3 that the constant acceleration tracker assumes correct data associations have been made at $t = k, k\text{-}1, k\text{-}2$. The reason for this is that the acceleration term is calculated using the previous three positions of the trajectory. If any one of these three position terms represents a clutter point, the calculated acceleration value will not reflect the true acceleration of the feature point at $t = k+1$.

The prediction error for the constant acceleration tracker is as follows (using similar approach as before).

$$\begin{aligned} \mathbf{e}_{k+1} &= \mathbf{p}_{k+1} - \mathbf{p}_{k+1|k} \\ &= \mathbf{p}_{k+1} - (\mathbf{p}_k + \mathbf{v}_k + \mathbf{a}_k + \eta_k) \\ &= \mathbf{v}_{k+1} - (\mathbf{v}_k + \mathbf{a}_k + \eta_k) = \mathbf{a}_{k+1} - (\mathbf{a}_k + \eta_k) \\ &\approx \eta_k \end{aligned} \tag{5.12}$$

Under the constant acceleration assumption, $\mathbf{a}_{k+1} \approx \mathbf{a}_k$ *(higher order motion terms are assumed negligible compared with the noise term, therefore not considered in (5.12))*. Using equation (5.12), the probability of correct association for CAT (denoted as $P_{CAT}\{k+1\}$) is given by;

$$P_{CAT}\{k+1 \mid \eta_k\} = P_0^{\pi|\eta_k|^2} \tag{5.13}$$

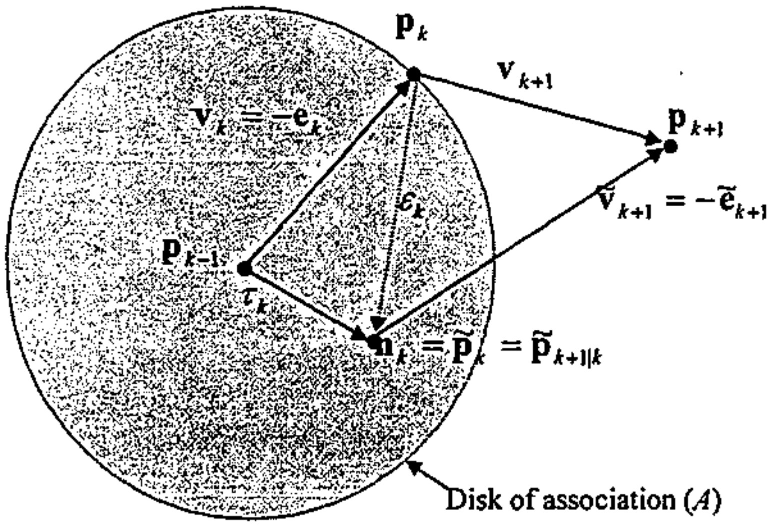
The probability of correct association for CAT can be derived by setting **a** = 0 in Equations (5.9) - (5.11), and further simplifying the expression reduces to (see Appendix C for proof),

$$P_{CAT}\{k+1\} = \frac{1}{\sqrt{\left(1-2\pi\sigma_x^2 \ln P_0\right)\left(1-2\pi\sigma_y^2 \ln P_0\right)}} \tag{5.14}$$

Alternatively, the expression (5.14) can be obtained by expanding (5.13) and then simplifying using the fact that the area under a density function is unity (see Appendix C for a detail derivation).

5.5 Probability of Correct Association for Recovering from a False Match (PCA-FM)

This section describes the method to obtain an expression for the probability of correct association for each of the trackers (ZVT, CVT and CAT) when they recover from a mismatch to regain correct track (after a false match occurring in the previous time step *k*). An analytical derivation for ZVT is provided, but for CVT and CAT, PCA-FM expressions are provided using a combination of analytical derivation and Monte-Carlo experiments.



$\mathbf{p}_k$
$\mathbf{v}_{k+1}$
$\mathbf{p}_{k+1}$
$\mathbf{v}_k = -\mathbf{e}_k$
ε_k
$\widetilde{\mathbf{v}}_{k+1} = -\widetilde{\mathbf{e}}_{k+1}$
$\mathbf{p}_{k-1}$
τ_k
$\mathbf{n}_k = \widetilde{\mathbf{p}}_k = \widetilde{\mathbf{p}}_{k+1|k}$
Disk of association (A)

Figure 5.2: Prediction error $\mathbf{v}_{k+1}$ includes the offset due to data association error ε_k.

5.5.1 Derivation of PCA-FM for the Zero Velocity Tracker (ZVT)

If a clutter point occurs within the disk shown in Fig. (5.2) at $t = k$, then that clutter point would be selected for the trajectory at $t = k$. The probability of correct association when recovering from a false match (PCA-FM), denoted as $P'_{ZVT}\{k+1\}$, is a measure of the likelihood that the ZVT will perform a correct data association at $t = k+1$ given that an incorrect association had occurred at $t = k$. This situation is illustrated in Fig. (5.2).

Assume the existence of a clutter point $\mathbf{n}_k$ inside the disk of association at time $t = k$. The nearest neighbour criterion would associate $\mathbf{n}_k$ with the trajectory at $t = k$, which is designated the symbol $\tilde{\mathbf{p}}_k$, where the tilde denotes an incorrect association. When applying the zero velocity prediction scheme, the incorrectly associated point also becomes the new prediction point, ie. $\tilde{\mathbf{p}}_{k+1|k} = \tilde{\mathbf{p}}_k$, and thus leading to a prediction error given by the following equation.

$$\tilde{\mathbf{e}}_{k+1} = \mathbf{p}_{k+1} - \tilde{\mathbf{p}}_{k+1|k} = \tilde{\mathbf{v}}_{k+1} \tag{5.15}$$

From Fig. (5.2) it can be seen that $\tilde{\mathbf{v}}_{k+1}$ is the vector difference of the true velocity $\mathbf{v}_{k+1}$ and the two-dimensional random variable ε_k as given by equation (5.16).

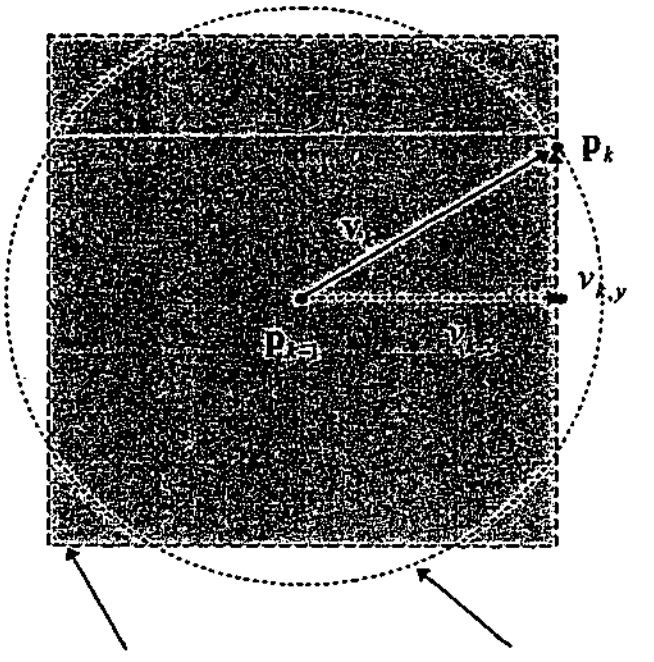
$$\tilde{\mathbf{v}}_{k+1} = \mathbf{v}_{k+1} - \varepsilon_k \tag{5.16}$$

Substituting the prediction error into equation (5.5) yields the probability of correct association given as follows.

$$P'_{ZVT}\{k+1 \mid \mathbf{v}_{k+1}, \varepsilon_k\} = P_0^{\pi|\tilde{\mathbf{v}}_{k+1}|} = P_0^{\pi|\mathbf{v}_{k+1}-\varepsilon_k|^2} \tag{5.17}$$

The probability $P'_{ZVT}\{k+1 \mid \mathbf{v}_{k+1}\}$ can be formed from equation (5.17) by integrating out the random term ε_k using the total probability theorem. The probability density function of ε_k is a uniform distribution inside the disk of association A, and zero outside. That is, $p(\varepsilon_k) = 1/\pi\|\mathbf{v}_k\|^2$ if ε_k is inside A, and 0 otherwise [139].

For mathematical convenience, the disk of association (A) is approximated by a square (S) as shown in Fig. (5.3), which is centred at $\mathbf{p}_{k-1}$ and has sides of length $2v_{\max,k}$, where $v_{\max,k}$ is defined as: $v_{\max} = \max\left(|v_x|,|v_y|\right)$.



Square region of integration (S)   Disk of association (A)

Figure 5.3: Approximating the circular region of integration A by a square region S.

With suitable substitution and integration, the final expression can be given by (see Appendix C for proof):

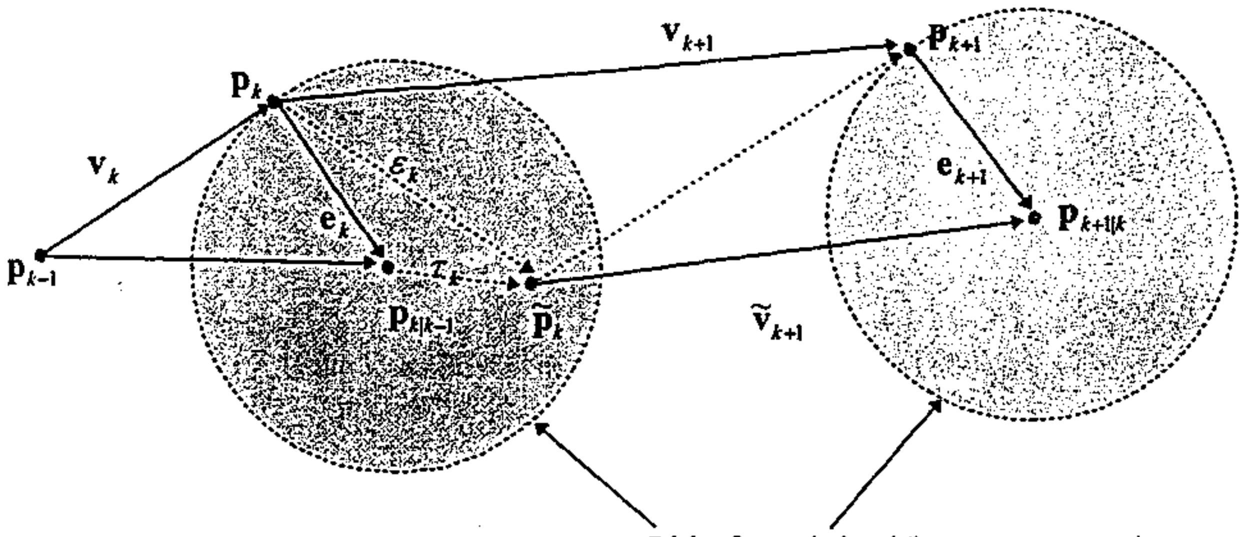
$$P'_{ZVT}\{k+1\,|\,\mathbf{v}\} = \frac{1}{(2v_{\max})^2} P_0^{\pi|2\mathbf{v}|^2} I_x I_y \tag{5.18}$$

where, $I_x = \dfrac{\sqrt{\pi}}{2\sqrt{-a}}\exp\{-ab^2\}\left[erf\left(\sqrt{-a}(v_{\max}+b)\right)+erf\left(\sqrt{-a}(v_{\max}-b)\right)\right]$

with $a = \ln P_0\pi$ and $b = -2v_x$. A similar expression can also be found for I_y. Note that this expression is only valid for a small velocity $\mathbf{v}$ (to approximate a zero velocity tracker).

5.5.2 Derivation of PCA-FM for the Constant Acceleration Tracker (CAT)

A complete expression for the probability of correct association for a constant acceleration tracker recovering from a false match (prediction for $t = k+1$ given that at $t = k$ there was a mismatch) is rather more complicated than the zero velocity case. However, we provide a geometrical derivation using a similar vector diagram to Fig. (5.2). An expression can be found fo: a *variable* disk size (instead of a fixed disk size as in Fig. 5.2) as shown in Fig. 5.4. For the ZVT it is known that the radius of disk of association (A) is a constant velocity error ($\mathbf{e}_{k+1} = |\mathbf{v}_{k+1}|$). But for a CAT we model the radius r (with components r_x, r_y in the x, y direction) of A as a variable quantity.



Disk of association (A)

Figure 5.4: Vector diagram for the probability of correct association for CAT and CVT when recovering from a mismatch (the radius of the disk of association is modelled as a variable quantity).

From vector diagram (Fig. 5.4), and with the motion equation for CAT it can be shown (with assumptions (i), (ii) given below) that:

$$\mathbf{e}_{k+1} \approx (r + \tau_k)/2 \tag{5.19}$$

where $\mathbf{e}_k$ $(= r)$ is the radius of the disk of association (the error) for the matching at $t = k$.

Assumptions for deriving (5.19):

(i) τ_k is a random error term (non Gaussian) between the actual position of the feature and the selected mismatched feature (at $t = k$).

(ii) $\tau_k >> \eta_k$ (for low noise levels).

By a similar analysis as to that carried out for the ZVT, the probability of correct association for PCA-FM (recovering from a false match), denoted as $P'_{CAT}\{k+1 \mid r\}$, can be given by (see Appendix C for details):

$$P'_{CAT}\{k+1 \mid r\} = \frac{1}{4r_{\max}^2} P_0^{\frac{\pi}{4} r_{\max}^2} I_x I_y \tag{5.20}$$

where $2r_{\max}$ is the length of the square (whose area approximates that of the circle, as for the ZVT case) and $r_{\max} = \max\left(|r_x|, |r_y|\right)$. The I_x is given by,

$$I_x = \frac{\sqrt{\pi}}{2\sqrt{a}} \exp\left\{a r_x^{\,2}\right\}\left[erf\left(\sqrt{a}(r_{\max} + r_x)\right) + erf\left(\sqrt{a}(r_{\max} - r_x)\right)\right]$$

with $a = -\ln P_0 \frac{\pi}{4}$ and r_x is the x component of r, and can be given by the following expression.

$$\begin{aligned} r_x &= [\ \alpha + \beta \exp\left\{-(bh^2 + ch + d)\right\}\ ].h_x \\ &= f(h_x, h_y).h_x \end{aligned} \tag{5.21}$$

where h is an error quantity (a higher order motion, considered as an error term) of the dynamic system (h_x is the x component of h), and α, β, b, c, d are constants. The expression for r_x was evaluated by performing extensive Monte-Carlo simulations (a mathematical closed form expression for r_x is very difficult to derive). The following approximate values for the constants were also obtained by Monte-Carlo methods; $\alpha = 2, \beta = 28, b = 1.5, c = 2, d = 0.03$ for CAT. A similar expression to Eq. (5.21) can be given for I_y and r_y.

Equation (5.20) is very similar to equation (5.18) except that r is a *variable size* in this case. Through a series of Monte Carlo simulations, using a range of modelled error terms (such as the rate of change of acceleration) and disk radius values *(r)*, we were able to create close matches between the experimental probability of correct association (PCA-FM) and the theoretical expression given in equation (5.20). The result of these experiments is given in the plot in Fig. 5.5. This shows the variation of $f(h_x, h_y)$ with h. A perfect CAT is obtained when *h -> 0*. Conversely, with the use of this plot for a given error term (eg: a constant higher order motion term) we can find r (using equation 5.21), and in turn be able to find the probability of correct data association using Equation (5.20).

5.5.3 Derivation of PCA-FM for the Constant Velocity Tracker (CVT)

For the derivation of PCA-FM for CVT (denoted as $P'_{CVT}\{k+1|r\}$), a similar analysis can be carried out as for the CAT. In this case a constant velocity model is assumed with a constant acceleration error term added to the dynamic motion equation. The probability of correct association (recovering from a mismatch) is still given by equation (5.20), but r_x is now given by the following expression.

$$\begin{aligned} r_x &= [\ \alpha + \beta \exp\{-(bg+c)\}\ ].g_x \\ &= f(g_x, g_y).g_x \end{aligned} \tag{5.22}$$

where g is the acceleration (modelled error term) of the dynamic system (g_x is the x component of g), and α, β, b, c are constants (as before) and were found to be approximately: $\alpha = 2.2, \beta = 18, b = 2.5, c = 0.03$ through a series of simulations. The variation of $f(g_x, g_y)$ with g is shown in Fig. 5.6. As before a perfect CVT is obtained when g -> 0.

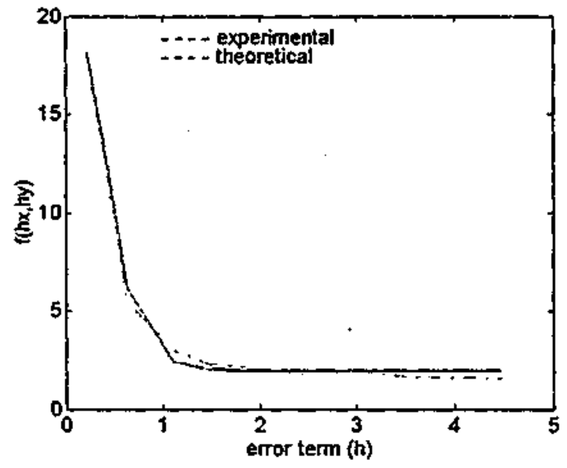


Figure 5.5

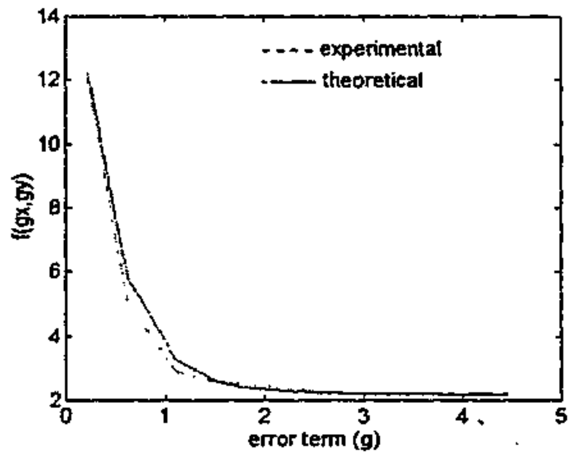


Figure 5.6

Figure 5.5: Variation of $f(h_x, h_y)$ with the error term (h). As h->0 a perfect CAT model is obtained.

Figure5. 6: Variation of $f(g_x, g_y)$ with the error term (g). As g->0, a perfect CVT model is obtained.

5.6 Experimental Procedure

This section describes the experimental procedure used to empirically validate the probabilities of correct association.

5.6.1 Simulation Method

In the tracking experiments, the probabilistic component relates to the generation of random corner points that clutter the scene in which tracking occurs. The steps involved in the Monte-Carlo experiments for the analysis are shown in the following pseudo code.

```
Generate reference trajectory (applying the tracker with no clutter
points)
 For each level of clutter
   For each N trials
    Create cluttered point set from reference trajectory.
    Apply tracking algorithm to noisy token set.
    Compare output trajectory with reference trajectory.
    End
   End
 End
Calculate probability of correct associations.
```

5.6.2 Generation of Reference Trajectory

Separate reference trajectories (trajectory of the corner feature) were created using the ZVT, CVT and CAT for each of the image sequences considered. The estimated (predicted and tracked) position of a feature point is generated by applying the dynamic equations given in section 5.2 for each tracker without considering clutter points. The position estimation step is followed by a manual check to verify the existence of a feature in every frame. Each of the image sequences considered (details given in section 5.7.1) had at least a small (non-zero) motion component due either to feature motion or camera motion. Even to evaluate the ZVT, a small motion is necessary, because the performance measures developed for the ZVT is velocity dependent (as given by the theoretical expression in section 4.1), and it makes sense to evaluate the measure at a given velocity (small value). The reference trajectory generated by using a CAT is based on the dynamic motion equation (5.3) with a random error component added (noise and a small constant higher order motion term). Similarly for the CVT tracker, the dynamic equation (5.2) is used with a random component, consisting of a noise term and a small constant acceleration.

5.6.3 Generation of Clutter Points

For each frame of a sequence, a set of random tokens is added to represent clutter. The random tokens are uniformly distributed in each frame using a specified clutter density P_1 (see section 5.3.1 for definition of P_1). The spatial region over which the new tokens are deposited is centred at the reference corner point and is a square region with a sufficient number of pixels so that the expected number of deposited (within the region) clutter tokens is *N*. In the experiments *N* was set to 10. The value of *N* cannot be set too high because the extent of the clutter region grows very large with small values of P_1. Having *N* large can be computationally impractical in the current implementation. The set of P_1 values used, were (0.0001 0.0002 0.0005 0.001 0.002 0.005 0.01 0.02 0.05 0.1 0.156 0.277 0.625). These are the 13 clutter levels at which the probability of correct association is evaluated.

5.6.4 Calculation of Probability of Correct Association

The probabilities of correct association (PCA and PCA-FM) for simulations are calculated using the following expressions for each tracker (where event types *A*, *B*, *C*, and *D* are defined below):

$$P_{ZIT}\{k+1;\mathrm{v}_{k+1}\}, P_{CIT}\{k+1\}, P_{CAT}\{k+1\} = \frac{A}{A+B}$$
$$P'_{ZIT}\{k+1;\mathrm{v}_{k+1}\}, P'_{CIT}\{k+1\}, P'_{CAT}\{k+1\} = \frac{C}{C+D}$$

Type *A*: Indicates correct tracking at time *k*

Type *B*: Incorrect tracking at time *k*

Type *C*: Indicates recovery from incorrect tracking at time *k*

Type *D*: Continuation of error due to incorrect tracking at time *k*

Table 5.1 gives the requirement for each tracker to obtain event types *A*, *B*, *C*, and *D*.

	ZVT				CVT				CAT			
Time	***A***	***B***	***C***	***D***	***A***	***B***	***C***	***D***	***A***	***B***	***C***	***D***
k	1	0	1	0	1	0	1	0	1	0	1	0
k-1	1	1	0	0	1	1	0	0	1	1	0	0
k-2	X	X	1	1	1	1	1	1	1	1	1	1
k-3	X	X	X	X	X	X	1	1	1	1	1	1
k-4	X	X	X	X	X	X	X	X	X	X	1	1

Table 5.1. Event types (A, B, C, D) for each tracker, where a '1' indicates correct matching at time k, and a '0' indicates incorrect matching at time k. An 'X' indicates a don't care.

5.7 Results

The following sub-sections describe the qualitative and quantitative results obtained for the various empirical evaluations presented. The simulated tracking performance is evaluated under 2 categories. The *track life* (Total number of correct trajectory points regardless of trajectory order) and *track purity* (Number of frames to first incorrect trajectory point). Our analysis as discussed below shows that the PCA and PCA-FM derived (for each separate motion model) are a good representation for the simulated *track life* and *track purity*, respectively, for each image sequence considered. All our experiments presented here are based on real life dynamic image sequences (some frames of each sequence are displayed in Appendix G). Finally, we employ a complete feature tracker (the Multiple Hypothesis Tracker [58] is chosen for demonstration) and show the quality of feature trajectories obtained using different motion models described. The MHT is also compared with the KLT tracker (a non-prediction based tracker) at varied noise levels to test the robustness of the tracker.

5.7.1 Track Life and Track Purity Results Under Clutter

The results presented in Figs. (5.7-5.12) shows that the closed form expressions (PCA and PCA-FM) are a reasonable match to the Monte Carlo experiments (using synthetic data), provided there is no violation of assumptions made in deriving the theoretical expressions (as given in sections 5.4 and 5.5). This suggests that the theoretical expressions are a credible representation of tracker performance (for each separate motion model) under varied clutter level. For each experiment more than 100 separate Monte Carlo runs were made and the average considered.

The trackers (ZVT, CVT, CAT) were applied to track corner features (extracted independently of the tracker, using the KLT and Harris corner detection algorithms) for image sequences ranging from about 10 to 50 frames. We used a variety of different image sequences with the feature to be tracked moving with varied motion (due either to feature motion or camera motion). Figures (5.13)-(5.16) and (5.17)-(5.20) shows the track life and track purity results for the PUMA (30 frames), Toy-Car (9 frames), Walking-Man (50 frames) and RUBIC (20 frames) image sequences respectively. From these results it is reasonably clear that a CAT or a CVT tracker gives the best tracking results for the PUMA sequence, while a CVT seems more suitable for the Walking man and Toy car sequences. A ZVT is adequate for the RUBIC sequence, this is because the inter-frame motion for the RUBIC sequence is very small, thus a zero velocity tracker is able to produce feature trajectories that are comparable to CVT and CAT. The same results are independently verified by visually inspecting the qualitative results given in Fig. 5.21 (these tracks are obtained by using the MHT employing the 3 different motion models discussed. They are the same figures as shown in Chapter 4 with the addition of ZVT applied to each of the sequences). The motion model that gives the best quality trajectories (for each image sequence) is the same as the ones revealed by the quantitative results displayed in Figs. 5.13-5.20), thus confirming the consistency of the results presented.

Another noteworthy observation is that the ZVT recovers better from a false match than CVT or CAT. A CVT recovers better than a CAT. The reason is that the ZVT requires only 1 past position to predict the next estimated position, where as a CVT requires 2 past position and a CAT requires 3 past positions. For example, for a CAT to completely recover from a mismatch, it needs to wait 3 time steps (requiring correct association at each of the 3 time steps) to make the next correct prediction.

5.7.2 MHT versus KLT in the Presence of Noise

In this section we provide a direct comparison between two types of feature point trackers. The 2 trackers considered are totally different in nature. The first tracker, the MHT uses a prediction (and matching) strategy for tracking features. The second tracker, the Kanade-Lucas-Tomasi (KLT) uses a non-prediction scheme (KLT uses an 'image patch comparison' strategy, the detail of the tracker was described in Chapter 3). The purpose of the comparison was to evaluate the robustness of the trackers in the presence of varied noise. The noise (uncorrelated noise) is artificially added to each frame (except the first frame) of a sequence at specified noise variances. The process was followed by feature extraction prior to applying the trackers.

For the MHT tracker, the most suitable motion model as applicable to each of the image sequence considered (obtained from results reported in sections 5.7.1) was employed. The number of features extracted in the initial frame for each sequence was limited to around 25 for clarity. The features were detected using the KLT feature detector [190]. Figures (5.22)-(5.25) shows the qualitative tracking results for both trackers, while Figure (5.26) shows the quantitative track-life results.

For the PUMA sequence, both trackers perform well at low noise levels ($\sigma < 10$), but the MHT provides better quality feature trajectories (longer trajectories) at higher noise levels than the KLT. This is clear from the results reported in Figs. (5.22) and (5.26a,b). For the Rubic sequence, both trackers provide equally good track results up to a noise level of around $\sigma = 20$ (Fig. 5.23), mainly because of the small inter frame displacement. At higher noise levels (noise variance $\sigma > 20$), MHT results in longer trajectories, this is evident from Fig. (5.26c,d), but not clearly observable from Fig. 5.23.

For the Walking man sequence MHT gives good track results at all noise levels considered, compared with the KLT tracker (see Figs. 5.24 and 5.26e,f). Since this is an outdoor sequence (generally prone to more image plane noise), added synthetic noise has a greater impact on the quality of trajectories obtained. It should also be noted that the features extracted from the walking person are not uniform throughout the sequence, because of the non-rigid nature of the object (walking man). As a result, the KLT tracker fails to produce good quality trajectories even at low noise levels due to failure of its image patch matching technique.

For the Toy car sequence, up to a noise level of $\sigma = 10$, the MHT gives good quality results (Fig. 5.25), but at higher noise levels, several incorrect trajectories are reported. The KLT, on the other hand gives shorter trajectories but are more reliable than the one obtained using MHT (see also Fig. 5.26c,d). The reason for this observation is because of occlusion (the jeep and the van are occluded between frames 4-6, see Appendix G). KLT cannot cope with occlusion because of its patch comparison strategy in consecutive frames, while MHT can cope with occlusion, but during occlusion added noise can distract the MHT tracker due to spurious measurements in the absence of the occluded object.

A general observation of the track results indicate that, MHT outperforms KLT in a noisy environment (for the examples considered), mainly because of its prediction /matching strategy. One reason for trackers breaking down at higher noise levels is because, the robustness of the corner detector employed (KL detector) decreases with the increase of noise (this fact is true for any corner detector). As the noise increases the number of false corners detected increases rapidly, which in turn results in high clutter density. Increase in clutter density lead to poor quality trajectories as discussed in the previous sections. Another reason, particularly for KLT, is the patch comparison mechanism it employs to match and track features in consecutive frames. Unfortunately the correlation technique that KLT employs is not suitable when considerable amount of noise is present in the image plane. Because of uncorrelated noise added to the sequences, the correlation match obtained between frames are very low, giving rise to poor quality trajectories (the KLT gives up tracking a feature if the match of image patches containing the feature between frames fall below a certain predefined threshold).

5.7.3 Theoretical versus Experimental Results

From the plots given in Figures 5.7–5.20, it is clear that there is deviation between the theoretical and experimental (simulation) results. The reasons for the deviation could be attributed to the following.

(a) In deriving the theoretical expressions, many assumptions were made (as described in sections 5.4.1-5.4.3 and 5.5.1-5.5.3). These were necessary in order to obtain feasible mathematical formulations. Whereas the simulated experiments were all based on the true dynamic motion models. Failure to meet these assumptions could have caused discrepancy.

(b) Obtaining PCA and PCA-FM using simulations were based on several hundreds of Mote Carlo runs (100-300). The number of runs might not have been sufficient for true representation of event types *A, B, C* and *D*. Particularly, the conditions required for event types *C* and *D* are rather restricted and thus the probabilities of events *C* and *D* occurring is very low, which in turn could have led to the calculation of less reliable probability values. This could have caused some deviation between the theoretical and simulated results.

(c) Numerical approximation and compromises in simulation implementation methods could also have caused some deviation, which were not accounted for in the analytical study.

However, despite these factors, the correspondence between the theoretical formulation and experimental results are close for low and high values of clutter density.

Theoretical versus experimental results (using synthetic data)

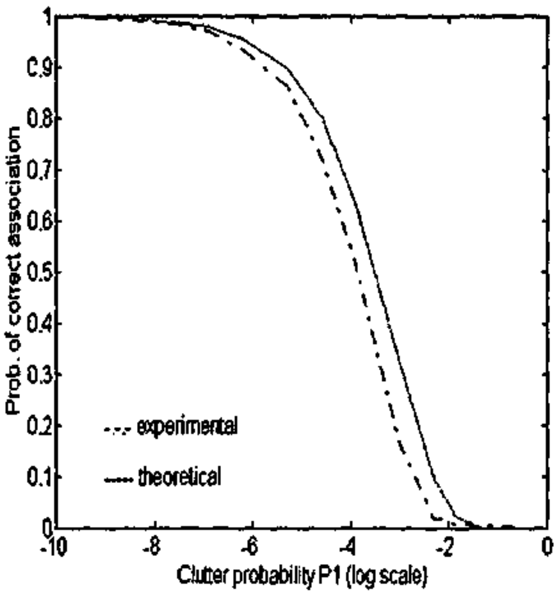


Figure 5.7

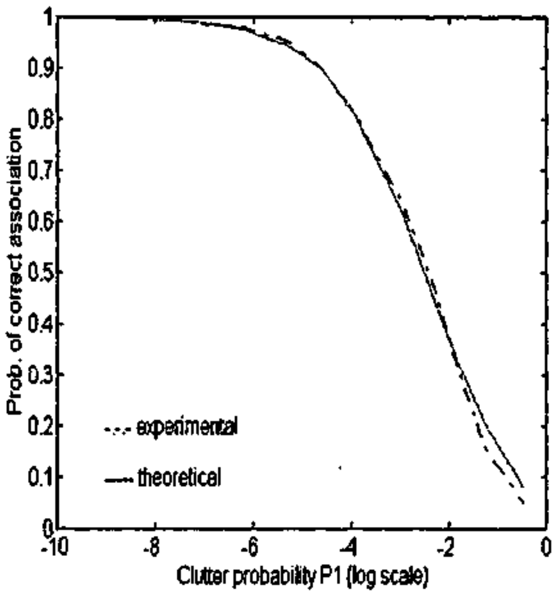


Figure 5.8

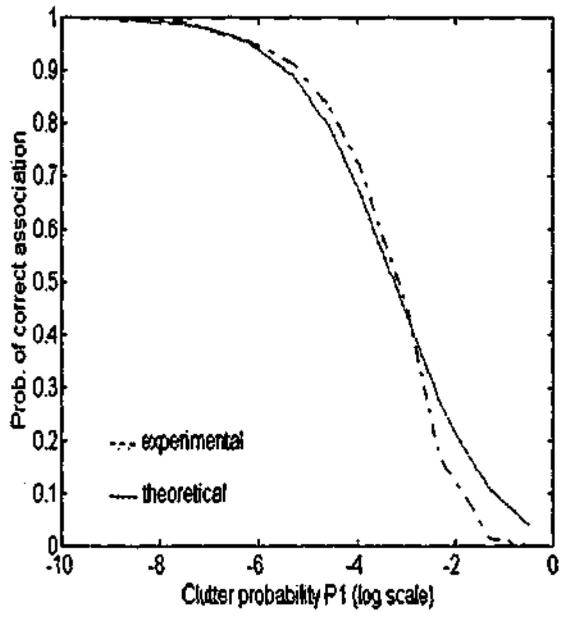


Figure 5.9

Figure 5. 7: Probability of correct association (PCA) for the ZVT. P_1 is shown on a log scale. A velocity error (v) of 6.25 pixels/unit time was modelled as an error term for the ZVT tracker.

Figure 5.8: Probability of correct association (PCA) for the CVT. The Gaussian noise variance σ=1.3 with an added acceleration error of 0.01 pixels/unit time/unit time (modelled as an error).

Figure 5.9: Probability of correct association (PCA) for the CAT. The Noise variance is set to σ=1.3.

Theoretical versus experimental results Cont.

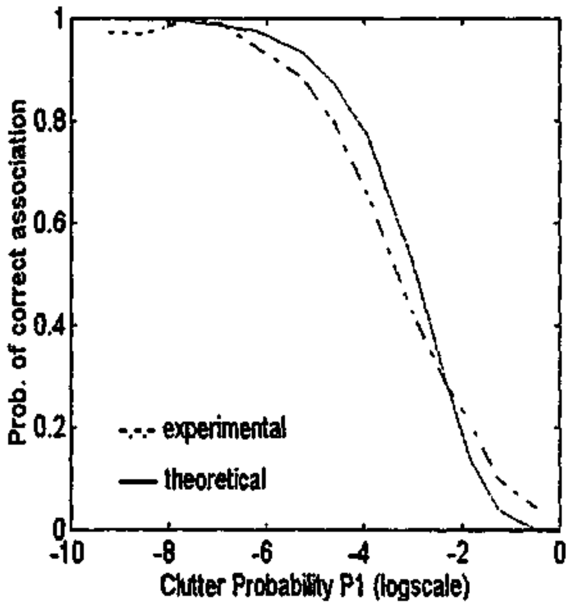


Figure 5.10

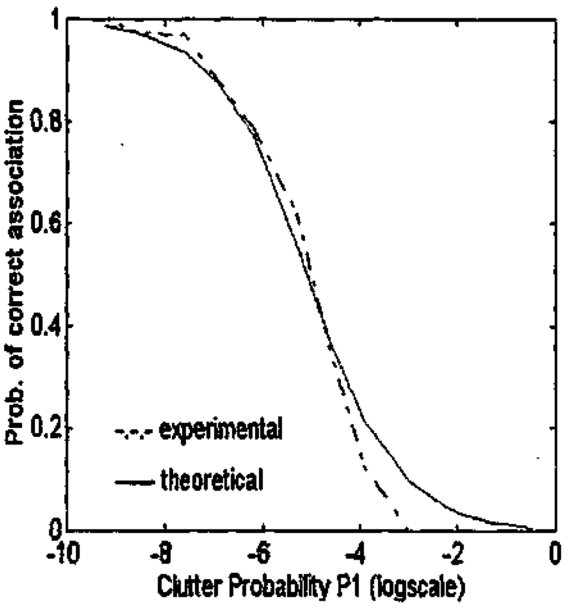


Figure 5.11

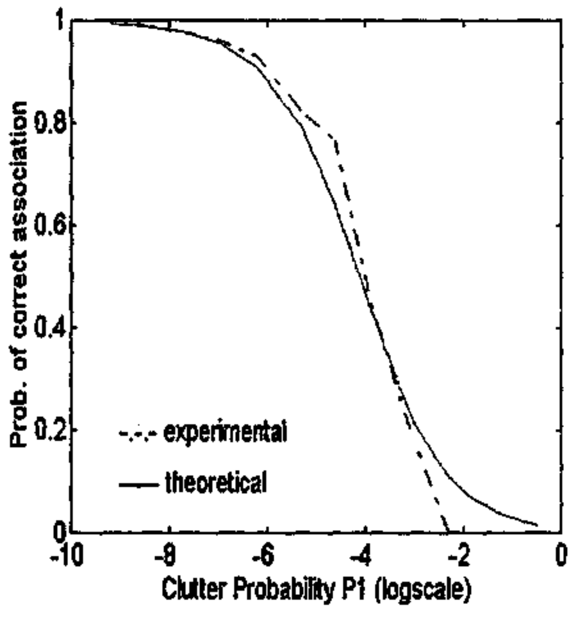


Figure 5.12

Figure 5.10: Probability of correct association for the ZVT for recovering from a mismatch (PCA-FM). P_1 is shown on a log scale. A velocity error v=1.0 pixels/unit time is applied for this example.

Figure 5.11: Probability of correct association for the CVT for recovering from a mismatch (PCA-FM). A 'disk of association' radius $r = 4.4$, (with $f(g_x, g_y) = 2.2$) is applied for this experiment. See text for details.

Figure 5.12: Probability of correct association for the CAT for recovering from a mismatch (PCA-FM). A 'disk of association' radius $r = 2.9$, (with $f(h_x, h_y) = 2$) is applied for this experiment. See text for details.

Track-life results (using real data)

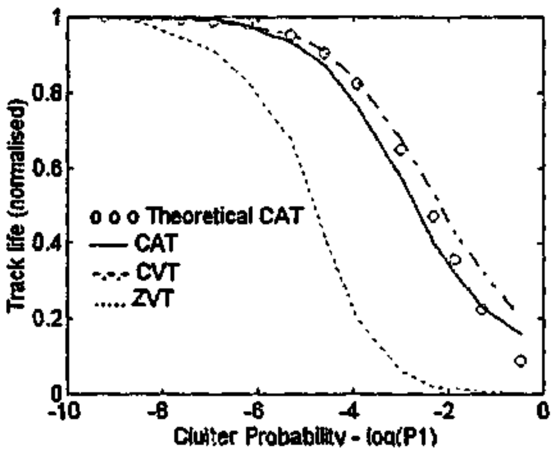


Figure 5.13

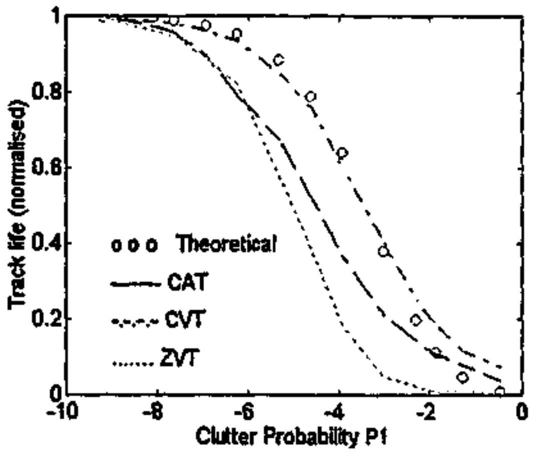


Figure 5.14

Figure 5.13: Track-life for the PUMA sequence plotted against clutter probability (shown in log scale). The theoretical model presented for the probability of correct association (for CAT) closely matches the track life obtained using a constant acceleration tracker (CAT) experimentally. The plot shows that a CAT or CVT provides the best quality trajectories under clutter for the PUMA sequence.

Figure 5.14: Track-life for Walking man sequence (shown in log scale). The track-life obtained by experiments for walking man sequence using a constant velocity model (CVT) gives the best match for the theoretical CVT model.

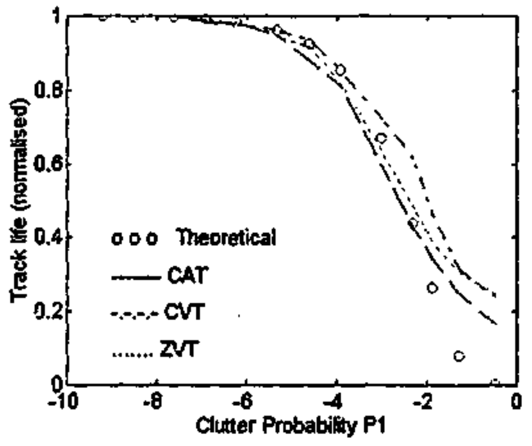


Figure 5.15

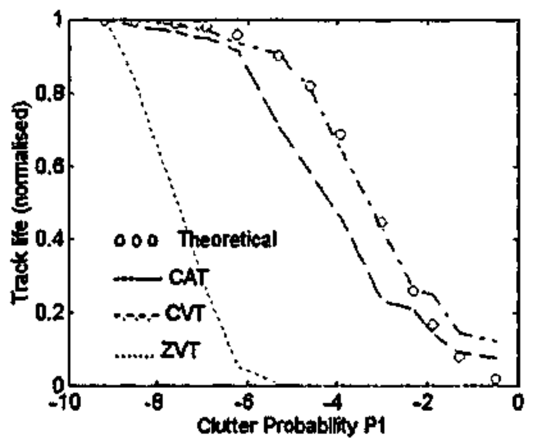
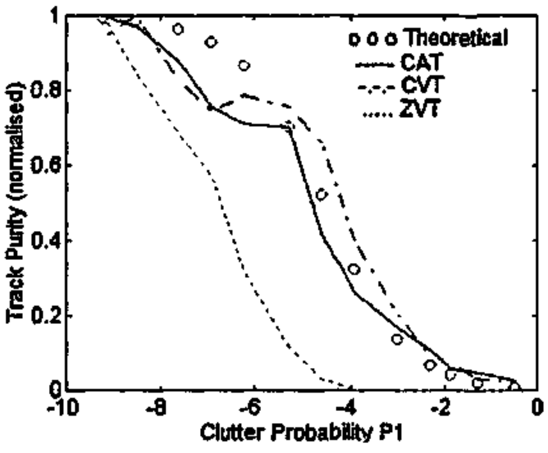


Figure 5.16

Figure 5.15: Track-life for the RUBIC sequence plotted against clutter probability (shown in log scale). The theoretical model presented for the probability of correct association (for ZVT) matches track-life obtained by all 3 trackers. Since the motion is very small between frames, a ZVT tracker seems adequate.

Figure 5.16: Track-life for Toy car sequence (shown in log scale). The track-life obtained for toy car sequence using a constant velocity model (CVT) gives the best match for the theoretical CVT model presented.

Track-purity results (using real data)



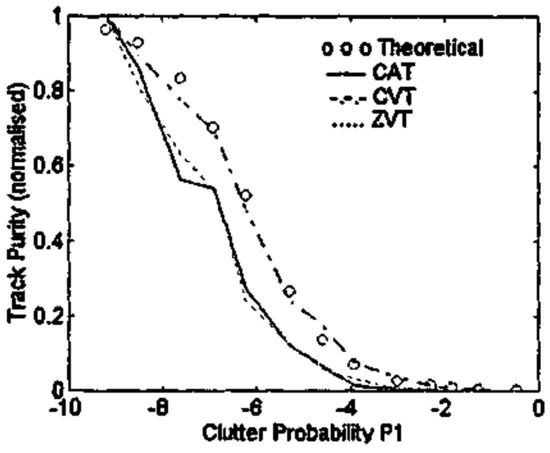
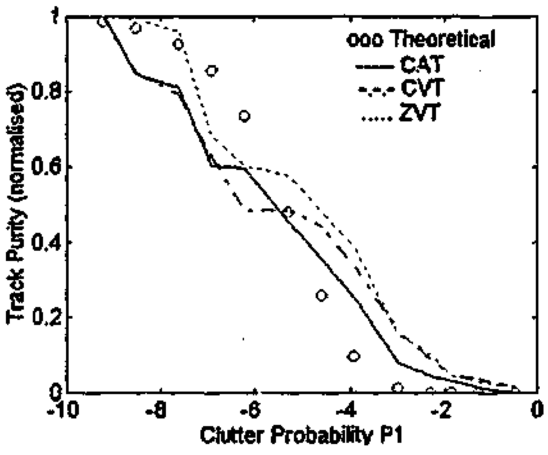


Figure: 5.17 *Figure: 5.18*

Figure 5.17: Track-purity for the PUMA sequence plotted against clutter probability (shown in log scale). The theoretical model presented for the probability of correct association for recovering from a mismatch (for CAT) is a reasonable approximation to the track-purity obtained using a constant acceleration tracker (CAT) by simulations. This is an indication that for feature tracking for the PUMA sequence a CAT or CVT provides the best track result.

Figure 5.18: Track-purity for Walking man sequence (shown in log scale). The track-purity obtained for walking man sequence (by simulations) using a constant velocity model (CVT) gives the best match for the theoretical CVT model.



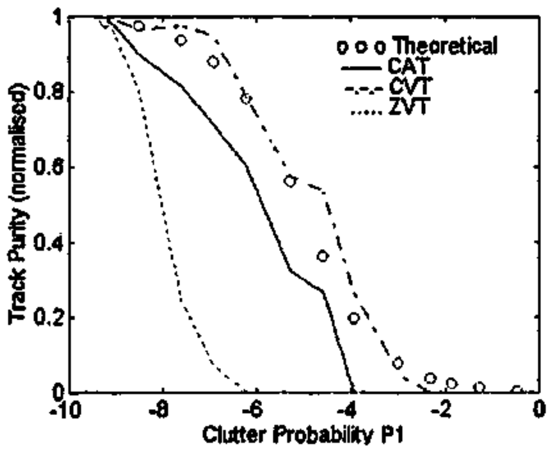


Figure: 5.19 *Figure: 5.20*

Figure 5.19: Track-purity for the RUBIC sequence plotted against clutter probability (shown in log scale). The theoretical model presented for the probability of correct association (for ZVT) when recovering from a mismatch is a reasonable match for tracks obtained by all 3 trackers (using simulations). Since the motion is very small between frames, a ZVT tracker is adequate.

Figure 5.20: Track-purity for Toy car sequence (shown in log scale). The track-purity obtained (by simulations) for toy car sequence using a constant velocity model (CVT) is reasonably a good match for the theoretical CVT model presented.

Note: The track-purity results are consistent with the track-life results (Fig. 5.13-Fig. 5.16).

Tracks Obtained Using Different Motion Models

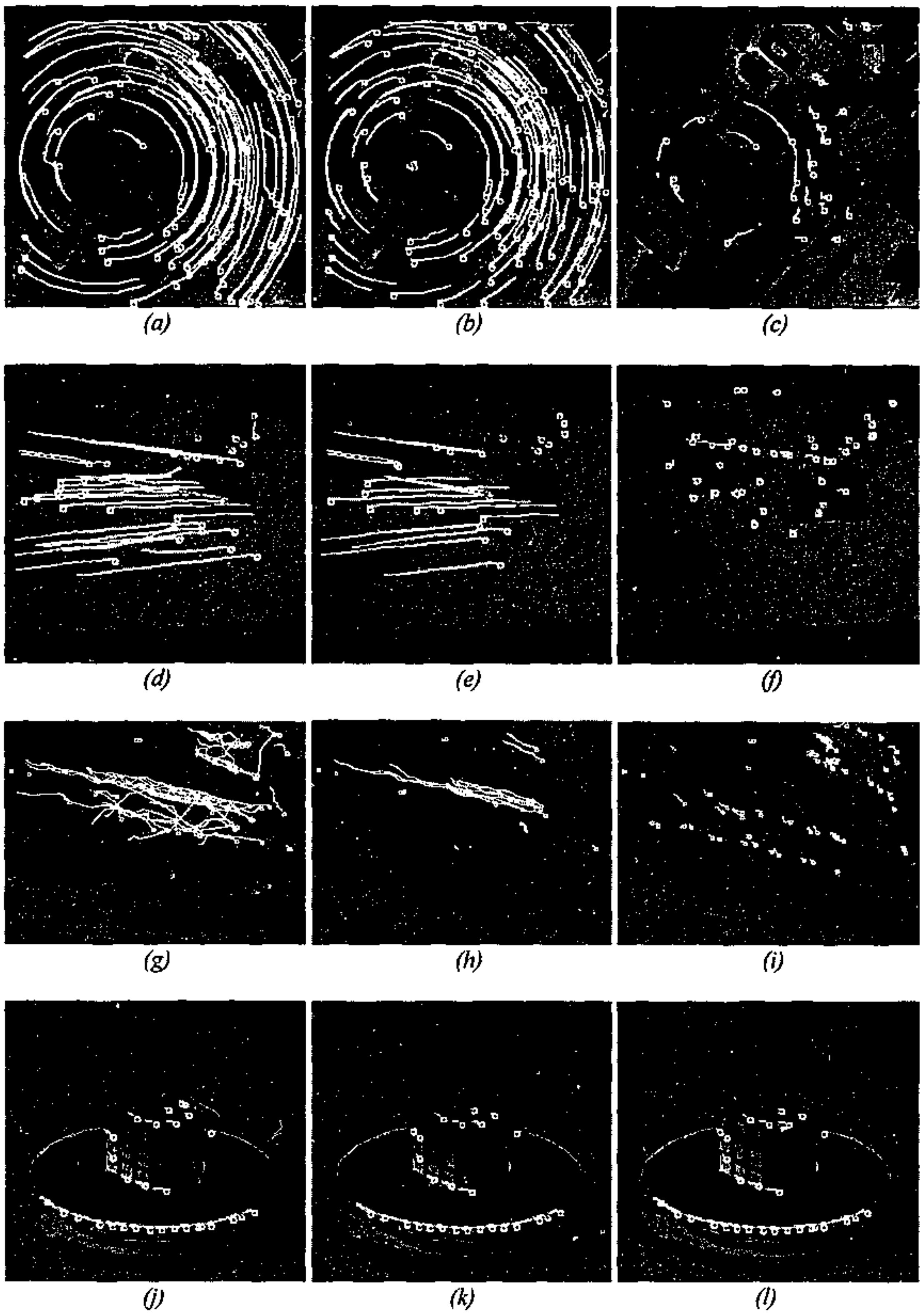
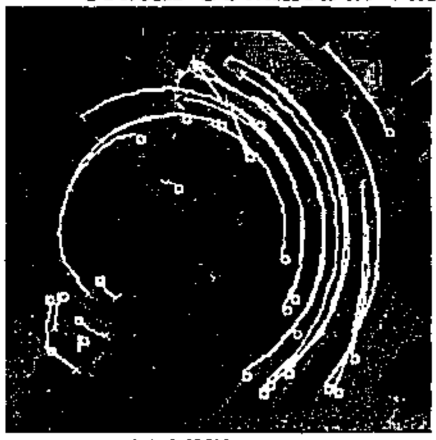
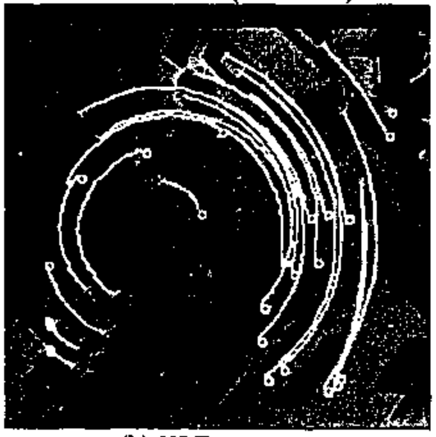
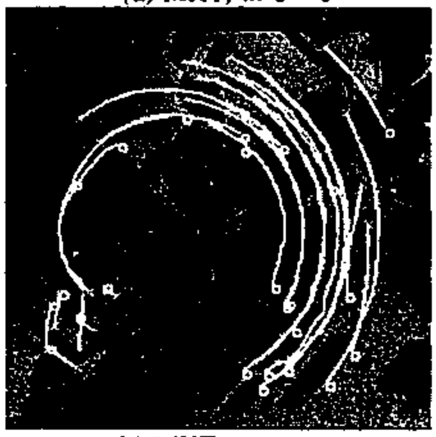


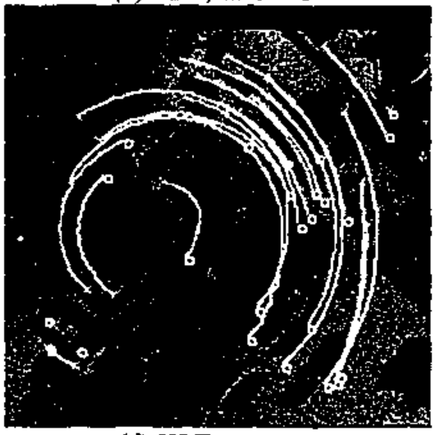
Figure 5.21. Qualitative results of trajectories obtained by employing the MHT tracker with different motion model. For each sequence the tracks are displayed on frame-1. The circle indicates the end or termination of a track. (a) PUMA using a CAT, (b) PUMA using a CVT (shorter tracks than using CAT), (c) PUMA using a ZVT. (d) Toy-car using a CAT (observe the number of false trajectories despite longer tracks), (e) Toy-car using a CVT, (f) Toy-car using a ZVT. (g) Walking man using a CAT (large number of false tracks due to over-constraint), (h) Walking man using a CVT, (i) Walking man using a ZVT. (j) Rubic using a CAT, (k) Rubic using a CVT, (l) Rubic using a ZVT (similar results to using CAT or CVT). The results show that the quality of trajectories obtained for each image sequence is influenced by the motion model employed. Thus choosing the most appropriate motion model is of importance for good tracking result.

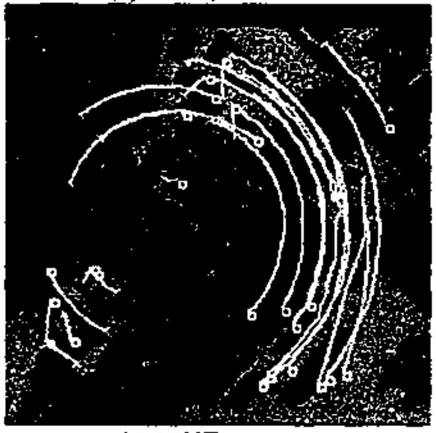
Tracks Obtained at Varied Noise Levels (PUMA)

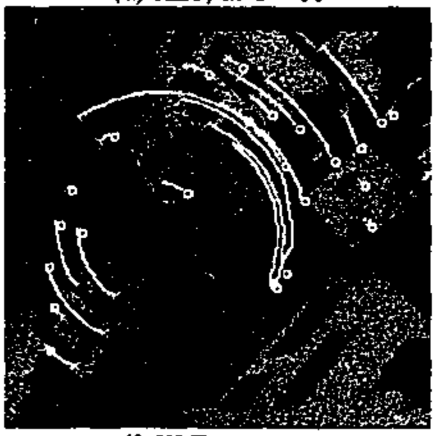

(a) MHT, at σ = 0

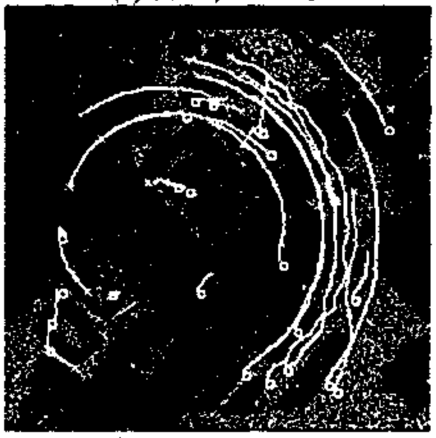

(b) KLT, at σ = 0


(c) MHT, at σ = 10


(d) KLT, at σ = 10


(e) MHT, at σ = 20


(f) KLT, at σ = 20


(g) MHT, at σ = 30

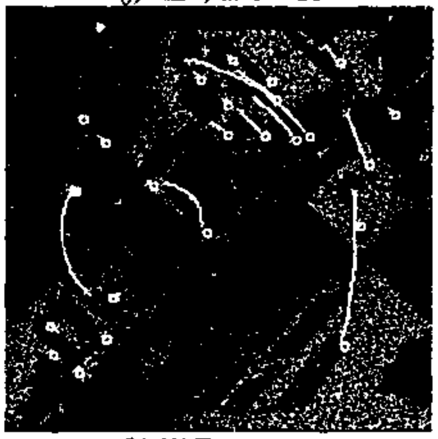
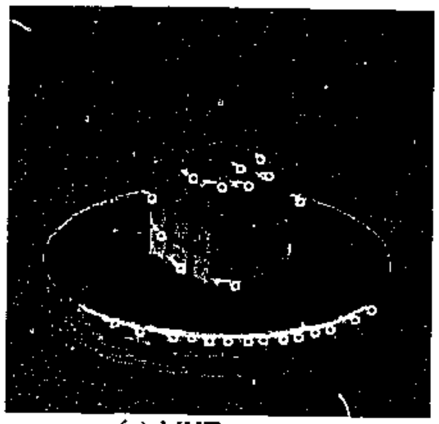
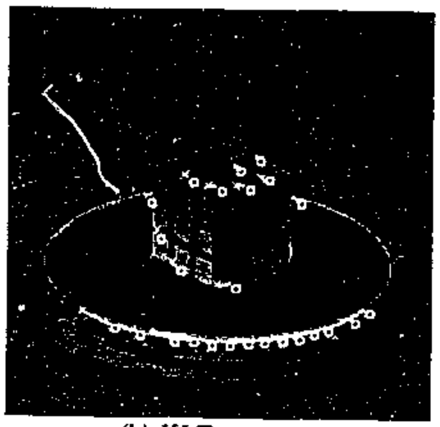
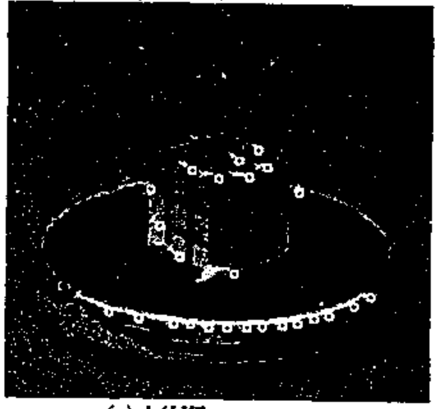

(h) KLT, at σ = 30

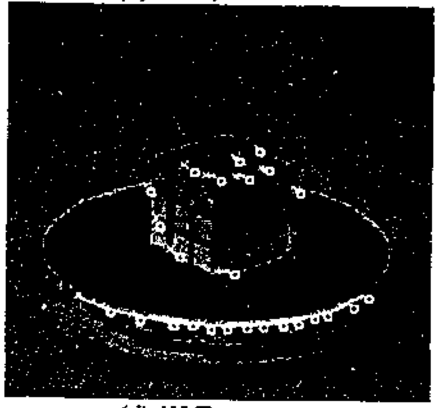
Figure 5.22: Qualitative results for the PUMA sequence under noise: Performances of MHT (using CAT) at noise variance (a) σ = 0, (c) σ =10, (e) σ =20 and (g) σ =30. Performances of KLT at noise variance (b) σ = 0, (d) σ =10, (f) σ =20 and (h) σ =30.

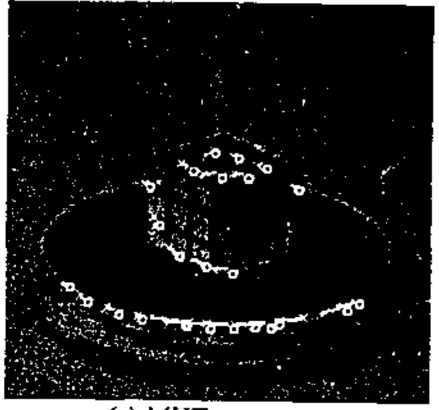
Tracks Obtained at Varied Noise Levels (Rubic)

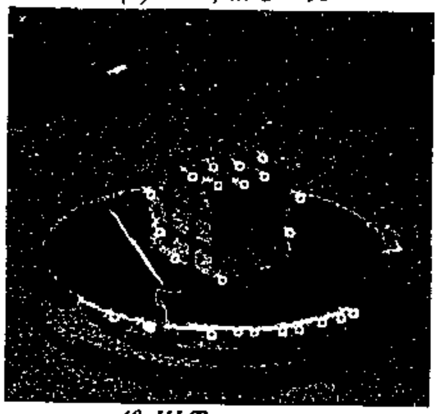

(a) MHT, at $\sigma = 0$

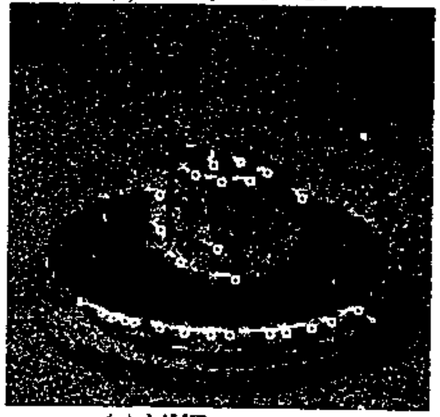

(b) KLT, at $\sigma = 0$


(c) MHT, at $\sigma = 10$


(d) KLT, at $\sigma = 10$


(e) MHT, at $\sigma = 20$


(f) KLT, at $\sigma = 20$


(g) MHT, at $\sigma = 30$

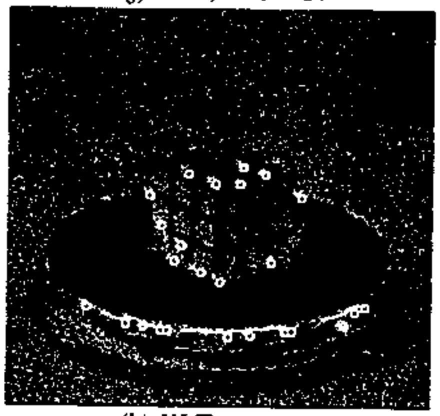
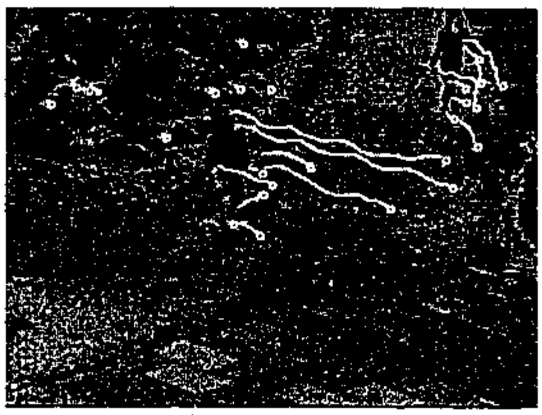
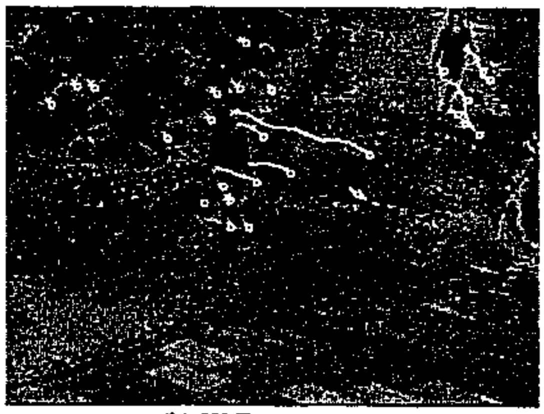
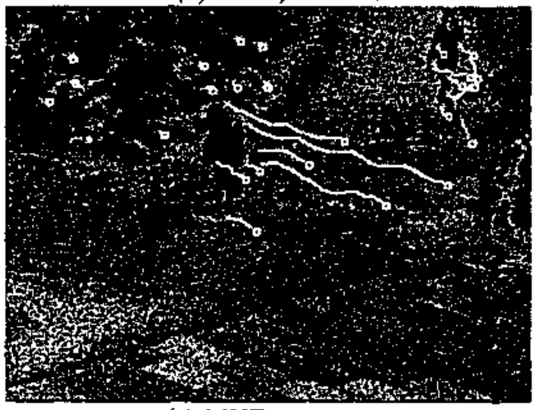

(h) KLT, at $\sigma = 30$

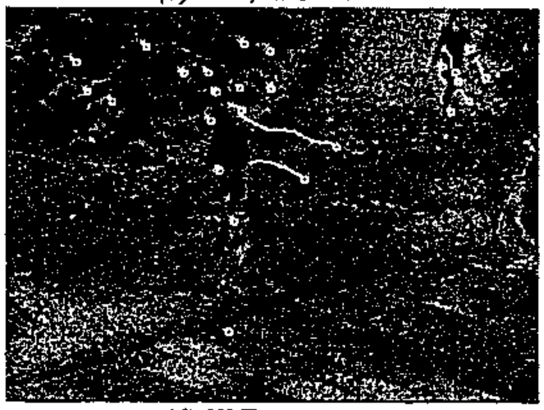
Figure 5.23: Qualitative results for the Rubic sequence under noise: Performances of MHT (using ZVT) at noise variance (a) $\sigma = 0$, (c) $\sigma = 10$, (e) $\sigma = 20$ and (g) $\sigma = 30$. Performances of KLT at noise variance (b) $\sigma = 0$, (d) $\sigma = 10$, (f) $\sigma = 20$ and (h) $\sigma = 30$.

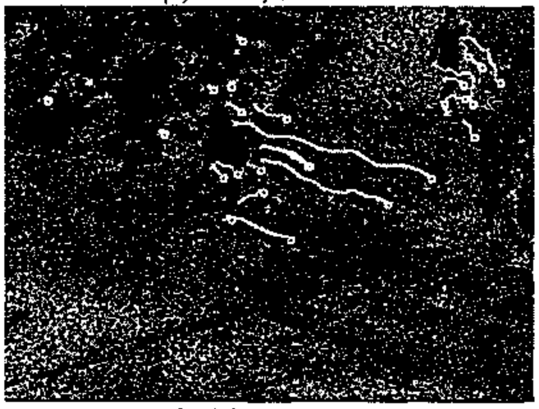
Tracks Obtained at Varied Noise Levels (Walking Man)

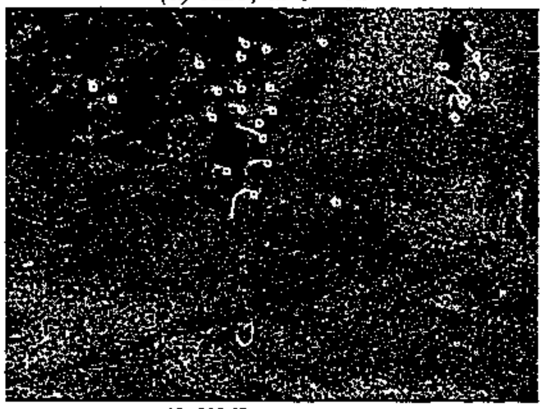

(a) MHT, at $\sigma = 0$

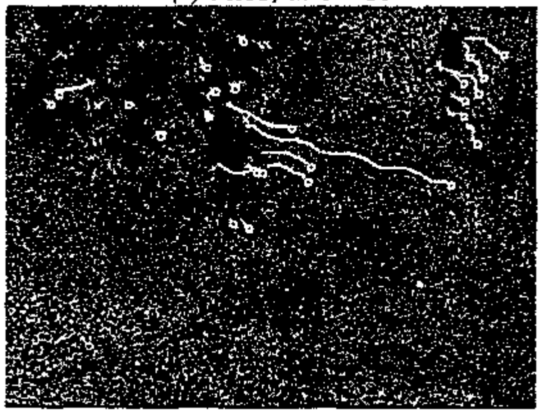

(b) KLT, at $\sigma = 0$


(c) MHT, at $\sigma = 10$


(d) KLT, at $\sigma = 10$


(e) MHT, at $\sigma = 20$


(f) KLT, at $\sigma = 20$


(g) MHT, at $\sigma = 30$

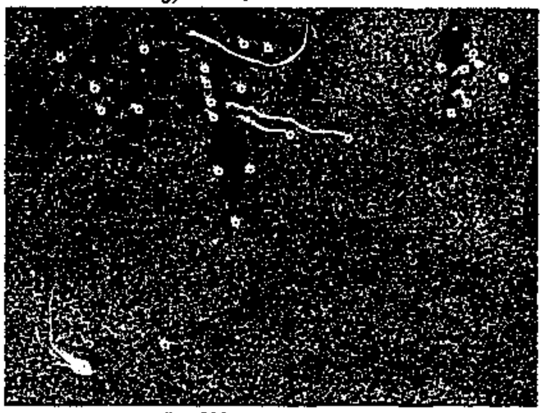
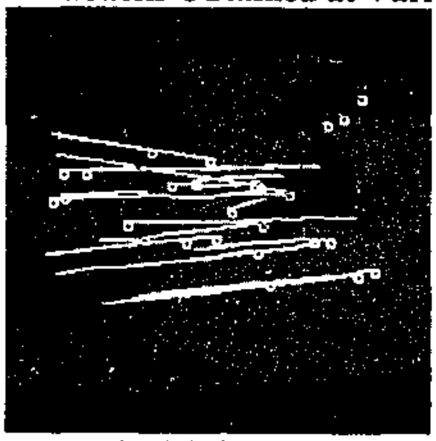
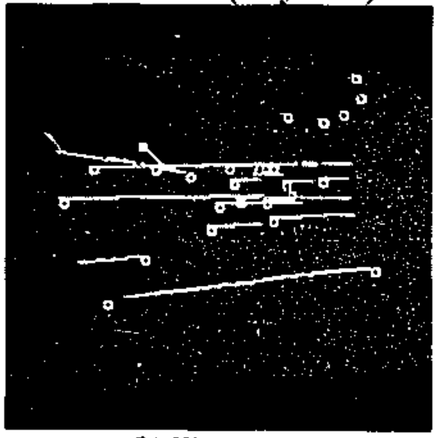
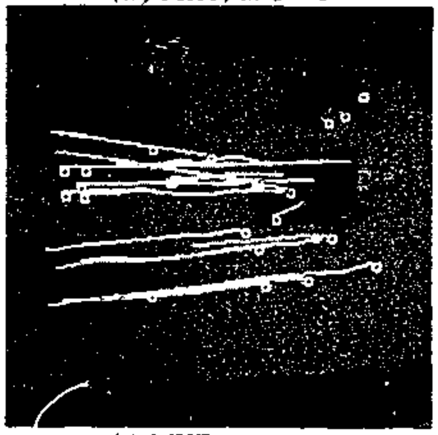

(h) KLT, at $\sigma = 30$

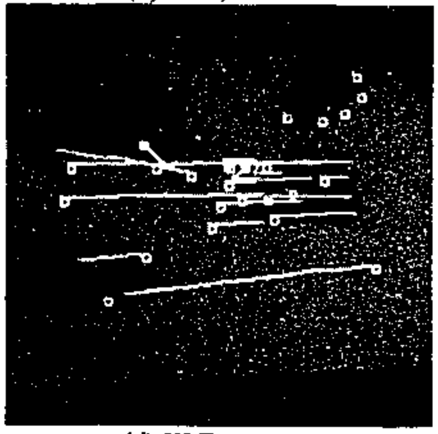
Figure 5.24: Qualitative results for the Walking man sequence under noise: Performances of MHT (using CVT) at noise variance (a) $\sigma = 0$, (c) $\sigma = 10$, (e) $\sigma = 20$ and (g) $\sigma = 30$. Performances of KLT at noise variance (b) $\sigma = 0$, (d) $\sigma = 10$, (f) $\sigma = 20$ and (h) $\sigma = 30$.

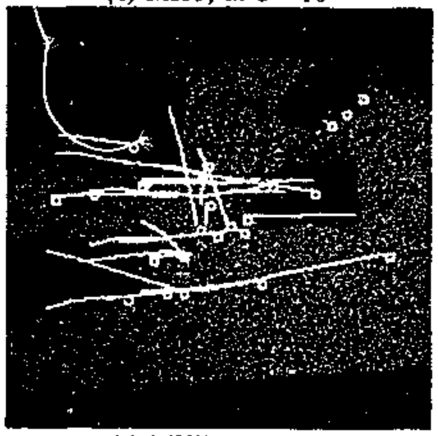
Tracks Obtained at Varied Noise Levels (Toy Car)

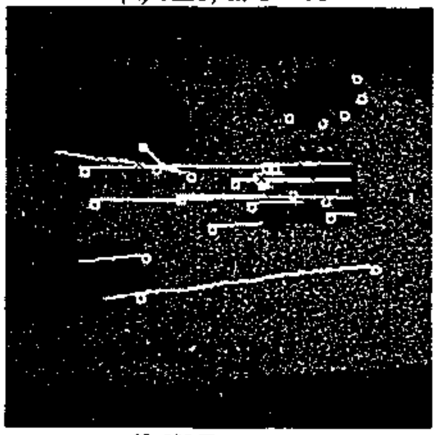

(a) MHT, at σ = 0

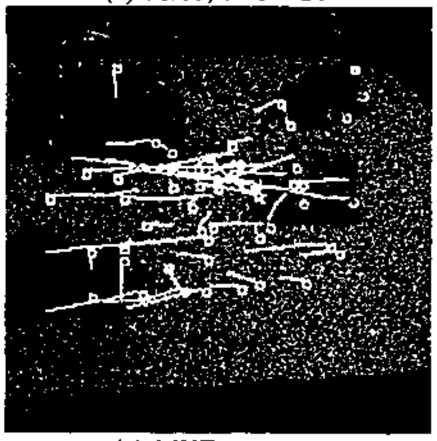

(b) KLT, at σ = 0


(c) MHT, at σ = 10


(d) KLT, at σ = 10


(e) MHT, at σ = 20


(f) KLT, at σ = 20


(g) MHT, at σ = 30

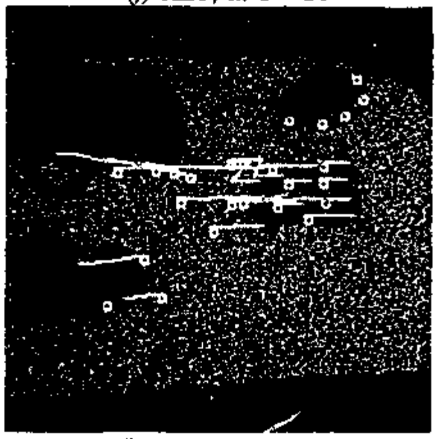
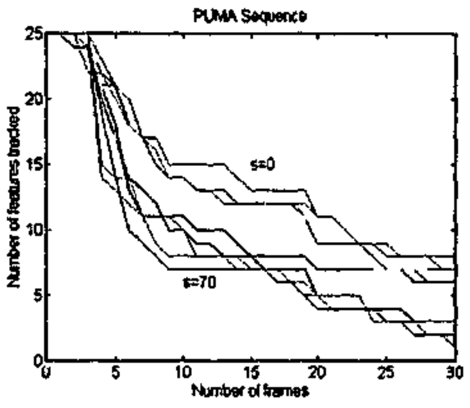
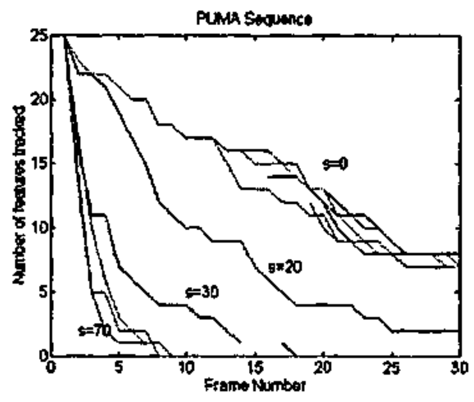

(h) KLT, at σ = 30

Figure 5.25: Qualitative results for toy car sequence under noise: Performances of MHT (using CVT) at noise variance (a) σ = 0, (c) σ =10, (e) σ =20 and (g) σ =30. Performances of KLT at noise variance (b) σ = 0, (d) σ =10, (f) σ =20 and (h) σ =30.

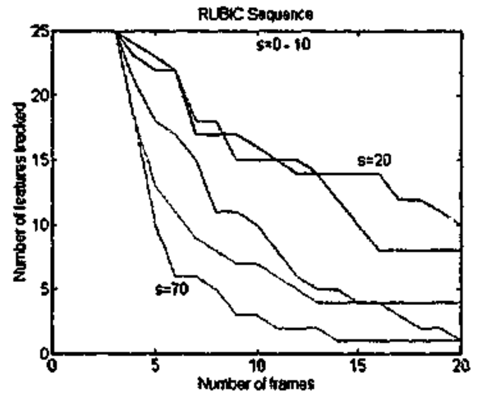
Track-Life Results



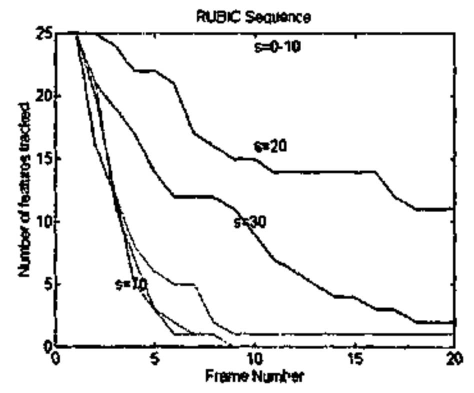
(a)



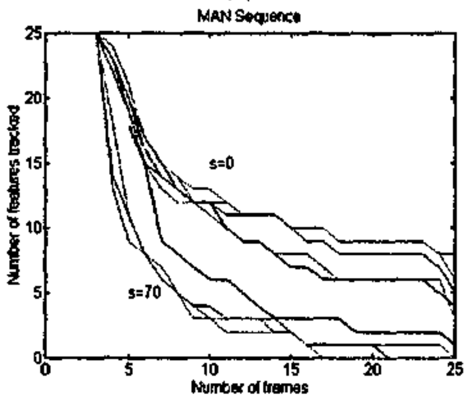
(b)



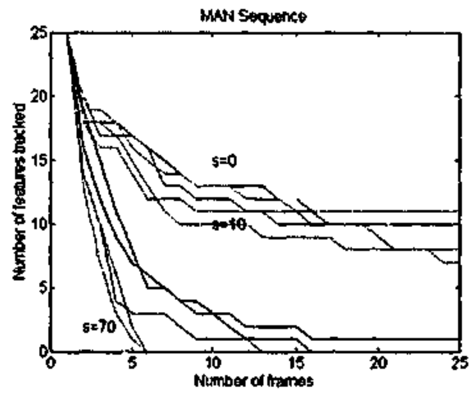
(c)



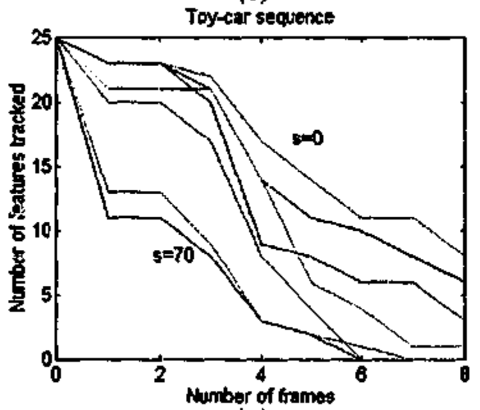
(d)



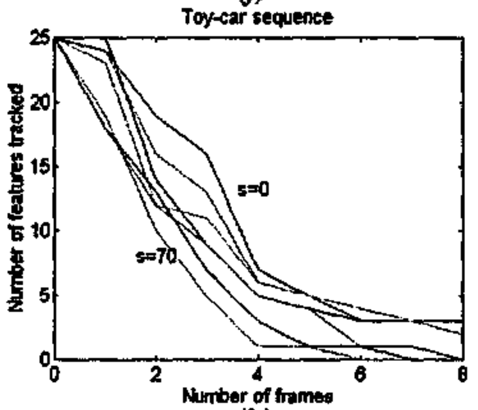
(e)



(f)



(g)



(h)

Figure 5.26: Track life for the 4 image sequences considered (s refers to noise variance). (a) Using MHT tracker for the PUMA sequence. (b) Using KLT tracker for the PUMA sequence. (c) Using MHT tracker for the Rubic sequence. (d) Using KLT tracker for the Rubic sequence. (e) Using MHT tracker for the Walking man sequence. (f) Using KLT tracker for the Walking man sequence. (g) Using MHT tracker for the Toy-car sequence. (h) Using KLT tracker for the Toy-car sequence.

5.8 Conclusion

We have provided a performance prediction scheme for simple linear trackers based on different motion models when tracking under clutter. The closed form expressions provided for performance prediction have been shown to be more efficient than the conventional Monte-Carlo experiments. The method provided is useful to compare the performances of a visual tracker based on different motion models, thus indicating to the motion model that gives the best quality trajectories. We have also experimentally demonstrated that choosing the most appropriate motion model is important for any tracker to provide good tracking results. The tracker performance under noise has revealed the degree of robustness of the trackers. This was demonstrated by applying the MHT and KLT trackers to track features at varied noise levels.

It is also worth noting that the best motion model selected for the image sequences considered in Chapter 4 are consistent with the results reported in this Chapter, thus confirming the consistency of motion model selection results for each sequence considered.

Chapter 6

Extension of a Point Feature Tracker for Rigid Object Tracking

Abstract

This chapter presents an object tracking technique based on the Bayesian Multiple Hypothesis Tracking (MHT) approach. Two algorithms, both based on the MHT technique are combined to generate an object tracker [182]. The first MHT algorithm is employed for contour segmentation [57]. The segmentation of contours is based on an edge map. The segmented contours are then merged to form recognisable objects. The second MHT algorithm is used in the temporal tracking of a selected object from the initial frame (as explained in chapter 4). An object is represented by key feature points that are extracted from it. The key points (mostly corner points) are detected using information obtained from the edge map. These key points are then tracked through the sequence. To confirm the correctness of the tracked key points, the location of the key points on the trajectory are verified against the segmented object identified in each frame. If an acceptable number of key-points lie on or near the contour of the object in a particular frame (say n-th frame), we conclude that the selected object has been tracked (identified) successfully in frame n.

6.1 Introduction

The primary purpose of this chapter is to track a selected object (as opposed to a single point feature) from the initial frame through the image sequence. The process is an attempt to extend the point feature tracking introduced in chapter 4 to object tracking. In this case, key points from the object are selected using a curvature scale space technique [134] to represent that object. The key points are temporally tracked and are validated against the object contour (obtained by grouping edge segments) in each frame. The tracking technique involves applying the MHT algorithm in two stages: The first stage is for contour grouping (object identification based on segmented edges) and the second stage is for temporal tracking of key features (from the object of interest). For the contour grouping process, we employed the algorithm developed by Cox et al [57], and for the key point tracking procedure we used the tracker introduced in chapter 4. Both algorithms combine to provide a rigid-object tracker (the tracker cannot effectively be applied to deformable objects for reasons explained later in this chapter).

The set of image contours produced by objects in a scene, encode important information about their shape, position, and orientation. Image contours arise from discontinuities in the underlying intensity pattern, due to the interaction of surface geometry and illumination. A large body of work, from such areas as model-based object recognition and contour motion flow (as discussed in chapter 2), depend critically on the reliable extraction of image contours. Reliable image contours are necessary to identify an object with certainty, which in turn is necessary for tracking the object over a period of time in a sequence of images. We use the term 'object' for a group of edge segments that form a recognisable object (identified as belonging to the same object). The object will be identified by an enclosed (*or* near-enclosed) contour.

This chapter is organised as follows: Section 6.2 gives a brief description of the Multiple Hypothesis Tracking (MHT) approach relating to edge segmentation. Section 6.3 shows how the multiple hypothesis approach can be used for object recognition. In section 6.4 we briefly show the process to extract key points from an object, and the MHT approach for tracking key point features through an image sequence. Section 6.5 provides the object-tracking framework employed using methods described in section 6.3 and 6.4. Section 6.6 gives results obtained from experiments. Section 6.7 gives a general discussion, and finally section 6.8 provides the conclusion.

6.2 Multiple Hypothesis Framework for Contour Grouping

This section briefly describes the multiple hypothesis approach in relation to contour segmentation. The details of which were discussed in chapter 4.

Fig. 6.1 outlines the basic operation of the MHT algorithm for contour grouping (observe the minor difference to that of Fig. 4.1. Instead of corners, edges are extracted). At each iteration, there are a set of hypotheses (initially null), each one representing a different interpretation of the edge points. Each hypothesis is a collection of contours, and at each iteration each contour predicts the location of the next edgel as the algorithm follows the contour in unit increments of arc length. An adaptive search region is created about each of these predicted locations as shown in Figure 6.2 [57]. Measurements are extracted from these surveillance regions and matched to predictions based on the statistical Mahalanobis distance (similar to that discussed in chapter 4). This matching process reveals ambiguities in the assignment of measurements to contours. This procedure provides an associated ambiguity matrix (Ω) for each global hypothesis from which it is necessary to generate a set of legal assignments. As a result, the hypothesis tree grow another level in depth, a parent hypothesis generating a series of hypotheses each being a possible interpretation of the measurements. The probability of each new hypothesis is calculated based on assumptions described in chapter 4. Finally, a pruning stage is invoked to constrain the exponentially growing hypothesis tree. This completes one iteration of the algorithm. Appendix B contains the required mathematical formulation for this application.

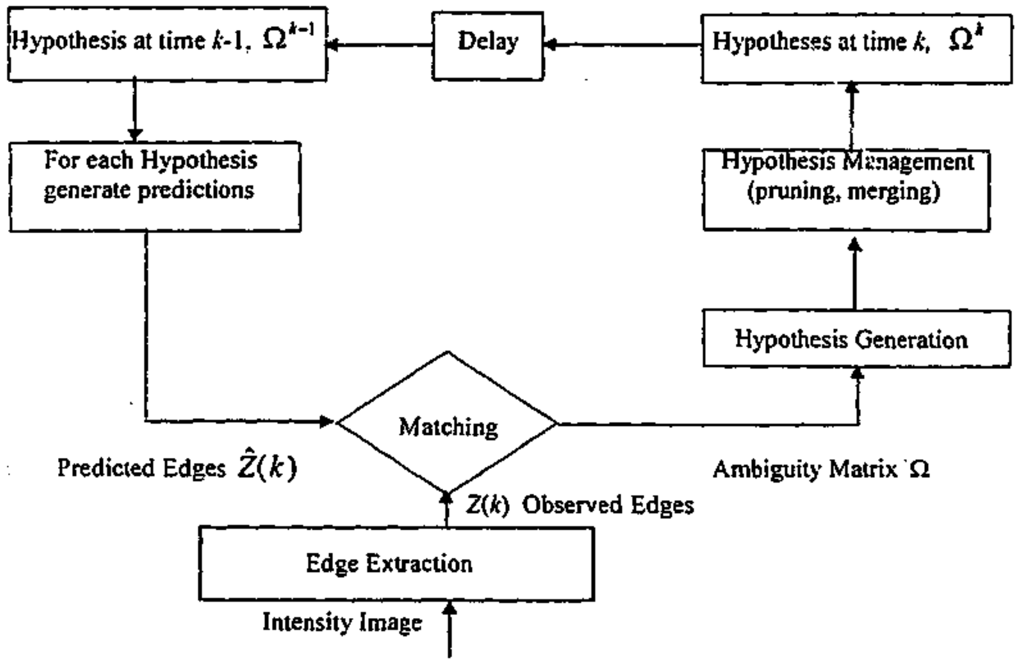

Hypothesis at time k-1, Ω^{k-1}
Delay
Hypotheses at time k, Ω^{k}
For each Hypothesis generate predictions
Hypothesis Management (pruning, merging)
Hypothesis Generation
Matching
Predicted Edges $\hat{Z}(k)$
Ambiguity Matrix Ω
$Z(k)$ Observed Edges
Edge Extraction
Intensity Image

Figure 6.1: Outline of the multiple hypothesis algorithm for edge grouping

In the following sections we briefly describe the contour grouping algorithm employed, and the key point selection and tracking process used. Both these methods are based on the multiple hypothesis approach.

6.3 Object Recognition

6.3.1 Contour Segmentation

The contour grouping problem examined in this chapter, involves assigning edge pixels produced by an edge detector [40, 31] to a set of continuous curves. Associating edge points with contours is difficult because the input data (from edge detectors) is noisy; there is uncertainty in the position of the edge, there may be false and / or missing points, and contours may intersect and interfere with one another. There are four basic requirements for a successful contour segmentation algorithm. First, there must be a mechanism for integrating information in the neighbourhood of an edgel to avoid making irrevocable grouping decisions based on insufficient data. Second, there must be a prior model for the smoothness of the curve to base grouping decisions on. This model must have an intuitive parameterisation and sufficient generality to describe arbitrary curves of interest. Third, it must incorporate noise models for the edge detector, to optimally incorporate noisy measurements, and detect and remove spurious edges. And finally, since intersecting curves are common, the algorithm must be able to handle these as well. The algorithm due to Cox et.al. [57, 59] is one which has a unified framework that incorporates these four requirements, and we will use this algorithm for contour segmentation.

The contour grouping is formulated as a Bayesian multiple hypothesis 'tracking' problem (as in [152]). The algorithm has 3 main components. A dynamic contour model that encodes the smoothness prior, a measurement model that incorporates edge-detector noise characteristics, and a Bayesian hypothesis tree that encodes the likelihood of each possible edge assignment and permits multiple hypothesis to develop in parallel until sufficient information is available to make a decision.

A key step in assigning probabilities to segmentation hypothesis is the computation of the likelihood that a given measurement originated from a certain contour. This likelihood computation depends on two things: a dynamic model that describes the evolution of the curve in the image, and a measurement model that describes how curves produce edgels. In this formulation, the curve state vector is $[x \;\; \dot{x} \;\; y \;\; \dot{y}]^T$ (where (x, y) are the position in a Cartesian coordinate) and its dynamics are described by a linear noise-driven acceleration model common in the tracking literature [5, 6] (also discussed in chapter 4). The autocorrelation of the white Gaussian acceleration noise can be varied to model curves of arbitrary smoothness. Thus the tip (end point) of the contour as a function of arc length, u, is $(x(u), y(u))$ and has tangent $(\dot{x}(u), \dot{y}(u))$. Since many edge detectors provide gradient information, it is assumed that the entire state vector is available for measurement (a good edge detector such as Canny [40], Boie-Cox algorithm [31] etc. which provide both position and coarse gradient information (horizontal, vertical, and two diagonals) is employable for this application). A Kalman filter is then employed to estimate curve state and predict the location of edgels. These predictions are combined with actual measurements to produce likelihoods.

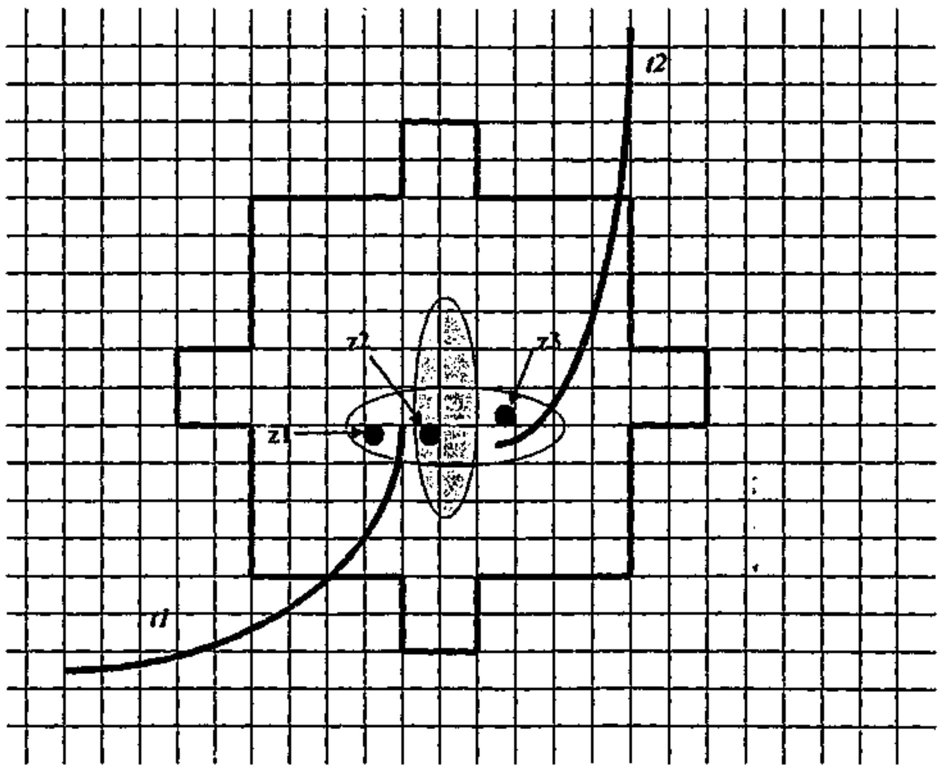


Figure 6.2: Predicted contour locations, a surveillance region and statistical Mahalanobis (elliptical) regions for a situation with two known contours (t1 and t2) and three new measurements (z1, z2 and z3).

Once the location of a given curve has been predicted by the Kalman filter and discretized to image coordinates, a surveillance region is employed to extract measurements. A surveillance region is an adaptive variable sized window that travels with the tip of the contour and is used to extract measurements from the edge map. Every iteration, each contour searches for edge points in a series of circles of increasing diameter centred at the predicted contour endpoint. The search halts as soon as at least one measurement is found, or the maximum search radius is reached. The size of the surveillance region determines the distance the curve must travel in that time period, and is reflected in the step size for the curve. The use of a set of windows of increasing size ensures that no more than one measurement from the given contour will be found in a single time period.

The search for measurements takes place after the prediction phase of the state estimator generates an extrapolated endpoint location, (x, y), for the contour. This location determines the discrete image coordinates, (x_i, y_j), at which the surveillance region is centred. If there is no edge at the predicted location, concentric circles (see Fig. 6.3), of radius $1, \sqrt{2}, 2, \sqrt{5}$, are searched for edgels (the radii define discrete pixel neighbourhoods). These surveillance regions are labelled 1 to 5 in Fig. 6.3 (It should be noted that the surveillance region of a contour is not equivalent to its validation region, which is defined by the Mahalanobis distance and is depicted in Fig. 6.3 as an ellipse). It is these measurements that form segmentation hypothesis whose probabilities are computed. See Appendix B and [57] for details.

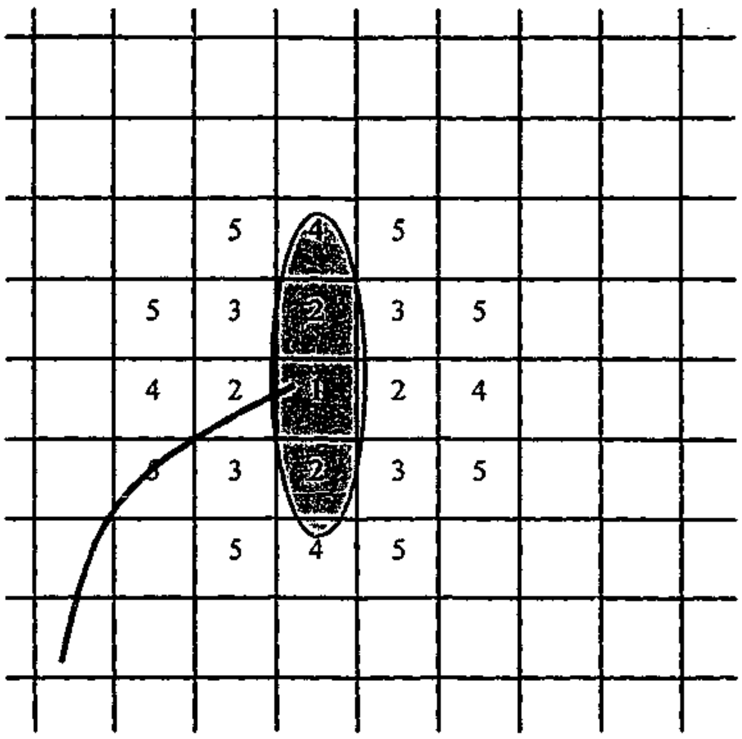


Figure 6.3: A contour, its surveillance region (labelled 1 – 5) and its validation region.

6.3.2 Contour Merging

The grouped contours resulting from the above mentioned process still might have breaks and gaps between segments of the same object. A further refinement process can be employed to merge segments to form identifiable objects. A merging technique is employed by using a distance test (eg: Mahalanobis distance) applied to the end points of contours (assuming a non-closed contour). In this case the multiple contours can be merged to recover the correct segmentation, compensating for the incorrect initial conditions. Two contours with state estimates $\hat{x}_i$ and $\hat{x}_j$ at common boundary are merged if $dx'_{ij}\,\mathrm{T}^{-1}_{ij}\,dx_{ij} \le \delta$, where $dx_{ij} = \hat{x}_i - \hat{x}_j$. T_{ij} is the covariance, and δ is obtained from χ^2 tables or set appropriately as a threshold. This test is applied after the algorithm produces an initial segmentation. The procedure resolves many ambiguities left by the contour segmentation algorithm. A simpler algorithm can also be used by just using the end-point positions and derivatives of the end points of each curve (produced by the edge detector) as shown below.

Algorithm for merging contours

```
1) (i)  Read Contour Segmentation map (this is the map which has details such as contour length, start and
        end positions of contours etc. for every contour).
        * Get number of contours (no_contours).
        * Get start and end positions of every contour.
        * Get the 'initial' and 'end' position, and their positional derivatives for every contour.

2) (i) for i = 1 -> no_contours
       {
         for j = i -> no_contours
         {
           if (contour j (start or end position) – contour j+1 (start or end position) < POS_THRESHOLD) &&
             if (contour j (derivative of start or end position) – contour j+1 (derivative of start or end position)
                 < DERIV_THRESHOLD)
           /* check the start and end position of every contour with the start and end position of every
              other contour, could use Mahalanobis distance for setting thresholds (see text) */
           {
              contour j will merge with contour j+1
              enter mergeable contour id, length, start, finish position etc. in a contour_merge_table
           }
         }
       }

3) Read the contour_merge_table and merge the contours which had passed the merge test in 2).
```

6.4 Temporal Tracking of Key Feature Points

In this section we discuss the process to extract key points from the object of interest and we also discuss the procedure to track them temporally.

6.4.1 Extracting Key Feature Points from Objects

In order to temporally track the object of interest, key points from the object are extracted to represent the object. The key point extraction method should ensure that only true corner points (or any clearly identifiable and definable points) are extracted. Extraction of multiple points within a small region should be avoided (eg: in a curved object, ideally only 1 point should be selected from the curved portion) for good tracking. Since contour grouping (discussed in the previous section) is based on an edge-map, it is desirable that key points should also be selected from the same edge map. Such a process will be efficient and will eliminate the requirement to employ a separate corner detection algorithm. Because of these limitations, we cannot effectively use any of the corner extraction algorithms mentioned in Chapter 3 (these calculate corner values directly from the raw image). Instead we have employed a method called the curvature scale space technique [134], which selects key points directly from an edge map efficiently. In the next section the curvature scale space technique is discussed in brief.

6.4.2 The Curvature Scale Space Algorithm (CSS)

The CSS technique is suitable for recovering invariant geometric features (curvature zero-crossing points and / or extrema) of a planar curve at multiple scales. To compute it, a curve Γ is first parameterised by the arc length parameter u:

$$\Gamma(u) = (x(u), y(u))$$

An evolved version Γ_σ of Γ can then be computed. Γ_σ is defined by [134]:

$$\Gamma_\sigma(u) = (\chi(u,\sigma), \gamma(u,\sigma)),$$

where

$$\chi(u,\sigma) = x(u) \otimes g(u,\sigma) \qquad \gamma(u,\sigma) = y(u) \otimes g(u,\sigma),$$

where $\otimes$ is the convolution operator and $g(u,\sigma)$ denotes a Gaussian of width σ (σ is also referred to as the *scale* parameter). The process of generating evolved versions of Γ as σ increases from zero to infinity is referred to as the evolution of Γ. This technique is suitable for removing noise from, and smoothing a planar curve as well as gradual simplification of its shape. In order to find curvature zero-crossings or extrema from evolved versions of the input curve, one needs to compute the curvature accurately and directly on an evolved version Γ_σ. Curvature κ on Γ_σ is given by [134]:

$$\kappa(u,\sigma) = \frac{\chi_u(u,\sigma)\gamma_{uu}(u,\sigma) - \chi_{uu}(u,\sigma)\gamma(u,\sigma)}{\left(\chi_u(u,\sigma)^2 + \gamma(u,\sigma)^2\right)^{1/2}}$$

where

$$\chi_u(u,\sigma) = x(u) \otimes g_u(u,\sigma) \qquad \chi_{uu}(u,\sigma) = x(u) \otimes g_{uu}(u,\sigma)$$

$$\gamma_u(u,\sigma) = y(u) \otimes g_u(u,\sigma) \qquad \gamma_{uu}(u,\sigma) = y(u) \otimes g_{uu}(u,\sigma)$$

6.4.3 CSS Key Point Detection Method

6.4.3.1 Brief Overview

The corners (key points) are defined as the local maxima of the absolute value of curvature. At a very fine scale, there exist many such maxima due to noise on the digital contour. As the scale is increased, the noise is smoothed away and only the maxima corresponding to the real corners remain. The CSS detection method finds the corners at these local maxima.

As the contour evolves, the actual locations of the corners change. If the detection is achieved at a large scale the localisation of the corners may be poor. To overcome this problem, local tracking is introduced in the detection. The corners are located at a high scale σ_{high}, assuring that the corner detection is not affected by noise. σ is then reduced and the same corner points are examined at lower scales. As a result, location of corners may be updated. This is continued until the scale is very low and the operation is very local. This improves localisation and the computational cost is low, as curvature values at scales lower than σ_{high} do not need to be computed at every contour point but only in a small neighbourhood of the detected corners.

There are local maxima on the evolved contours due to rounded corners or noise. These can be removed by introducing a threshold value t. The curvature of a sharp corner is higher than that of a rounded corner. The final stage to the candidate corner declaration is that each local maximum of the curvature is compared to its two neighbouring local minima. The curvature of a corner point should be double the curvature of a neighbouring extremum. This is necessary since if the contour is continuous and round, the curvature values can be well above the threshold value t and false corners may be declared.

6.4.3.2 CSS Detection Process

The CSS key point detection process can be given by the following steps:

1. Utilise an Edge detector (such as Canny [40] or Boie-Cox [31] etc.) to extract edges from the original image.
2. Extract the edge contours from the edge image:
   - Fill gaps in the edge contours
   - Find the T-junctions and mark them as T-corners
3. Compute the curvature at highest scale σ_{high} and determine the corner candidates by comparing the maxima of curvature to the threshold t and the neighbouring minima.
4. Track the corners to the lowest scale to improve localisation.
5. Compare the T-corners to the corners found using the curvature procedure, and remove corners which are very close.

The details of the CSS process can be found in [134].

6.4.4 Tracking Point Features

The MHT-IMM (MHT coupled with a multiple model Kalman filter, as discussed in chapter 4) algorithm can be applied for tracking key point features through an image sequence. The procedure for this was analysed in detail in chapter 4. The measurements for the tracking filter in this case will be the key features extracted (from and near the object of interest) from every frame of a given image sequence (key points are searched within a region of interest surrounding the estimated object centroid). The object centroid position is initially calculated in the first frame by taking the mean of the sum of object key point positions. In the subsequent frames the object centroid is estimated using the MHT-IMM tracker. The extracted measurements are then matched to predictions based on the *Mahalanobis* distance.

The advantage in using the key point tracking algorithm is that we can verify each of the temporally translated key points on the object (selected) against the likely contour of that object in every frame. By doing so, we examine to see whether the object as a whole is tracked correctly (the process for doing this is explained in the next section). In the next section we give the procedure involved in combining the point feature tracking algorithm and the contour grouping algorithm for object tracking.

6.5 Object Tracking

This section shows how the contour grouping and key point feature tracking procedures combined can be applied for object tracking in image sequences. One cycle of the algorithm recursion is displayed in Fig. (6.4).

The object tracking principle underlying this algorithm is shown in Fig. 6.5. For every frame in an image sequence we first apply the contour segmentation algorithm. This process will group segments of edges that are likely to be from the same object. The result of such a process applied to our test sequences are given in Figures 6.6(a) – 6.9(a). The procedure as seen from these figures, fail at high curvature contour regions, or is unable to bridge a gap in edgels extracted. As a result, contours from the same object are often broken or separated. To overcome this limitation we applied the contour merging algorithm, which resulted with recognisable object contours (Fig. 6.6b – 6.9b).

Once the object contours are categorised separately, we can now track a selected object (selection of object can be automated by using a snake type algorithm (eg: Gsnake [120, 121]) or any other suitable algorithm) from the initial frame through the sequence. To track the selected object, we first select some key features (points) from the object (these are selected using the edge map information and then applying the CSS algorithm) as discussed section (6.4).

The key features of the object are extracted in every frame and the object centroid calculated (this is the mean position of the sum of key points of the object contour). The key points (and the centroid) from the first frame are now tracked through the sequence using the MHT-IMM algorithm (as discussed in chapter 4). The tracking process is achieved by predicting the object centroid position in the following frame, and then searching a region of interest surrounding the centroid to look for the key points, this process is followed by matching the key points to a grouped object contour within that region. This procedure will provide trajectories for every key point of the object. Each trajectory point is validated against a grouped contour in each frame. By imposing a distance threshold between the tracked key points and the key points on the segmented contour (in each frame), we can verify whether the points have been tracked to an acceptable level of precision. If an acceptable number of key points tracked are identified to lie on or near the object contour (that is passing the threshold test) in each of the frames, we conclude that the object has been tracked successfully. If a key point fails the threshold test, then that point will not be considered as part of that feature trajectory any further.

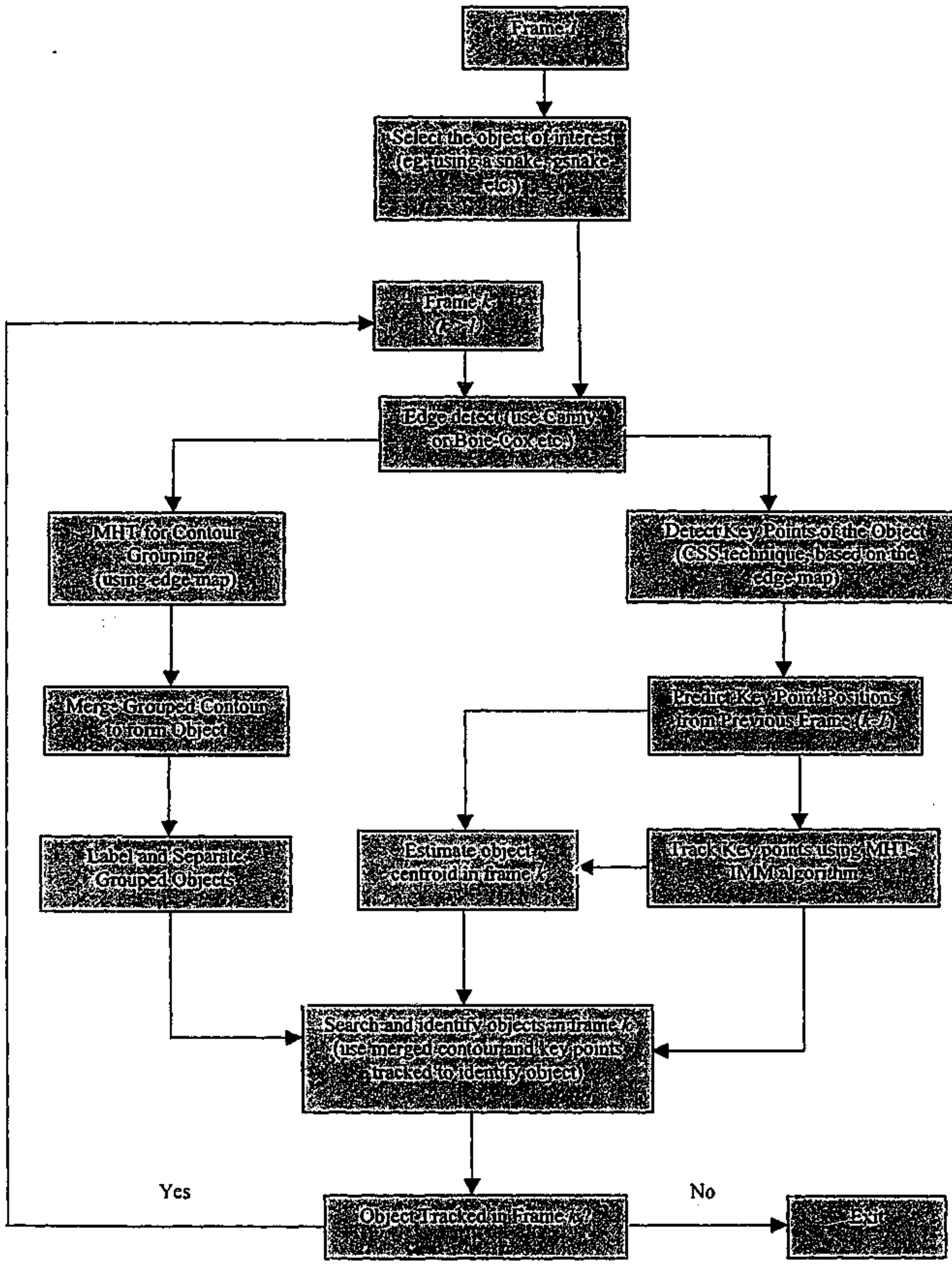

Frame 1
Select the object of interest (eg. using a snake, gsnake etc.)
Frame k (k>1)
Edge detect (use Canny or Boie-Cox etc.)
MHT for Contour Grouping (using edge map)
Detect Key Points of the Object (CSS technique, based on the edge map)
Merge Grouped Contour to form Object
Predict Key Point Positions from Previous Frame (k-1)
Label and Separate Grouped Objects
Estimate object centroid in frame k
Track Key points using MHT-IMM algorithm
Search and identify objects in frame k (use merged contour and key points tracked to identify object)
Yes
No
Object Tracked in Frame k?
Exit

Figure 6.4: Overview (1 cycle) of the object tracker proposed

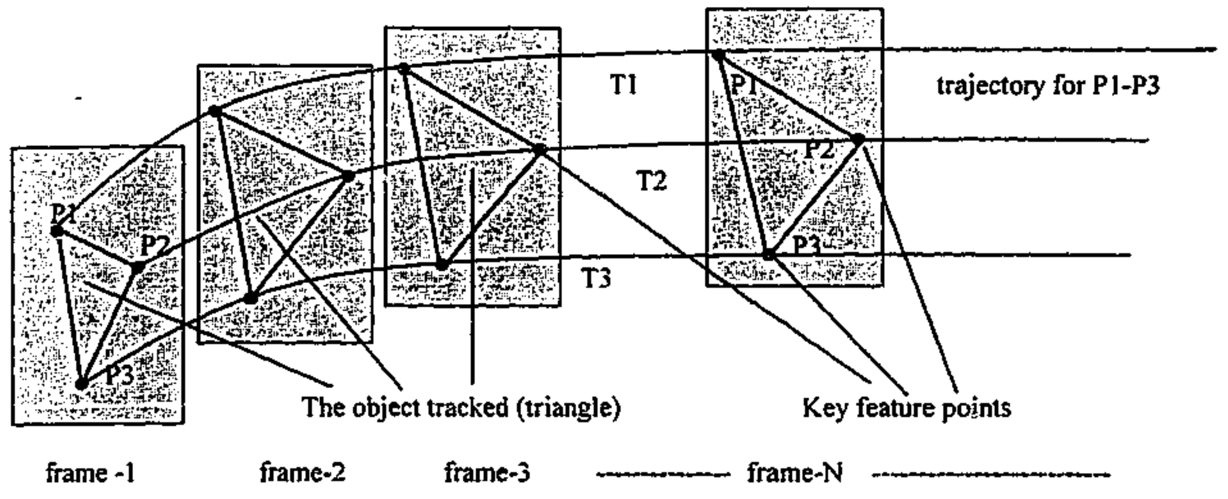


Figure 6.5: A graphical illustration of the object tracking principle. The object (triangle) contour is formed by the key feature points P1, P2, P3 (black blobs in frame-1). The object formed in every frame is achieved using contour segmentation coupled with a contour merging technique (section 6.3). T1, T2, T3 are the trajectories of the key points. The trajectories are obtained by using the MHT-IMM tracking algorithm (section 6.4). If the key points tracked (trajectories T1-T3) in every frame lie on or near the contour (triangle) in each frame, then we have the object itself tracked through the image sequence (up to frame N).

6.6 Results

Image Sequence (frames length)	**Number of key points selected to be tracked**	**Attempted number** of points tracked (& percentage)		**Number of features tracked for more than 2/3's of the seq. length** (& percentage)	
UMASS Lab (11)	8	8	100 %	8	100 %
PUMA (30)	4	4	100 %	4	100 %
Indoor cones (8)	3	3	100 %	3	100 %
Outdoor cones (20)	12	10	83.3%	10	100 %

Table 6.1: Object tracking statistics for the 4 test image sequences considered.

The 4 sequences considered gives a variety of scenarios to test our algorithm. In all 4 cases the tracking results are promising (see Figures 6.6 – 6.9). Table 6.1 provides quantitative performance values for the object tracker. For the UMASS lab sequence 100% of the key-points selected as forming the object (posters) in the first frame are successfully tracked for the entire sequence length. Similar observations can be made for the PUMA and the indoor cone sequences. Finally a multiple object example is demonstrated. For the outdoor cone sequence, 4 cones are considered as part of an object. As the result suggests, 10 out of the 12 key corner features are tracked successfully for more than 2/3's of the sequence length.

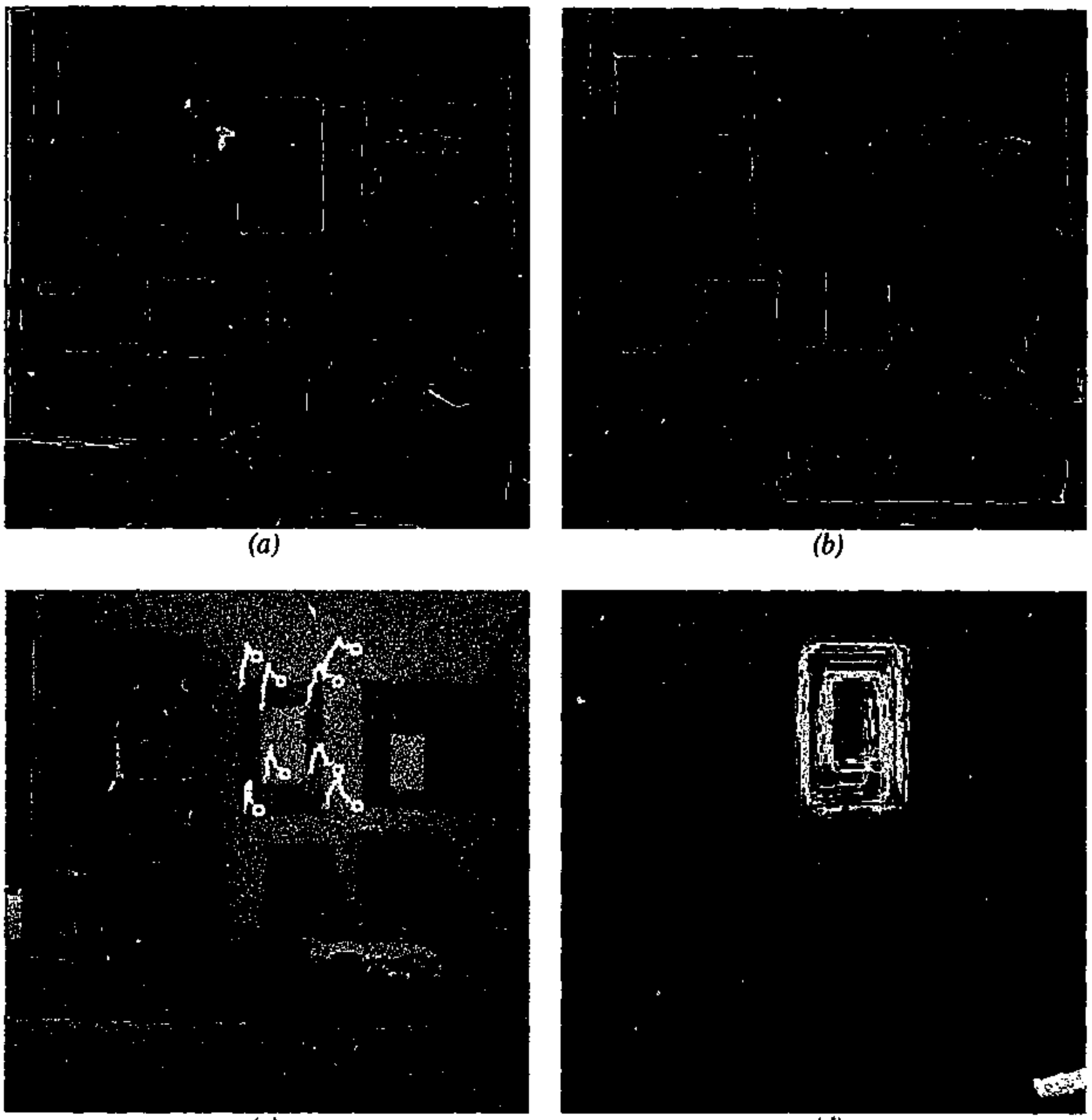
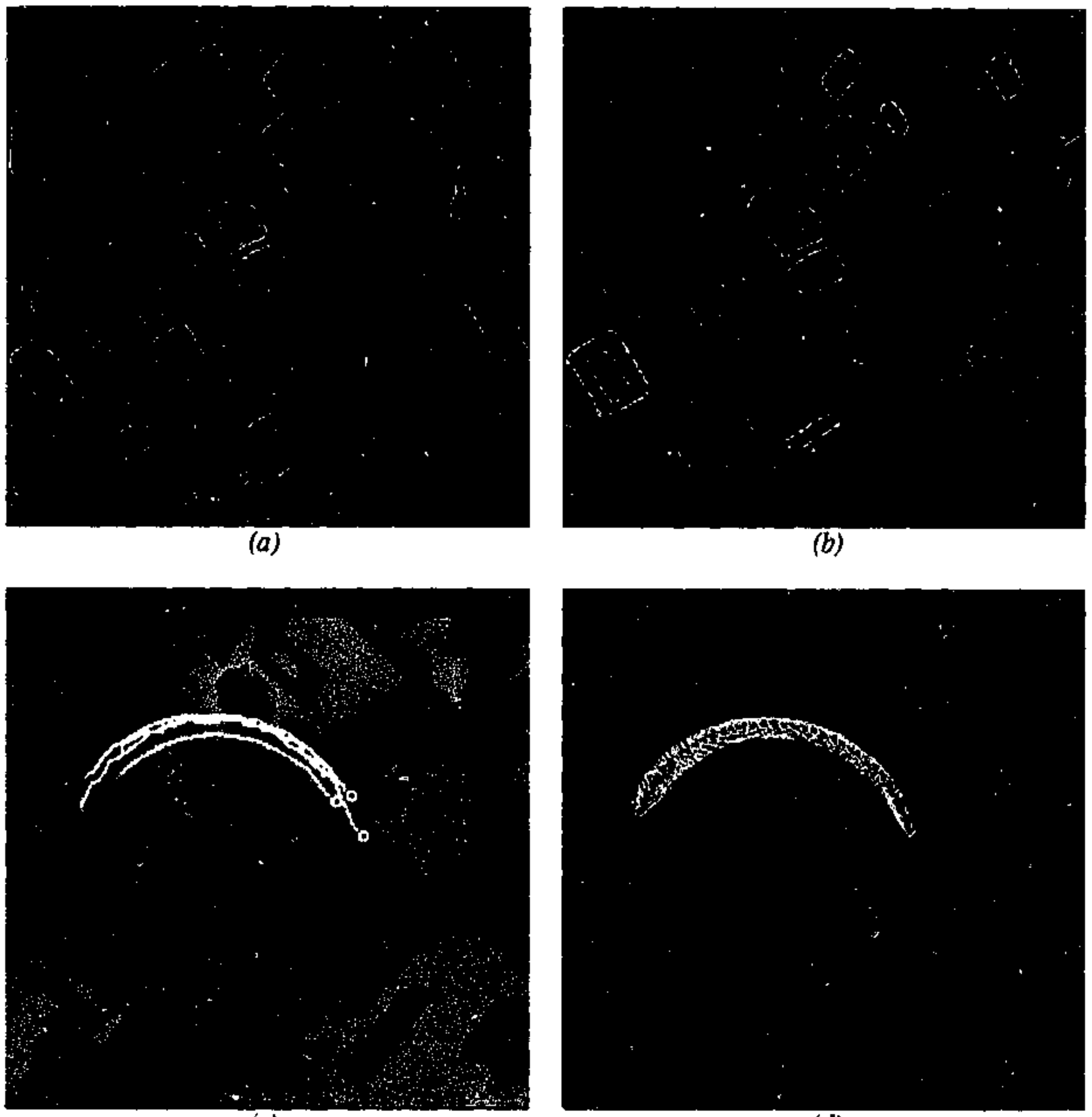

(a) (b) (c) (d)

Figure 6.6: UMASS lab sequence result. (a) Contours grouped by applying the contour segmentation algorithm based on the edge map obtained (one frame of the sequence is displayed). Each segmented contour (grouped edges) is shown with a different color. (b) Result after the application of segment merging algorithm (observe that the segments that are identified as forming the same object are merged together in most instances). (c) The trajectory of the 8 key points by applying the MHT-IMM algorithm. The 'x' shows the start of the trajectory while the little white circle indicates the end of trajectory. (d) The identified object trajectory (poster). The white contours (identified as belonging to the same object in each frame) are superimposed on the first frame of the sequence to show the motion of the object.


(a)

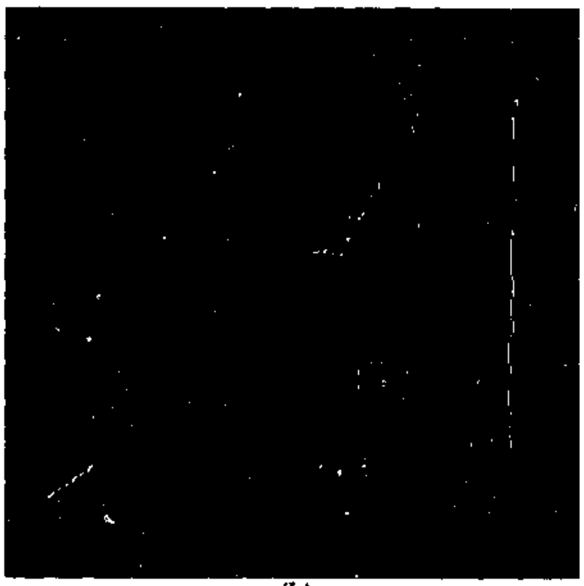
(b)

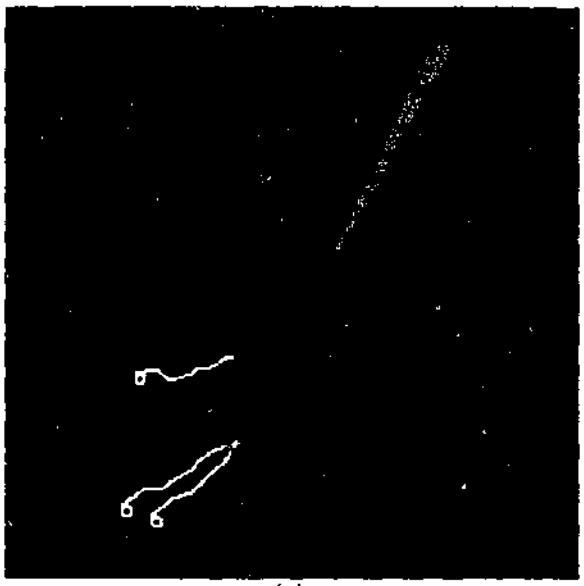
(c)

(d)

Figure 6.7: PUMA lab sequence result. (a) Contours grouped by applying the contour segmentation algorithm based on the edge map obtained (one frame of the sequence is displayed). Each segmented contour (grouped edges) is shown with a different color. (b) Result after the application of segment merging algorithm (observe that the segments that are identified as forming the same object are merged together in most instances). (c) The trajectory of the 4 key points by applying the MHT-IMM algorithm. The 'x' shows the start of the trajectory while the little white circle indicates the end of trajectory. (d) The identified object trajectory (window). The white contours (identified as belonging to the same object in each frame) are superimposed on the first frame of the sequence to show the motion of the object.

(a)


(b)


(c)

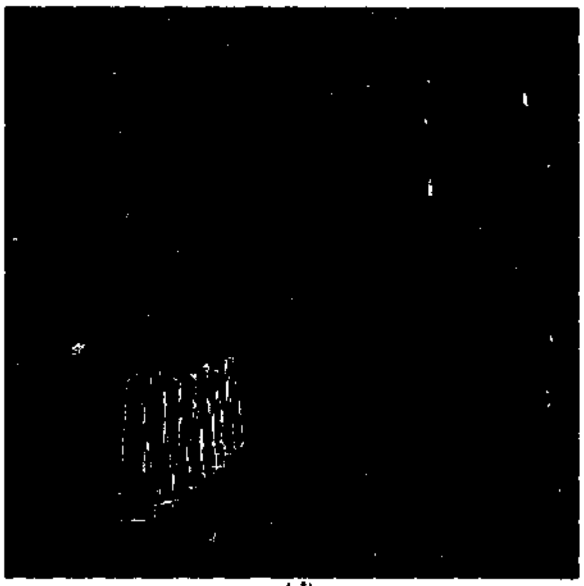

(d)

Figure 6.8: Indoor cone sequence result. (a) Contours grouped by applying the contour segmentation algorithm based on the edge map obtained (one frame of the sequence is displayed). Each segmented contour (grouped edges) is shown with a different color. (b) Result after the application of segment merging algorithm (observe that the segments that are identified as forming the same object are merged together in most instances). (c) The trajectory of the 3 key points by applying the MHT-IMM algorithm. The 'x' shows the start of the trajectory while the little white circle indicates the end of trajectory. (d) The identified object trajectory (cone). The white contours (identified as belonging to the same object in each frame) are superimposed on the first frame of the sequence to show the motion of the object.

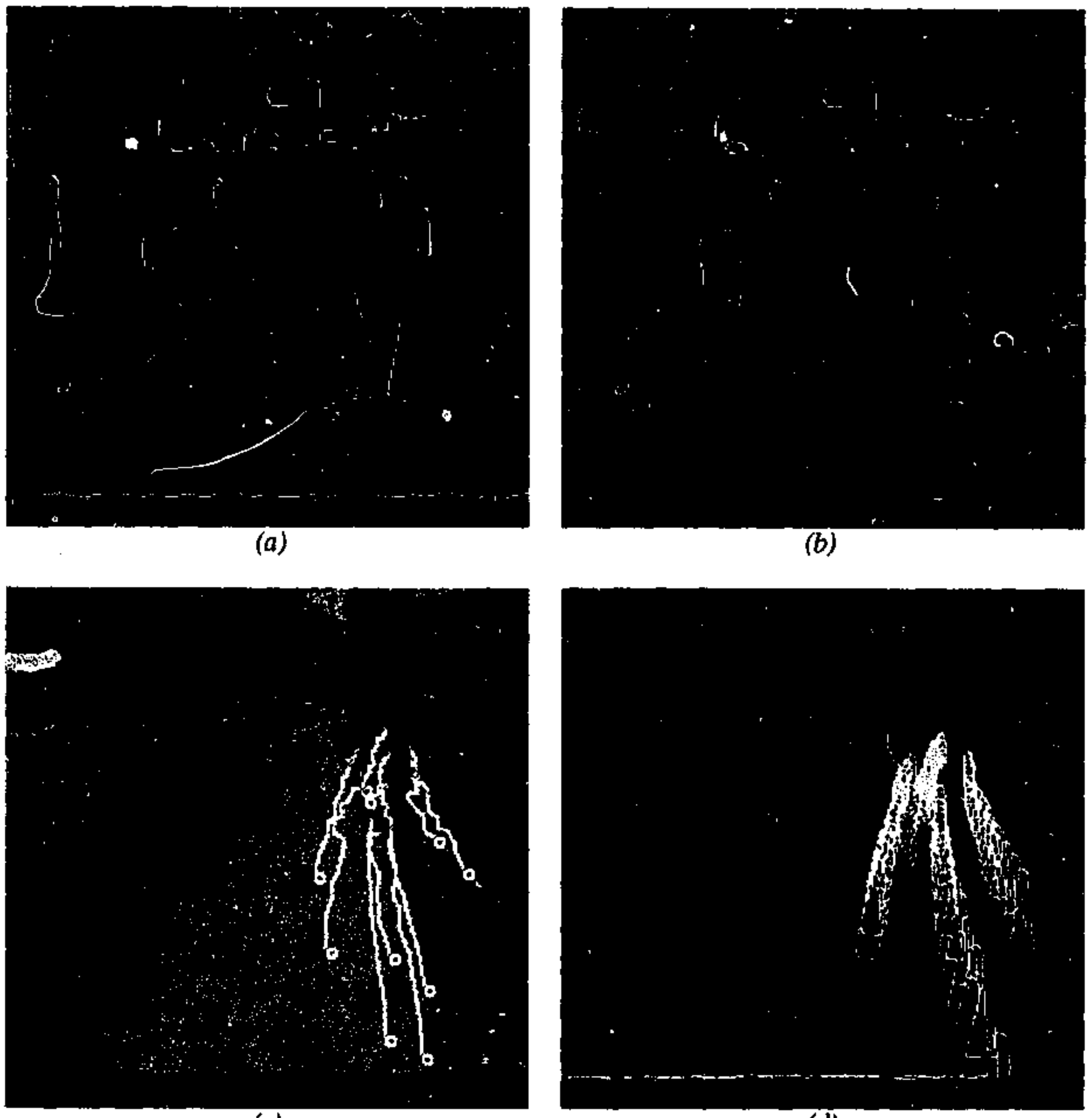

(a) (b) (c) (d)

Figure 6.9: Outdoor cone sequence result. (a) Contours grouped by applying the contour segmentation algorithm based on the edge map obtained (one frame of the sequence is displayed). Each segmented contour (grouped edges) is shown with a different color. (b) Result after the application of segment merging algorithm (observe that the segments that are identified as forming the same object are merged together in most instances). (c) The trajectory of the 10 key points by applying the MHT-IMM algorithm. The 'x' shows the start of the trajectory while the little white circle indicates the end of trajectory. (d) The identified object trajectory (4 cones). The white contours (identified as belonging to the same object in each frame) are superimposed on the first frame of the sequence to show the motion of the objects.

6.7 Discussion

Figure 6.6(a) – 6.9(a) shows the result of applying the contour segmentation algorithm. It can be seen that the segmentation algorithm fails to group segments of the same edge around sharp curves. Since the algorithm scans the edge image by "walking" along the contours, it may encounter a new contour at any point along its length. When tracking begins in the interior of a curve, it is usually partitioned, erroneously into two or more segments sharing common boundary points. As a result of this, contours belonging to the same object can be grouped as separate objects. To overcome this limitation we applied the contour-merging algorithm (as described in section 6.4) which provided better results (Fig. 6.6(b) - 6.9(b)). It can be clearly seen that most of the segments belonging to the same object have now been grouped together successfully (the quality of the segmentation also depends on the thresholds that are used for both algorithms [57-59]).

For the PUMA sequence, the window on the top left corner of frame 1 (see Figure 6.7) was tracked through the sequence. The result of the tracking is given in Figure 6.7(d) and the corresponding trajectories of the key points are given in Figure 6.7(c). From visual inspection the results are promising. Similar results are observed for the UMASS lab sequence (Fig. 6.6(c,d)), despite the short irregular translation of the posters (top right corner of frame 1). The qualitative results are supported by quantitative results presented in Table 6.1.

For the indoor cone sequence, a cone on the left side (middle) is tracked. As can be observed from the results (6.8a,b), the cone is identified as a separate object and the 3 key points are tracked successfully. The qualitative result is shown in Fig. (6.8). Figure (6.9) shows the result of the outdoor cone sequence. In this case, multiple objects are tracked (4 cones on the right). Each cone is treated as a separate entity, while all 4 cones combine to form a 'grouped-object'. Each of the key points from the 4 cones are tracked and matched to the segmented shape (the 4 cones). Apart from the last frame, where the cone in the front gets segmented with the road, 83% of the key points have been tracked correctly (see Table (6.2)), thus successfully tracking the 4 cones.

6.8 Conclusion

In this chapter we have shown how the multiple hypothesis technique can be used for rigid object tracking in image sequences. The contour of object tracked is achieved by first applying the MHT approach to group segments of the same object. This process is followed by applying the contour merging algorithm to identify recognisable object contours. Then by selecting key point features of this object, temporal tracking (matching) of key points is achieved by using the MHT again. The validity of the trajectory of the key points is verified by inspecting whether the key points were lying on or near the contour of the tracked object (searched within a region of interest). The results are promising for objects that are not occluded and can be recognised clearly in every frame.

One of the main drawbacks of the system is that the contour grouping process can break down due to occlusion of the object being tracked. The MHT can predict possible trajectory for the key points despite the occlusion [58] and thus retain the trajectory (as shown in chapter 4). But the contour segmentation and grouping process will fail, as it looks only at the edge map to group contours. As a result the object tracker fails in its primary purpose. The tracking process presented can also fail for deforming objects. This is because the key point tracking phase will not be robust enough to track unexpected deformation of object contours.

Recognising and tracking objects using point features as presented in this chapter is possible for relatively simple objects (as demonstrated in the results). For complex objects the process is inefficient, and can lead to errors in object identification and tracking. A more versatile method of object tracking will require an object contour to be represented using a parameterised curve, such as using Snakes [113, 120], Deformable templates [27], or using B-splines [12, 61, 23, 24]. In the next chapter we provide an efficient algorithm to track multiple objects (contours of multiple deformable objects) in extended image sequences. The algorithm is also capable of tracking objects that move with variable motion.

Chapter 7

Contour Tracking with Automatic Motion Model Switching

Abstract

In this chapter we present an efficient contour- tracking algorithm which can track 2D silhouettes of objects in extended image sequences. We demonstrate the ability of the tracker by tracking highly deformable contours (such as people walking) captured by a static camera. We represent contours (silhouette) of moving objects by using a cubic B-spline. The tracking algorithm is based on tracking a lower dimensional shape space (as opposed to tracking in spline space). Tracking the lower dimensional space has proved to be fast and efficient. The tracker is also coupled with an automatic motion-model switching algorithm (IMM), which makes the tracker robust and reliable when the object of interest is moving with multiple motion. The model based technique provided is capable of tracking rigid and non-rigid object contours with good tracking accuracy [184].

7.1 Introduction

Most contour tracking methods reported in the literature assume that the changes of shape of objects, between frames, are very small. They also assume that the object of interest is moving with a constant motion model (examples in Chapter 2). Provided these conditions are satisfied the reported algorithms are claimed to perform well for their respective applications. However, in reality one cannot make such assumptions for applications such as tracking a walking/running person. The motion of a walking person can be variable and also the inter-frame shape changes cannot be assumed to be 'very small', particularly for objects captured by a camera at low frame rates (eg: 10-15 frames/sec). Our contribution in this chapter primarily addresses these issues. We provide a shape space decomposition technique (for highly deformable object contours) which makes the contour tracking process easier and robust. The entire tracking mechanism evolves around the decomposed shape space, thus reducing computational burden to a large extent. The contour tracker introduced can also cope with larger inter-frame displacements and can automatically switch motion models according to the motion of the object of interest.

This chapter is organized as follows: Section 7.2 provides an overview of general contour representation schemes, while section 7.3 focuses specifically on using B-splines for contour representation. Section 7.4 describes a rigid body shape representation method using B-splines, and section 7.5 provides a method for non-rigid shape representation using a 'shape training-set' analysis. Section 7.6 gives an efficient shape space decomposition technique, which enables the tracking process to be robust and efficient. Section 7.7 gives an overall description of the dynamic system employed. Section 7.8 outlines the tracking filter, describing the measurement and prediction models used. Section 7.9 extends the tracking analysis to cope with automatic motion model switching. Section 7.10 describes the object detection and separation process employed (including occlusion handling). Section 7.11 gives the main results and provides a brief discussion. Finally section 7.12 gives the conclusion and possible future research directions.

7.2 Contour Representation

There are several ways of representing a 2D contour of an object (rigid and non-rigid). Among the many representations, edge based methods [97, 127], snakes [113], active shape models [204], and B-spline methods have been used for contour tracking as discussed in chapter 2.

Edge based methods primarily depend on using some reliable edge detection algorithm [40, 97]. Once the edges are detected, the different objects are separated using some segmentation scheme [174]. A common problem with edge based methods (to represent objects) is that edges extracted are not continuous around corners (broken or split) and the quality of edges are limited by threshold values used in most edge detection algorithms. The discontinuous nature of edges makes these approaches less desirable for contour (object) tracking applications [24].

Snakes on the other hand are based on an elastic-energy minimization principle [113] (also discussed in chapter 2), the result of which attract snakes to some feature of interest such as edges, valleys or ridges [113]. Snakes are also not perfectly smooth since the energy functions employed in snake algorithms have first and second order differentials, and for any practical purpose they have to be approximated by some finite difference method which makes them less smooth [24].

The prior shape constraints implicit in a snake model are soft in general: encouraging rather than enforcing a particular class of favored shapes, and these favored shapes have rather limited variety. To overcome such limitations a parametric shape model with relatively few degrees of freedom can be employed. The resulting template is known as 'deformable template' [204] (discussed in chapter

2). One down side to this approach is that the template is parameterized to fit some apriori shape using a number of geometric parameters which can vary non-linearly with a parametric vector [24]. The number of parameters used depends on the complexity of the shape of interest. For any real time temporal shape tracking, such methods, in general, are not efficient.

The use of B-splines for curve representation has been shown to overcome most of the limitations caused by using other contour representation methods. B-splines are a parametric curve passing through (or nearly passing through) N control points. B-splines in general maintain a degree of smoothness and continuity which is essential for object representation in tracking applications. B-splines also provide computational advantages over other representations [24, 27, 60]. Particularly for tracking purposes, prior knowledge can be incorporated into a tracker by an elastic coupling with a template B-spline [25]. This persistent template mechanism improves stability by incorporating shape memory: restricting the prior distribution of the contour shape. In this chapter we will use cubic B-splines to represent 2D contours (also referred as silhouettes in this chapter).

7.3 Contour Representation using B-splines

B-splines are an efficient representation of curves with limited degrees of freedom. They have a number of desirable properties including the fact that they obey the constraint that all the curve lies within the convex hull of control points. They can also be made continuous even at break points, up to a certain desired order. Large image features may be represented by a B-spline using a few control points, rather than as a list of pixels (this was the case in chapter 6), and this reduction in degrees of freedom enables real-time (or near real-time) implementation of tracking algorithms feasible.

The following notation for the contour is used throughout this chapter. The continuous curve parameter is s, which in general varies over $0 \leq s \leq N$ along the length of the curve. For simplicity, we assume that each span (between control points) has unit length in s. Hence the n-th span is defined over the interval $n-1 \leq s \leq n$. The continuous curve for a spline of order d is then given by [23, 60]:

$$\mathbf{Q}(s,t) = \mathbf{H}(s)\mathbf{Q}(t), \tag{7.1}$$

where $\mathbf{Q}(t)$ is the vector of control points of size $2N$x1 (and t is the instant of time in which the control points are considered), ie.

$$\mathbf{Q}(t) = [x_1(t),\ldots,x_N(t) \quad y_1(t),\ldots,y_N(t)]^T$$
$$= \begin{bmatrix}\mathbf{X}(t)\\ \mathbf{Y}(t)\end{bmatrix}$$

and

$$\mathbf{H}(s) = \begin{bmatrix} h(s) & 0 \\ 0 & h(s)\end{bmatrix}, \text{ where, } h(s+n-1) = \mathbf{s}^T B_n G_n, \qquad 0 \le s \le 1, \;\; 0 \le n \le N$$

In this notation:

- B_n is the shape matrix (B-spline basis matrix) for the n-th span, derived from the number of knots at break-points around the span as in [7].
- G_n describes the controllability of each span by selecting from the global control point vector Q(t) only those control points which control the n-th span, thus G_n is a shifted identity matrix augmented by zero columns [23].
- $\mathbf{s} = (1, \;\; s, \;\; \ldots \;\; , s^d)^T$ contains the polynomial terms in the spline parameter s up to the spline order d (eg: d=3 for cubic B-splines).

For the rest of this chapter the subscript t is omitted for notational convenience.

7.4 Rigid Contour Shape Representation for Tracking

Blake et. al.'s [22, 23] pioneering work in proposing B-splines to represent contour shapes has made contour tracking in general much efficient and reliable than the better know "snake" approach [113]. Any contour can be approximated by N control points using a B-spline of order d (as discussed in the previous section). A tracker now could conceivably be designed to allow arbitrary variations in control point positions over time. This would allow maximum flexibility in deforming to accommodate moving shapes. However, particularly for complex shapes requiring many control points to describe them, this is known to lead to instability in tracking [25]. The instability occurs when features are temporarily obscured and the tracker is 'bumped out' of its steady state. The more complex the shape to be tracked, the worse is the instability that occurs when the lock to the shape is lost.

Fortunately, it is not necessary to allow so much freedom for the control points. An image of a moving hand, for instance, provided the fingers are not flexing, is a rigid, approximately planar shape. Provided perspective effects are not too strong, a good approximation to the curve shape as it changes over time can therefore be obtained by specifying **Q**, a linear vector valued function of the B-spline coordinates (**X**,**Y**). The Q-parameterization of the curve embodies the reduced degrees of freedom for motion, which vary online, leaving intact the full set (**X**,**Y**) of geometrical parameters to do justice to the detail of complex shapes and to be varied offline only [25]. Such a representation can be achieved by reducing the number of degrees of freedom to a lower dimension and preserving the original base-shape of object. As an example, for a planar affine transform of a template shape, the degrees of freedom are six. That is, the Degrees of Freedom are limited to: translation (2 DOF), scaling (2 DOF), rotation (1 DOF) and shear (1 DOF).

In mathematical terms: if a known template spline $\mathbf{Q}_0 = (\mathbf{X}_0, \mathbf{Y}_0)^T$ is allowed to undergo planar affine transformation, the resulting vector spline can be expressed by [12, 50, 51]:

$$\mathbf{Q} = \mathbf{D}(a_x, a_y)\mathbf{Q}_0 + \mathbf{U} \tag{7.2}$$

where **U** ($2N$ x 1) represents the translation (2 DOF) vector of control points, while **D** ($2N$ x $2N$) represents an alignment matrix (scaling and rotation) and these are given by,

$$\mathbf{D} = \begin{bmatrix} \Phi & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \Phi \end{bmatrix} \quad \text{and} \quad \mathbf{U} = [\underbrace{u_x, u_x, \ldots, u_y, u_y}_{N\ times}]^T$$

where

$$\Phi = \begin{bmatrix} f_x \cos\vartheta & -f_y \sin\vartheta \\ f_y \sin\vartheta & f_x \cos\vartheta \end{bmatrix} = \begin{bmatrix} a_x & -a_y \\ a_y & a_x \end{bmatrix}$$

Φ is the alignment matrix for each control point. The scaling factor f_x, f_y (in the x and y direction respectively) and rotation ϑ are in relation to a template shape.

The planar affine transformation of the template (equation (7.2)) can be expressed as [24],

$$\mathbf{Q} = \mathbf{W}\mathbf{T} + \mathbf{Q}_0 \tag{7.3}$$

which is a linear mapping between a spline control point vector and a lower dimensional vector space T. The lower dimensional space is also referred to as a "shape space". The shape matrix W for the planar affine case is given by [24],

$$\mathbf{W} = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{X}_0 & \mathbf{0} & \mathbf{0} & \mathbf{Y}_0 \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{Y}_0 & \mathbf{X}_0 & \mathbf{0} \end{pmatrix} \tag{7.4}$$

where the 1st two columns correspond to translation (where $\mathbf{1} = [1,1,...,1]^T$, $\mathbf{0} = [0,0,...,0]^T$), and the last 4 columns correspond to changes in scaling, rotation and shear.

In equation (7.3), the elements of **T** acts as weights on the columns of **W**. The interpretation of **T** in terms of planar affine transform can be explained as follows.

$$\mathbf{T} = [u_x, u_y, (f_x \cos\vartheta - 1), (f_y \cos\vartheta - 1), f_x \sin\vartheta, -f_y \sin\vartheta]^T .$$

Conversely, given a known spline **Q**, the shape space vector **T** can be recovered by using the following expression [25]:

$$\mathbf{T} = \mathbf{V}[\mathbf{Q} - \mathbf{Q}_0] \tag{7.5}$$

where

$\mathbf{V} = \left(\mathbf{W}^T \mathbf{J} \mathbf{W}\right)^{-1} \mathbf{W}^T \mathbf{J}$, where **J** is the '*metric matrix*' defined in [24, 60] and is given by,

$$\begin{aligned} \mathbf{J} &= \int_0^N \mathbf{H}^T(s)\mathbf{H}(s)ds \\ &= \begin{bmatrix} \beta & 0 \\ 0 & \beta \end{bmatrix} \quad \textit{where} \quad \beta = \sum_{n=1}^{N} G_n^T B_n^T \left[\int_0^1 \mathbf{s}\mathbf{s}^T ds \right] B_n G_n \end{aligned}$$

The affine transformation of a template limits the shape matrix **W** (and **T**) to only rigid body shape changes (6 DOF). For deformable (non-rigid) object tracking, **W** has to be extended to accommodate for non-rigid shape changes. We provide a method to efficiently extend the matrix **W**, based on the work reported in [51, 52, 14]. The details of which are explained in the next section.

7.5 Deformable Contour Shape Representation for Tracking

There are several methods reported in the literature to account for the deformation of non-rigid objects (some methods were briefly discussed in chapter 2). Blake et. al. proposed a 'key frame' techniques [24] where known poses of shape variation can be accommodated into the shape matrix. A potential problem in this method is that a shape can be captured in several poses, and to accommodate all different pose variation becomes less efficient. Another method is to capture a short sequence of the object in motion and then learn the dynamic parameters from this test sequence and apply them to the actual tracker. Though efficient, such a 'learning tracker' can be useful only to a narrow variety of applications. It also presents the problem of capturing a short image sequence prior to tracking the object of interest every time [25]. Other methods include articulated motion models using kinematic motion models [151]. Such a method causes non linearity in the tracking system, thus requiring non linear dynamic systems modeling. Then there are the statistical models such as learning shape variation from a prior distribution of known shapes [12, 14, 50, 51]. These techniques primarily employ a Principal Component Analysis (PCA), which reveals the most significant variations of a given training set (against a mean shape) in a lower dimensional subspace. We adapt this method to account for deformable changes of objects because of its generality.

7.5.1 Shape Training Analysis

Cootes et. al. [50, 51] provided a 'shape training' method termed Linear Point Distribution Model (LPDM) where a training set of varied shapes (different poses of a similar objects) are analyzed (mentioned in chapter 2). Each training shape (in spline space) is represented as Q (as in Eq. 7.2). Where each point is the position of the i-th 'landmark point' (specific point) on the training shape. The training shapes are then aligned (scaled, rotated, and translated to a mean shape or any other standard shape size) using a Generalized Procrustes Analysis technique [10]. The process is in fact similar to using a weighted least square method to align each shape to a mean shape. The weights are chosen so that more significance is given to more 'stable' landmark points (clearly identifiable points). This process of marking landmark points for each shape manually is laborious. To overcome this difficulty Baumberg et. al. [11-14] proposed an automated method of labeling points by placing N B-spline control points along the contour of interest starting from a reference point and placing the points in a clock-wise direction. The N control points are approximately equally spaced along the contour. Fig. 7.1 shows some example shapes of walking persons (as part of a training set) captured in different poses and Fig. 7.2 provides a scheme to automatically generate a training set (A process that Baumberg et al. followed [10]).

Once the shapes are aligned and the control points assigned, then a statistical analysis can be performed on the training shape set. This process results in a mean shape-vector $\overline{\mathbf{Q}}$ and a set of aligned training shape vectors $\mathbf{Q}_k$ (k = *1,...,M*). In mathematical terms, given the training data $\mathbf{Q}_1,\ldots,\mathbf{Q}_M$, then the mean and covariance of the training set is given by the following equations respectively.

$$\overline{\mathbf{Q}} = \frac{1}{M}\sum_{k=1}^{M}\mathbf{Q}_k \qquad \mathrm{S} = \frac{1}{M}\sum_{k=1}^{M}\left(\mathbf{Q}_k - \overline{\mathbf{Q}}\right)\left(\mathbf{Q}_k - \overline{\mathbf{Q}}\right)^T \tag{7.6}$$

Finding the m significant eigenvectors (m << M) corresponding to m most significant eigenvalues ($\lambda_0 \geq \lambda_1 \geq \ldots \geq \lambda_{m-1} \geq 0$) of **SJ** will give the m most significant mode of variation of the training set (the reason for using **SJ** instead of S is given in Appendix D, and also explained in [10]). Therefore, any shape in the entire training set sequence can be approximately represented by the following equation.

$$\mathbf{Q} = \overline{\mathbf{Q}} + \mathbf{Pb} \tag{7.7}$$

where P is a $2N$ x m matrix whose columns are the m most significant eigenvectors of **SJ** and $\mathbf{b} = [b_0,\ldots,b_{m-1}]^T$ is a shape parameter vector with m coefficients, N is the number of control points representing the contour. By varying the shape parameters within suitable limits, different feasible shapes can be generated [49, 11]. Explicitly, for the i-th mode shape-vectors, $\mathrm{Q}^{(j)}$ are calculated using,

$$\mathbf{Q}^{(j)} = \overline{\mathbf{Q}} + step\left(\frac{j}{\sqrt{\lambda_i}}\right)\mathbf{e}_i \tag{7.8}$$

where j varies between $-k$ and k (eg: -2, -1, 0, 1, 2) and *step* is a suitable step size in standard deviations. $\mathbf{e}_i$ is the eigenvector for the i-th mode. Fig. 7.3 shows an example of shape variation for the first 10 eigenvetors of a training set (containing walking people). It is quite evident that only the first few eigenvectors provide the most significant shape variation of the training set (against the mean shape).

Conversely, given an aligned shape vector $\mathbf{Q}'$, the minimum least squares approximation to the shape in the model space is given by a linear projection,

$$\mathbf{b} = \mathbf{P}^T(\mathbf{Q}' - \overline{\mathbf{Q}}) \tag{7.9}$$

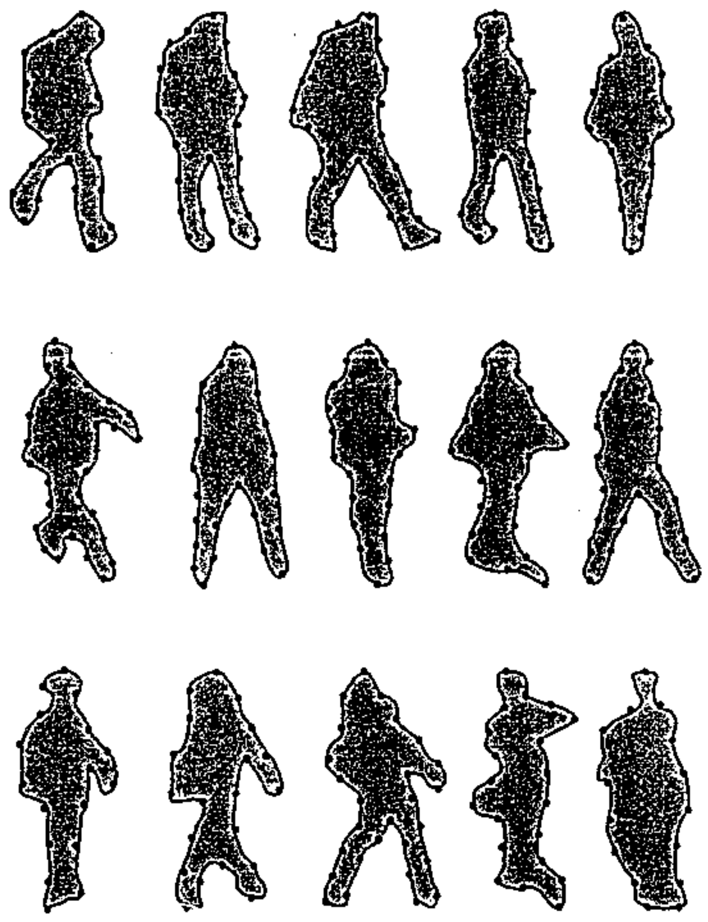


Figure 7.1: Examples of some training shape vectors of a pedestrian sequence.

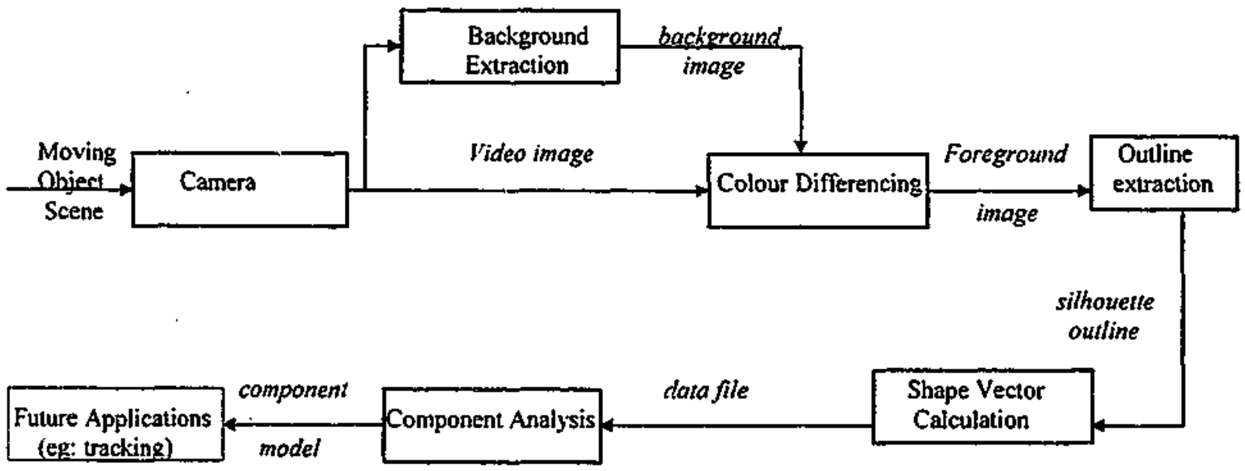


Figure 7.2: Overview of a system to obtain a 'shape training set' [10].

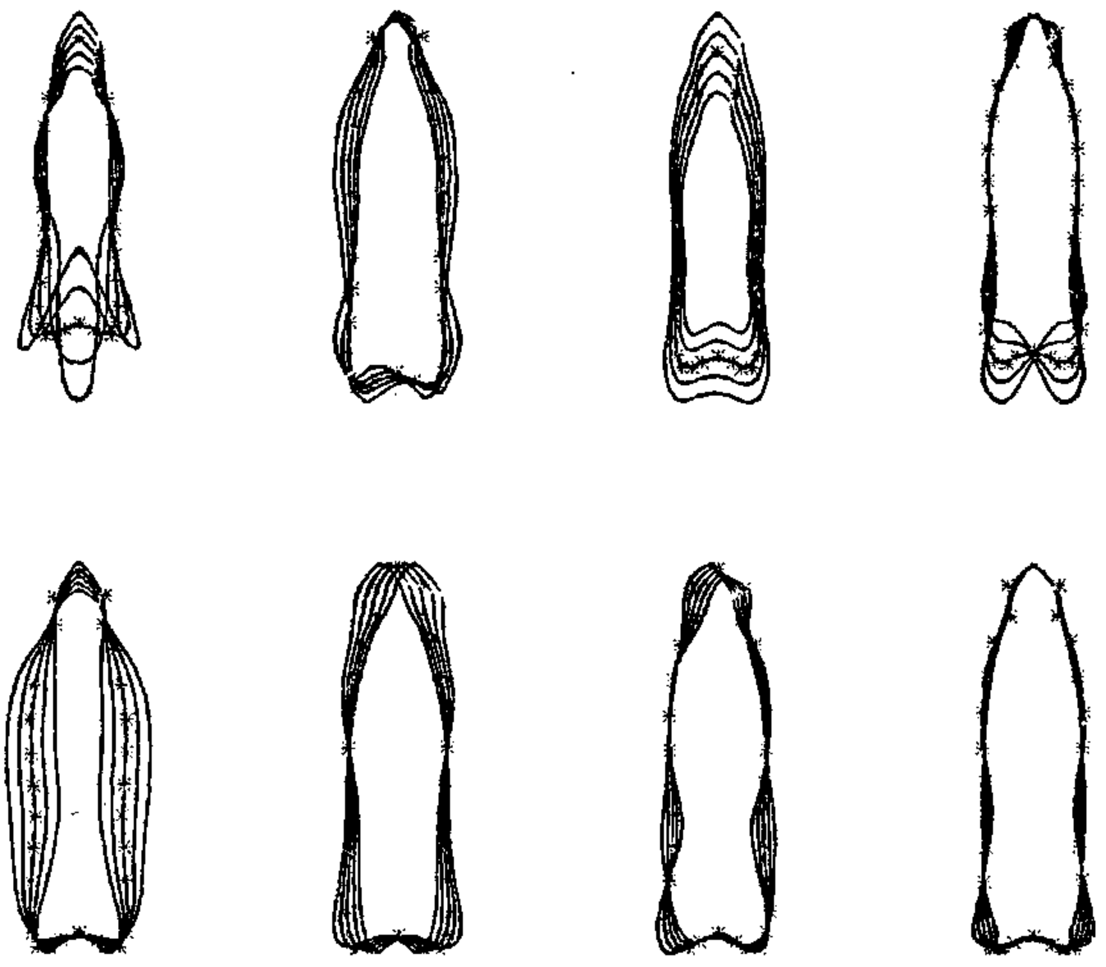


Figure 7.3: The effect of deformable (non rigid) shape variation of a training sequence. The 1st, 2nd, 3rd, 4th, 5th, 6th, 8th and 10th principal component variation of a traning set comprising about 600 different human shapes captured at different poses. The diagrams are plotted according to equation (7.8), with 5 possible variations for each principal model. The mean shape is marked with the stars (*). This illustrates that the first few principal vectors of a traing sequnece account for most non rigid shape varaitions. Note that the figures are not to scale.

7.5.2 Extension of Shape Matrix

If we assume that a template contour $\mathbf{Q}_0$ (or any contour) can be expressed by equation (7.7), then it is reasonable to use the relation,

$$\mathbf{Q}_0 = \overline{\mathbf{Q}} + \mathbf{Pb} \qquad (7.10)$$

Substituting Eq. (7.10) in equation (7.2), we get the following expression:

$$\mathbf{Q} = \mathbf{D}(a_x, a_y)\left[\overline{\mathbf{Q}} + \mathbf{Pb}\right] + \mathbf{U} \qquad (7.11)$$

where $\overline{\mathbf{Q}} = \left(\overline{\mathbf{Q}}_x, \overline{\mathbf{Q}}_y\right)^T$ is the *mean* of the training set.

Equation (7.11) can now be expanded, and by using simple mathematical manipulation it can be shown that it reduces to the following expression.

$$\mathbf{Q} = \mathbf{W}_{AD}\mathbf{T}_{AD} + \overline{\mathbf{Q}} \tag{7.12a}$$

where

$$\mathbf{W}_{AD} = \begin{bmatrix} 1 & 0 & \overline{\mathbf{Q}}_x & 0 & 0 & \overline{\mathbf{Q}}_y & \mathbf{P}_x & -\mathbf{P}_y \\ 0 & 1 & 0 & \overline{\mathbf{Q}}_y & \overline{\mathbf{Q}}_x & 0 & \mathbf{P}_y & \mathbf{P}_x \end{bmatrix}$$

$$\mathbf{T}_{AD} = [u_x, u_y, f_x\cos\theta - 1, f_y\cos\theta - 1, f_x\sin\theta, -f_y\sin\theta, f_x\cos\theta\mathbf{b}, f_y\sin\theta\mathbf{b}]^T$$

AD stands for planar Affine and Deformable shape changes.

Equation (7.12) shows that the m most significant eigenvectors of **SJ** can now be accommodated in the shape matrix to account for m non-rigid deformations. Where $\mathbf{P} = (\mathbf{P}_x, \mathbf{P}_y)^T$ contains the columns of the m most eigenvectors of **SJ**, and $\mathbf{b} = [b_0, \ldots, b_{m-1}]^T$ are the deformable (non-rigid) shape parameters. Any deformable shape can now be reasonably represented by equation (7.12a).

For most tracking applications the scale in the x and y direction are taken to be equal ($f = f_x = f_y$) any shear effects are disregarded as a planar affine change (absorbed by the non-rigid shape changes). In this case, the planar affine transformation (for the rigid part of shape) reduces to Euclidean similarity transformation. In which case Eq. (7.12a) can be expressed as:

$$\mathbf{Q} = \mathbf{W}_{ED}\mathbf{T}_{ED} + \overline{\mathbf{Q}} \tag{7.12b}$$

where

$$\mathbf{W}_{ED} = \begin{bmatrix} 1 & 0 & \overline{\mathbf{Q}}_x & -\overline{\mathbf{Q}}_y & \mathbf{P}_x & -\mathbf{P}_y \\ 0 & 1 & \overline{\mathbf{Q}}_y & \overline{\mathbf{Q}}_x & \mathbf{P}_y & \mathbf{P}_x \end{bmatrix}$$

$$\mathbf{T}_{ED} = [u_x, u_y, f\cos\theta - 1,, f\sin\theta, f\cos\theta\mathbf{b}, f\sin\theta\mathbf{b}]^T$$

ED stands for Euclidean similarity and Deformable shape changes.

One can define a state vector of size $\mathbf{T}_{AD}$ (or $\mathbf{T}_{ED}$) as reported in [25, 24] and track the shape space over time using a second order dynamic system, whose deterministic and stochastic components are learned from test image sequences. The tracker employed for such a contour tracking is either based on a Kalman filter [6] or Condensation [104] filter. A potential problem that appears with such a high dimensional tracking system is that the elements of $\mathbf{T}_{AD}$ cause linear dependence between the columns of $\mathbf{W}_{AD}$, especially for larger inter-frame displacements with highly deformable shape changes [24]. This is because the proportionality of change in translation, scaling, rotation and non-rigid shape parameters can have widely varying magnitudes. Such a scenario presents difficulty for the tracking algorithm to cope with varying degree of changes, and causes numerical instability in the tracking system.

Another problem with a multi-dimensional *learned dynamic system* is that, the second order unconstrained complex stochastic model generated cannot, in general, be decoupled into a set of independent orthogonal modes, nor can it be reduced to lower dimensional sub-spaces easily [10]. Hence the resulting tracking system will be computationally expensive, particularly for complex objects that deform in high dimensional shape spaces (eg: tracking a walking person captured at low frame rates) and be less reliable.

To avoid the problems discussed, and also in the interest of tracker speed, it is desirable to decompose $\mathbf{W}_{AD}$ into suitable components (thus also decomposing $\mathbf{T}_{AD}$) which will eliminate (or reduce) the linear dependencies on the columns of $\mathbf{W}_{AD}$. In the next section we introduce an effective shape space decomposition and tracking technique to overcome the limitations presented by high dimensional shape spaces.

7.6 Decomposition of Shape Matrix and Shape Space

In the interest of speed (to avoid high dimensional computation) and stability, it is best to separate translation, scaling, rotation and non-rigid shape changes into separate components. The result of such a decomposition (separation) procedure is discussed in this section.

Equation (7.12a) can be separated and represented as follows,

$$\begin{aligned}\mathbf{Q}&=\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}u_x\\u_y\end{bmatrix}+\begin{bmatrix}\overline{\mathbf{Q}}_x&0\\0&\overline{\mathbf{Q}}_y\end{bmatrix}\begin{bmatrix}f_x\cos\theta-1\\f_y\cos\theta-1\end{bmatrix}+\begin{bmatrix}0&\overline{\mathbf{Q}}_y\\\overline{\mathbf{Q}}_x&0\end{bmatrix}\begin{bmatrix}f_x\sin\theta\\-f_y\sin\theta\end{bmatrix}+\begin{bmatrix}\mathbf{P}_x&-\mathbf{P}_y\\\mathbf{P}_y&\mathbf{P}_x\end{bmatrix}\begin{bmatrix}(f_x\cos\theta)\mathbf{b}\\(f_y\sin\theta)\mathbf{b}\end{bmatrix}+\begin{bmatrix}\overline{\mathbf{Q}}_x\\\overline{\mathbf{Q}}_y\end{bmatrix}\\&=\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}T_1\\T_2\end{bmatrix}+\begin{bmatrix}\overline{\mathbf{Q}}_x&0\\0&\overline{\mathbf{Q}}_y\end{bmatrix}\begin{bmatrix}T_3\\T_4\end{bmatrix}+\begin{bmatrix}0&\overline{\mathbf{Q}}_y\\\overline{\mathbf{Q}}_x&0\end{bmatrix}\begin{bmatrix}T_5\\T_6\end{bmatrix}+\sum_{i=1}^{m}[\mathbf{E}_i\quad\mathbf{F}_i]\begin{bmatrix}\alpha b_i\\\beta b_i\end{bmatrix}+\begin{bmatrix}\overline{\mathbf{Q}}_x\\\overline{\mathbf{Q}}_y\end{bmatrix}\end{aligned}\qquad(7.13a)$$

where $\mathbf{E}_i=[\mathrm{P}_{x_{1,i}},\ldots,\mathrm{P}_{x_{N,i}},\mathrm{P}_{y_{1,i}},\ldots,\mathrm{P}_{y_{N,i}}]^T$, *and* $\mathbf{F}_i=[-\mathrm{P}_{y_{1,i}},\ldots,-\mathrm{P}_{y_{N,i}},\mathrm{P}_{x_{1,i}},\ldots,\mathrm{P}_{x_{N,i}}]^T$, where $\mathrm{P}_{\cdot,\cdot}$ are components of $\mathbf{P}$, and $\alpha=f_x\cos\theta,\ \beta=f_y\sin\theta$ respectively.

Assuming the contour undergoes Euclidean similarity and deformable shape changes, then Eq. (7.13a) can be given as,

$$\begin{aligned}\mathbf{Q}&=\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}u_x\\u_y\end{bmatrix}+\begin{bmatrix}\overline{\mathbf{Q}}_x\\\overline{\mathbf{Q}}_y\end{bmatrix}[f\cos\theta-1]+\begin{bmatrix}-\overline{\mathbf{Q}}_y\\\overline{\mathbf{Q}}_x\end{bmatrix}[f\sin\theta]+\begin{bmatrix}\mathbf{P}_x&-\mathbf{P}_y\\\mathbf{P}_y&\mathbf{P}_x\end{bmatrix}\begin{bmatrix}(f\cos\theta)\mathbf{b}\\(f\sin\theta)\mathbf{b}\end{bmatrix}+\begin{bmatrix}\overline{\mathbf{Q}}_x\\\overline{\mathbf{Q}}_y\end{bmatrix}\\&=\begin{bmatrix}1\\0\end{bmatrix}[T_1]+\begin{bmatrix}0\\1\end{bmatrix}[T_2]+\begin{bmatrix}\overline{\mathbf{Q}}_x\\\overline{\mathbf{Q}}_y\end{bmatrix}[T_3]+\begin{bmatrix}-\overline{\mathbf{Q}}_y\\\overline{\mathbf{Q}}_x\end{bmatrix}[T_4]+\sum_{i=1}^{m}[\mathbf{E}_i\quad\mathbf{F}_i]\begin{bmatrix}\alpha b_i\\\beta b_i\end{bmatrix}+\begin{bmatrix}\overline{\mathbf{Q}}_x\\\overline{\mathbf{Q}}_y\end{bmatrix}\end{aligned}\qquad(7.13b)$$

If the observation (measurement) process for the tracker can be separated for translation, scaling, rotation and deformable changes (non-rigid shape changes) separately, then the corresponding state vectors can also be separately grouped and defined as shown below.

$$\mathbf{T}_T=\begin{bmatrix}u_x\\u_y\end{bmatrix},\quad \mathbf{T}_S=[f-1],\quad \mathbf{T}_R=[\sin\theta],\quad \mathbf{T}_{SP}=[b_1,\ldots,b_m]^T$$

Note that in the above representation, $\mathbf{T}_S$ (scaling shape space) is defined free of rotational effects (*θ = 0*), $\mathbf{T}_R$ (rotation shape space) is defined disregarding scaling changes (*f = 1*), and $\mathbf{T}_{SP}$ (deformable shape space) is defined disregarding scaling *and* rotational effects (*f = 1* and *θ = 0*). From a theoretical point of view the separation of scaling and rotational changes are *not* possible (because of the non-linearity in Eq. 7.2), but for practical purposes it is reasonable to separate them to some extent as shown in section (7.8.1).

If $\mathbf{T}_T,\mathbf{T}_S,\mathbf{T}_R,\mathbf{T}_{SP}$ can be tracked separately, then at any instant in time step *k*, the contour Q can be recovered using Eq. (7.13b). This procedure is expected to provide better numerical stability and maintains linear independence between the columns of the shape matrix. One of the advantages of shape space separation is that, for the rigid part of shape change, only a 2 element state vector is required, thus limiting the maximum matrix size to be 2x2 in the tracking filter update calculation (assuming that a constant velocity model is employed). In the case of non-rigid (deformable) shape transformation, we assume that each shape parameter vary independently (and the noise process is isotropic) [10]. With this assumption (supported in section 7.8.2), it can be shown that each element of shape parameter vector **b** can also be separated into *m* independent shape parameters (see also [11]). Thus each shape parameter can now be defined as a scalar state parameter, requiring only scalar calculations in the filter update process.

The decomposition method introduced reduces the computational burden of the tracking system (described next) to a great extent, thus making real time tracking applications possible. Another advantage of using separate filters is that each separated filter ($\mathbf{T}_T, \mathbf{T}_S, \mathbf{T}_R, \mathbf{T}_{SP}$) can employ a different motion model(s) in the tracking process. Such a system allows more freedom for each filter and provides better quality results, as shown in section 7.11.

7.7 Dynamic System

A dynamic system is employed which in the general case follows an *M*-th order AR process, where *M* depends on the order of motion model used for the tracker. The dynamic model for the *M*-th order is given by,

$$\mathbf{T}_k = \sum_{j=1}^{M} A_j \mathbf{T}_{k-j} + B_0 \mathbf{w}_k \tag{7.14}$$

where A_j, B_0 are the deterministic and stochastic part of the dynamic system respectively, and **T** corresponds to the separated shape space ($\mathbf{T}_T, \mathbf{T}_S, \mathbf{T}_R, \mathbf{T}_{SP}$).

For an *M*-th order motion model, equation (7.14) can be expressed more compactly by defining a 'state vector'

$\mathbf{X}_k = \left(\mathbf{T}_{k-M}, \mathbf{T}_{k-M+1}, ..., \mathbf{T}_{k-1}, \mathbf{T}_k\right)^T$ which can then be written as,

$$\mathbf{X}_k = A\mathbf{X}_{k-1} + B\mathbf{w}_k \tag{7.15}$$

$$\text{where, } A = \begin{pmatrix} 0 & I & 0 & 0 \\ \vdots & \vdots & I & \vdots \\ 0 & 0 & \vdots & I \\ A_M & A_{M-1} & \cdots & A_1 \end{pmatrix}, \; and \;\; B = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ B_0 \end{pmatrix}$$

Where matrix A defines the deterministic part of the dynamics and B determines the stochastic part of dynamics. For the proposed tracker, the sizes of identity matrix I and 0 are 2x2 for each of the planar affine parameter filters (and a scalar for each of the Euclidean similarity parameter filters), and a scalar for the deformable (non-rigid) shape parameter filter.

Different motion models can be obtained by choosing suitable values for motion parameters $A_M,\ldots,A_1,B_0$. For any practical purposes, up to a motion model order of 3 ($M = 3$) is sufficient for most applications. Three motion models (a constant acceleration, a constant velocity, and a constant position model) were considered for the proposed tracker for applications reported in this chapter (refer to Appendix E for model details).

7.8 Tracking Filter

The entire tracking process is based on a multiple model filtering scheme (similar to what was reported in Chapter 4). The multiple model filtering framework provides the tracker with the ability to switch motion models according to the object's motion. Each filter in the multiple model framework uses a Kalman filter, and each filter is based on a different motion model. The final estimate of the state parameters considered are a weighted combination of the output of each filter. Details of the multiple model scheme employed is discussed in detail in section 7.9. The Kalman filtering process required for each of the individual filters is discussed in section 7.8.

In the interest of speed and stability, translation, scaling, rotation and non-rigid shape effects are filtered separately as discussed before. Because of this separation of transformation parameters (rigid and non-rigid parameters), the observation provided for each filter also need to be separated from the overall observation of contour (this procedure is required for the Kalman filter recursion to function effectively). Taking these factors into consideration, the basis on which the entire tracking system evolves is shown by the following steps at a given time step k (similar to the process in [10, 11]).

1. Assume the non-rigid shape, scaling and rotation parameters are fixed
2. Estimate the translational changes (making use of the object centroid)
3. Remove the effects of the translation from observation
4. Estimate the change in rotation
5. Remove the effects of rotation from observation
6. Estimate the change in scale

7. Remove the effects of scale from observation

8. Update each non-rigid shape parameter independently

9. Once all the transformation changes (rigid and non-rigid) are complete, combine the separated shape space to construct **T** to recover the new updated contour.

Note: Theoretically changes in scaling and rotational effects *cannot* be separated due to non-linearity of equation 7.2, but from a practical point of view they can be separated to some degree (not with high accuracy) for the tracking applications that we are interested in (details given later).

The measurement and prediction process required for each filter $(\mathbf{T}_T, \mathbf{T}_S, \mathbf{T}_R, \mathbf{T}_{SP})$ of the tracking process is discussed in the following sub-sections.

7.8.1 Measurement Model

The effectiveness and reliability of the tracker depends on reasonable measurements (observations) being provided to the tracker. In the following sections we explain the procedure used to obtain contour measurements.

For the application reported in this chapter, we obtain contour measurements by using the image differencing technique as opposed to obtaining them by casting normals to the estimated contour, and then selecting the high contrast points around the contour (as implemented by Bake et al. [24, 25, 60] and Baumberg et. al. [10, 11]).

Since our image sequences are captured by using a static camera (with known background without any moving objects), image differencing technique can be used to separate the moving objects from the background. The differenced image is later blurred using standard Gaussian blur filter, and the resulting blurred differenced image is thresholded to produce black and white images as shown by Fig. 7.4.

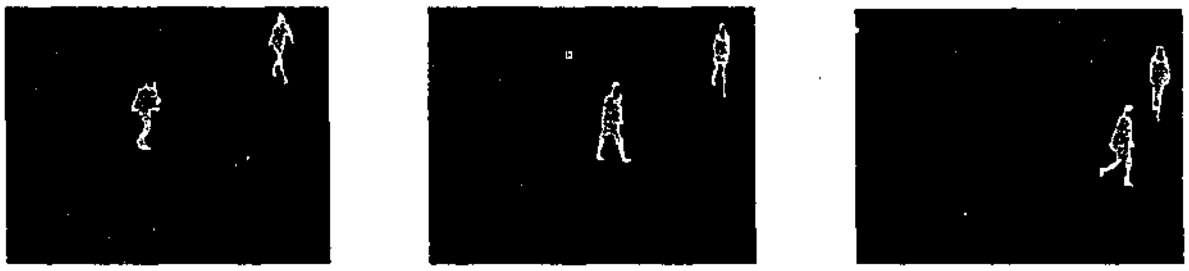


Figure 7.4: Moving objects detected after image differencing and thresholding.

When there is poor contrast between the moving object and the background, fragmentation can occur, resulting in several foreground regions where there should only be one connected region (if only one person is walking). This effect can be reduced by applying morphological filters to fill in the gaps (see [10] for further details). To further reduce and refine the binary image, techniques such as removing pixel clusters of less than a specific quantity can be used. A more advanced model based method like specifying the height to width ratio (for a walking person) or similar shape constraint can also be applied. However, these methods are required only for a noisy set of data.

Each moving object of a reasonable size (eliminating any noise) is now separated and labeled using a search and separate procedure as described in section (7.10). This procedure is applicable if multiple moving objects are detected. The top most point of each moving object's silhouette (for pedestrian tracking example) is assigned a reference point. Starting from the reference point, the N control points of a B-spline are now assigned (equally spaced) along the contour of each of the objects detected in a clock wise manner (see Fig. 7.5, similar to the shape training set process explained in section 7.5.1). This measurement procedure is followed for every frame with the same number of control points for each moving object. For the example of a pedestrian, the top most point in the contour is assumed to be the top of head as shown by Fig. 7.5. This is assuming that pedestrians are always walking in an upright position! For other type of object contours suitable reference point can be assigned (as long as they are uniquely identifiable).

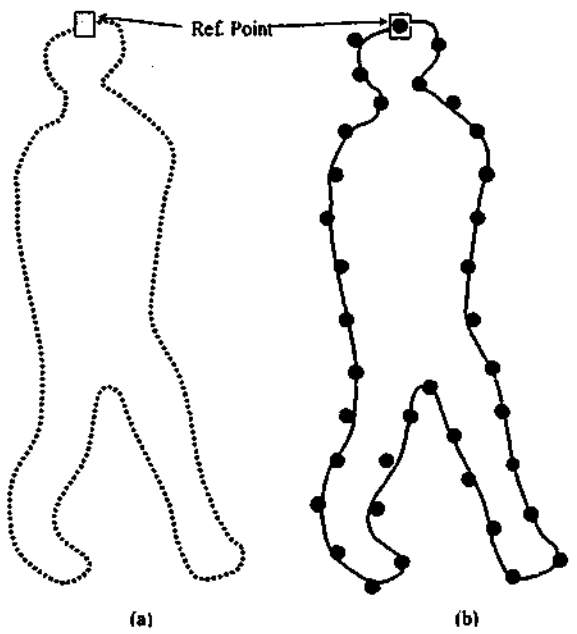


Figure 7.5: Assigning reference point and placing control points along the extracted silhouette.(a) original contour (b) B-spline approximated contour.

If the same number of control points are used to represent object contours in each frame, then it is reasonable to assume that for a given object contour, control point N_i of frame k will nearly correspond to control point N_i of frame ($k+1$). This is illustrated in Fig. 7.6 (not to scale). The control points of frame ($k+1$) will now be used as measurements for estimating the actual contour shape at frame ($k+1$).

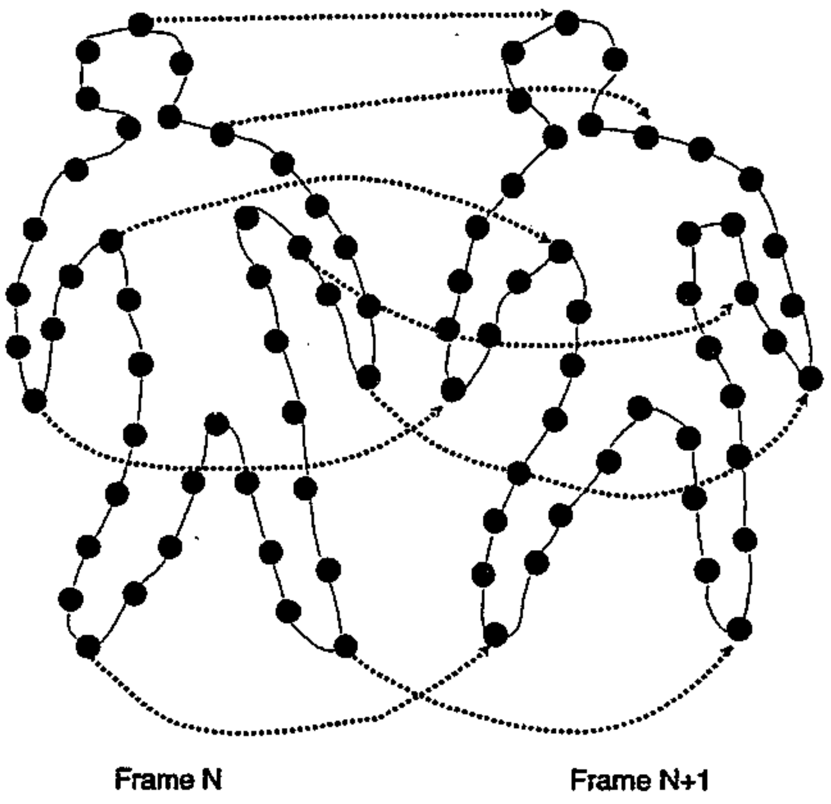


Figure 7.6: Control point correspondence between frames.

7.8.1.1 Separation of Measurements for Each Tracking Filter

Since measurements are made for the entire contour, they need to be reformulated in order to be applied to each of the separated filters. This process ensures that each tracking filter is fed with *only* the measurements that are relevant to them. Ideally the measurements for translation filter should be free of scaling, rotation, and deformable (non-rigid) shape changes. Similarly the measurements for rotation and scaling filters need to be free of translation and deformable shape changes. The deformable shape parameter filter should be free of translation, rotation and scaling changes.

To mathematically illustrate the measurement separation principle, let the overall contour measurements obtained at time step k be Z_k and measurements for translation, scaling, rotation,,and deformable shape parameter filters be denoted by $Z_{T_k}, Z_{S_k}, Z_{R_k}, Z_{SP_k}$ respectively. Each filter measurement procedure indicated can be obtained as described in the following sub-sections (See Fig. 7.7 for an overall illustration).

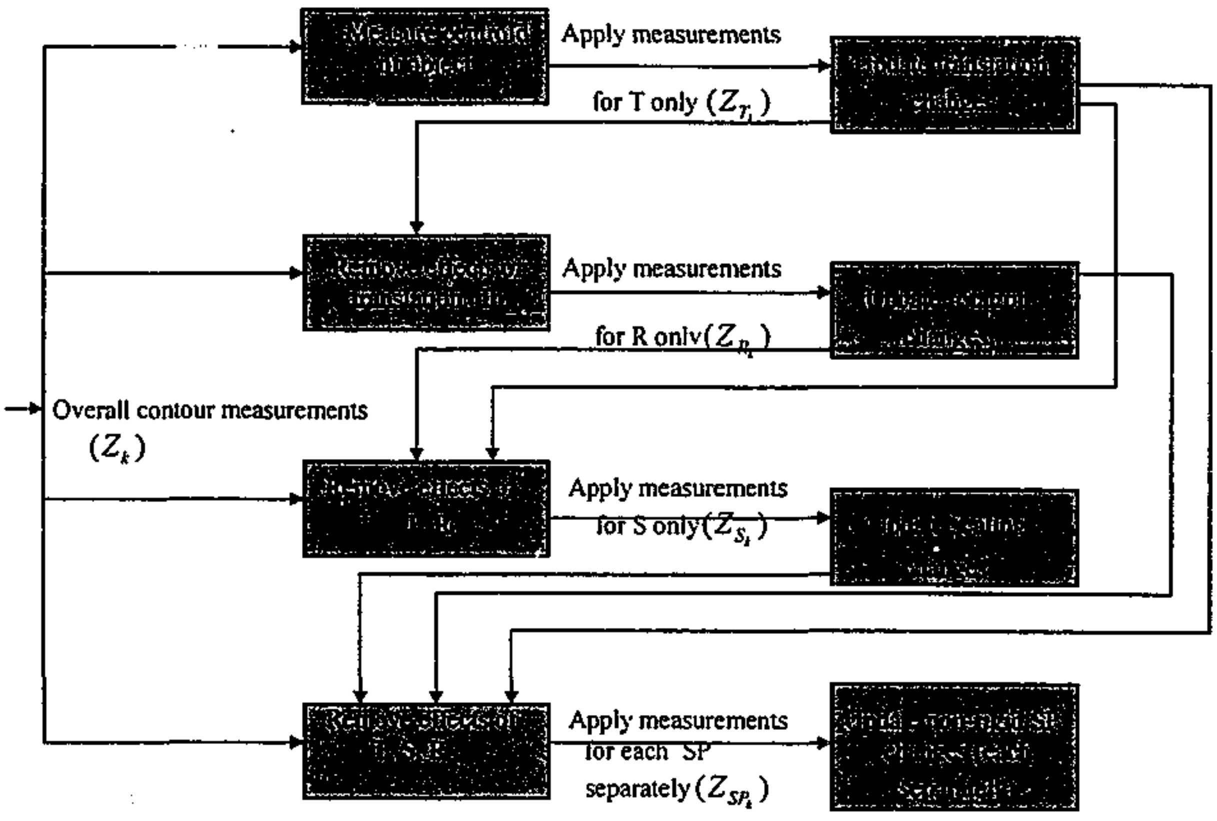


Figure 7.7: One cycle of the measurement application process for each separate filter. The symbols T, S, R, and SP stand for Translation, Scaling, Rotation and non-rigid Shape Parameters respectively.

7.8.1.2 Measurement for Translation

For the translation filter it is assumed that all other transformation parameters are constant while the translation filter is updated (that is, object translation is *only* considered for this filter). Therefore, the effects caused by scaling, rotation and non-rigid shape changes are excluded from the overall measurements. In other words, the translation filter is made to 'see' only the mean shape being translated across the sequence. This process can be achieved by only observing the object centroid (Eq. 7.17). Once the object centroid measurement is known, the translation filter sees the mean shape centered at $cent(Z_k)$. (The object centroid is measured in relation to the mean shape centroid). Mathematically for the *k*-th time step and *i*-th control point of the contour, the translation measurement process is given by,

$$Z_{T_k}^i = cent(Z_k) - cent(\overline{\mathbf{Q}}) \tag{7.16}$$

where *cent*(*) is the centroid point (with *x*, *y* coordinates) of the object contour. It is obtained as follows.

$$cent(Z_k) = \left[\frac{1}{N}\sum_{i=1}^{N} Z^i_{x_k}, \frac{1}{N}\sum_{i=1}^{N} Z^i_{y_k}\right] \tag{7.17}$$

where $Z^i_{x_k}, Z^i_{y_k}$ are the *x* and *y* coordinates of the measured control point *i* at time step *k*, and *N* is the number of control points used to represent the object contour. Normally $cent(\overline{\mathbf{Q}})$ is the origin (center of the mean shape). It should be noted that, as far as the translation filter is concerned each control point of the contour undergoes translation by the same distance as shown by equation (7.16).

The measurement matrix for the *i*-th control point is given by,

$$H^i_T = [\mathbf{H}(s)_i \, \mathbf{W}_T \quad \mathbf{0}] \tag{7.18}$$

where $\mathbf{H}(s)_i$ is the matrix (size *2* x *2N*) contributed by the *i*-th control point defined as in section 7.3. $\mathbf{W}_T$ is the translation part of the shape matrix. For the planar affine case, $\mathbf{W}_T = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$, which is of *size (2N* x *2)*. The **0** in equation (7.18) is a *2* x *2(O – 1)* matrix of zeros, where *O* is the order of the motion model considered. For example, *O=2* for a second order model and *O=3* for a third order model.

7.8.1.3 Measurement for Scaling

By using a similar principle as for the translation filter, the effects of translation, rotation and deformable shape changes are excluded for the scaling filter. This is achieved by placing the mean shape scaled to the same size as the overall measurement Z_k at $cent(\overline{\mathbf{Q}})$ (normally aligned at the origin), so that scaling filter measurements are taken with respect to the mean shape. The only measurement required is the scaling factor, which is given by Eq. (7.20). In effect, the scaling filter 'sees' *only* the scaling changes in relation to the mean shape size. For the *i*-th control point of the contour (at time step *k*), the measurement is given by,

$$
\begin{aligned}
Z^i_{S_k} &= [scale(Z_k)]\overline{\mathbf{Q}}^i - \overline{\mathbf{Q}}^i \\
&= [scale(Z_k) - 1]\begin{bmatrix} \overline{\mathbf{Q}}^i_x \\ \overline{\mathbf{Q}}^i_y \end{bmatrix}
\end{aligned} \tag{7.19}
$$

where $scale(*)$ is the scaling factor obtained as,

$$
scale(Z_k) = \left| \frac{\max(Z_{y_k}) - \min(Z_{y_k})}{\max(\overline{\mathbf{Q}}_y) - \min(\overline{\mathbf{Q}}_y)} \right| \tag{7.20}
$$

Note that only the *height* of the object is taken as the scaling factor for both x and y direction (particularly for human tracking). The same scaling factor is applied to each control point of the contour as shown by Eq. (7.19).

The measurement matrix for the i-th control point is given by,

$$
H^i_S = [\mathbf{H}(s)_i \, \mathbf{W}_S \quad \mathbf{0}] \tag{7.21}
$$

where $\mathbf{H}(s)_i$ is the matrix (size $2 \times 2N$) contributed by the i-th control point defined as in section 7.3. $\mathbf{W}_S$ is the scaling part of the shape matrix. For the planar affine case, $\mathbf{W}_S = \begin{bmatrix} \overline{\mathbf{Q}}_x & \mathbf{0} \\ \mathbf{0} & \overline{\mathbf{Q}}_y \end{bmatrix}$, which is of size ($2N \times 2$). The $\mathbf{0}$ in equation (7.21) is a $2 \times 2(O - 1)$ matrix of zeros, where O is the order of the model considered.

7.8.1.4 Measurement for Rotation

Again by a similar method, for the rotation filter, changes due to translation, scaling and non-rigid shape changes are excluded, thus allowing the filter to '*see*' only the rotational effects. The observation for the rotation filter is obtained by rotating the mean shape by the measured rotation angle θ. The rotation angle is defined as the angle between the vertical line (The mean shape is always vertical) and the line that connect the object centroid and the reference control point of the object contour. The reference point might be the top most or the bottom most control point of the object (the reference point should be identifiable despite rotation, which is object dependant). This method of measuring θ is only applicable (effective) for small angles. Mathematically the measurement process for the i-th control point of the contour is given by,

$$
\begin{aligned}
Z^i_{R_k} &= [rot(Z_k)]\overline{\mathbf{Q}}^i - \overline{\mathbf{Q}}^i \\
&= [rot(Z_k) - I]\begin{bmatrix} \overline{\mathbf{Q}}^i_x \\ \overline{\mathbf{Q}}^i_y \end{bmatrix}
\end{aligned}
\tag{7.22}
$$

where $rot(Z_k) = \begin{bmatrix} \cos\theta_k & -\sin\theta_k \\ \sin\theta_k & \cos\theta_k \end{bmatrix}$ and θ is the rotation angle.

As before, only a single measurement is required (θ), and is applied to each control point of the contour as given by Eq. (7.22). For the walking pedestrian examples reported in this report the objects are assumed to appear vertical and thus rotational effects are ignored.

The measurement matrix for the *i*-th control point of the contour is given by,

$$
H^i_R = [\mathbf{H}(s)_i \mathbf{W}_R \quad \mathbf{0}] \tag{7.23}
$$

where $\mathbf{H}(s)_i$ is the B-spline matrix (size *2* x *2N*) contributed by the *i*-th control point. $\mathbf{W}_R$ is the rotation part of the shape matrix. For the planar affine case, $\mathbf{W}_R = \begin{bmatrix} \mathbf{0} & \overline{\mathbf{Q}}_y \\ \overline{\mathbf{Q}}_x & \mathbf{0} \end{bmatrix}$, which is of size (*2 N x 2*). The size of **0** is as for the scaling filter.

7.8.1.5 Measurement for Deformable (Non-Rigid) Shape Parameters

Once the rigid transformation effects are removed from the overall contour measurements, then the appropriate non rigid shape parameter changes can be measured by using equation (7.24). This process will ensure that only the deformable shape change are considered.

$$
Z^{ij}_{SP_k} = [Z^i_k - cent(Z_k)][scale(Z_k)rot(Z_k)]^{-1} - \overline{\mathbf{Q}}^i \tag{7.24}
$$

where $Z^{ij}_{SP_k}$ is the measurement for the *j*-th deformable shape parameter for the *i*-th control point at time step *k* (or frame *k*). Equation (7.24) is described as the overall measurement (Z_k) of the contour at time *k*, translated, scaled and rotated, so that (Z_{SP_k}) and the mean shape ($\overline{\mathbf{Q}}$) are aligned at the origin. The difference of the 2 shapes will now give the measurements for the non-rigid (deformable) shape variation *only*.

The deformable shape parameter measurement matrix for the j-th non-rigid shape parameter for the i-th control point of the contour is simply,

$$H_{SP}^{ij} = p^{ij} \tag{7.25}$$

where p^{ij} is the element belonging to the j-th column eigenvector of **P** (matrix containing the m principal eigenvectors, see Eq. (7.7)) corresponding to the contribution made by the i-th control point.

7.8.2 Prediction Model

Prediction is applied to each filter once at each time step. The stages involved in the prediction phase are discussed in this section (the algorithm recursion is also displayed in Fig. 7.8).

7.8.2.1 Prediction for Translation, Scaling and Rotation Filters

The state vector for the filter based on an M-th order motion model is expressed as,

$$\mathbf{X}_{FT_k} = \left(\mathbf{T}_{FT_{k-M-1}}, \ldots, \mathbf{T}_{FT_k}\right)^T \qquad FT = T, S, R$$

where FT refers to Filter Type, and T, S, and R stand for Translation, Scaling and Rotation respectively.

For example, the state vectors at time step k for Translation, Scaling and Rotation filters are given by,

$\mathbf{X}_{T_k} = \left(\mathbf{T}_{T_{k-M-1}}, \ldots, \mathbf{T}_{T_k}\right)^T$, $\mathbf{X}_{S_k} = \left(\mathbf{T}_{S_{k-M-1}}, \ldots, \mathbf{T}_{S_k}\right)^T$, and $\mathbf{X}_{R_k} = \left(\mathbf{T}_{R_{k-M-1}}, \ldots, \mathbf{T}_{R_k}\right)^T$ respectively.

Similar notations apply to the state covariance matrix P, state transition matrix A, Process noise matrix G, Measurement noise matrix R, Kalman gain K and the measurement matrix H for each type of filter as described in the following sections.

The prediction process for each type of filters (T, S, and R) for time step k is shown below.

$$\mathbf{X}_{FT_k} = A_{FT}\hat{\mathbf{X}}_{FT_{k-1}} \qquad FT = T, S, R \tag{7.26}$$

and

$$P_{FT_k} = (A_{FT})P_{FT_{k-1}}(A_{FT})^T + G_{FT} \qquad FT = T,S,R \tag{7.27}$$

Where the process noise G_{FT} can be appropriately set (only a 2x2 matrix is required when considering Euclidean similarity case using a constant velocity model), or learned from a training sequence [24, 25], see Appendix E for some details.

Following the prediction step for a given time step, a number of measurements can be made (as explained before). For each measurement, the curve estimate is updated as follows:

$$\hat{\mathbf{X}}_{FT_{k+1}} = \hat{\mathbf{X}}_{FT_k} + K_{FT}\upsilon_{FT} \qquad FT = T,S,R \tag{7.28}$$

where υ_{FT} $(= Z_{FT} - H_{FT}\hat{\mathbf{X}}_{FT_k})$ is the innovation obtained separately for each filter (as applicable to each filter). The Kalman gain for the measurement is given by:

$$K_{FT} = P_{FT_k}(H_{FT})^T \left(H_{FT}P_{FT_k}H_{FT}{}^T + \left(R_{FT}\right)^2\right)^{-1} \qquad FT = T,S,R \tag{7.29}$$

where R_{FT} is the measurement noise matrix, set suitably (only a 2x2 matrix is required) or obtained by using the procedure given in [10, 24, 60]. After the measurement has been applied via the Kalman gain to the estimated state $\hat{\mathbf{X}}_{FT_k}$, its covariance must be updated using:

$$P_{FT_{k+1}} = (I - K_{FT}H_{FT})P_{FT_k} \qquad FT = T,S,R \tag{7.30}$$

where H_{FT} is the measurement matrix as applicable to each filter as given in the previous section, and *I* is the identity matrix whose size will depend on the order of the motion model employed.

The initial values for *P* and *R* can be selected manually for each filter type (for example, *P, R* will only be a 2x2 matrix if a constant velocity model is used) or selected by some other method [10, 60].

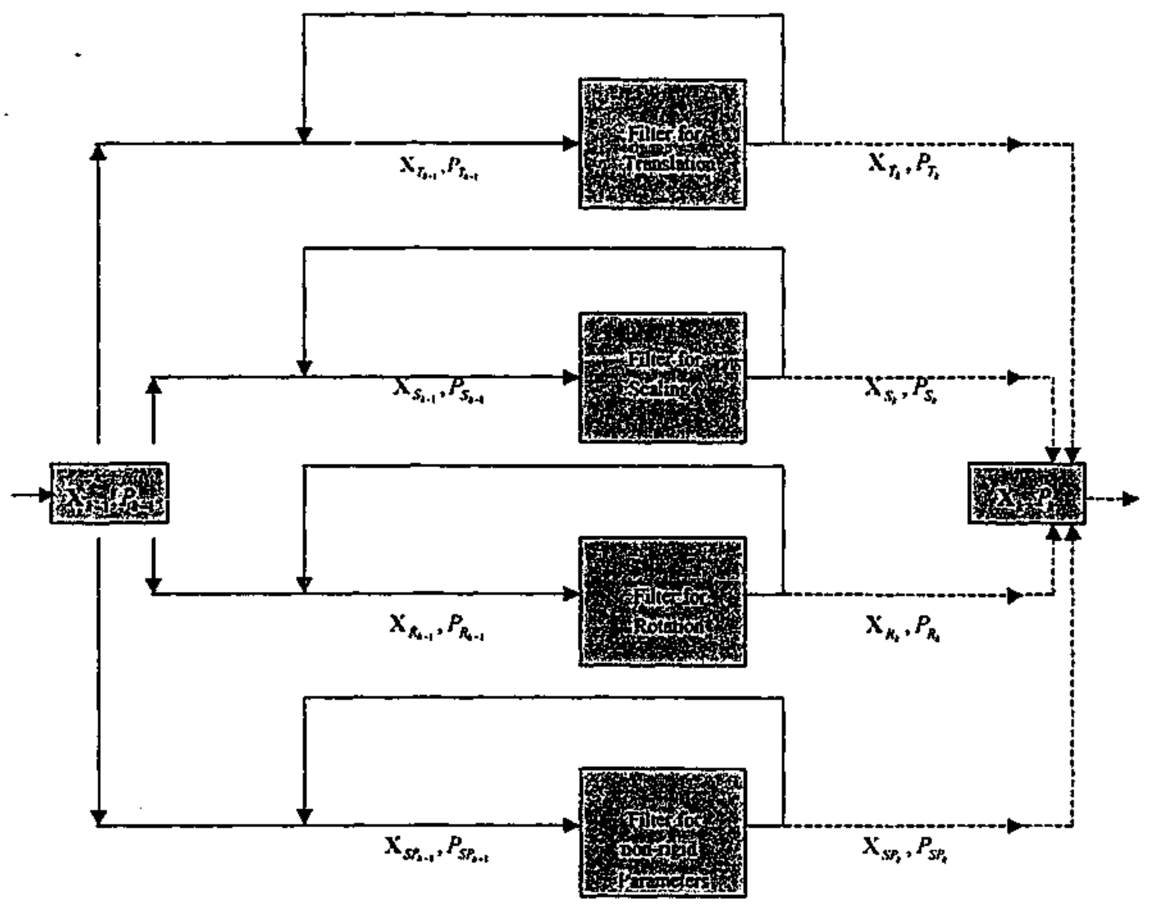


Figure 7.8: One cycle of the prediction process of the tracking system based on a single motion model.

7.8.2.2 Prediction for Non-Rigid (Deformable) Shape Parameter Filter

In this section we provide the state vector representation for the non-rigid shape parameters (deformable parameters) and the filter update process that is required for tracking deformable changes of objects.

7.8.2.2.1 State Vector Representation

The multidimensional (*m*-th order) state vector for non-rigid shape parameters can be decomposed into single state vectors (for each shape parameter).

The multidimensional state vector (assuming an *M*-th order motion model is employed) is given by,

$\mathbf{X}_{SP_k} = \left(\mathbf{T}_{SP_{k-M-1}},....,\mathbf{T}_{SP_k}\right)^T$ where $\mathbf{T}_{SP_k} = \left(T^1_{SP_k},....,T^m_{SP_k}\right)$. If each of the deformable parameter is separated (basis for decoupling is given in the next section), then the reduced order state vector is given by,

$$\mathbf{x}^j_{SP_k} = \left(T^j_{SP_{k-M-1}}, \dots, T^j_{SP_k}\right)^T$$

This is the state representation for the *j*-th non-rigid shape parameter.

Experiments suggest that the non-rigid shape parameters vary randomly between frames. Therefore, the assumption of a first order motion model was adequate for most of our tracking applications (though this is not a requirement). In this case, the state dynamic equation for the *j*-th non-rigid parameter now becomes:

$$x^j_{SP_k} = a x^j_{SP_{k-1}} + w^j_k$$

where the state scalar is now represented by $x^j_{SP_k} = \left(T^j_{SP_k}\right)$, and $a \sim 1$, and w is a Gaussian distributed noise at time step k. A finite element based technique to estimate a with reasonably high accuracy was given in [10], but the method requires additional computational cost for the extra accuracy in prediction.

7.8.2.2.2 State Covariance Update

The theoretical basis for decoupling the deformable shape parameters is briefly given below (see Appendix D, [10] for details).

Assuming that the effects of translation, scaling and rotation are filtered out, the covariance update equation (prior to decoupling each shape parameter) is given as [10]:

$$\begin{aligned} P^{-1}_{SP_k}(+) &= P^{-1}_{SP_k}(-) + [H^T_{SP_k} R^{-1}_{SP_k} H_{SP_k}] = P^{-1}_{SP_k}(-) + (\mathbf{P})^T [r\mathbf{J}^{-1}]^{-1} (\mathbf{P}) \\ &= P^{-1}_{SP_k}(-) + r^{-1} I \end{aligned} \tag{7.31}$$

where **P** is the matrix containing the m principal eigenvectors corresponding to the m most significant eigenvalues. **J** is the metric matrix as discussed in section (7.4), r is a scalar measurement variance constant $(R_{SP_k} = r\mathbf{J}^{-1})$.

Note: The matrix of eigenvectors **P** was derived such that $\mathbf{P}^T \mathbf{J} \mathbf{P} = I$ (see Appendix D).

Assuming $P^{-1}_{SP_k}(-)$ is diagonal, then after applying the measurements for this filter, the updated covariance matrix is still diagonal. Assuming P_0 is diagonal, the covariance matrix is always diagonal. Thus the system can be decoupled into *m* independent 1D Kalman filters. The covariance update equation (with scalar values) for the *j*-th filter ($1 \le j \le m$) now becomes [10]:

$$[\sigma^j(+)]^{-1} = [\sigma^j(-)]^{-1} + r^{-1} \tag{7.32}$$

where $\sigma^j = [P_k]_{j,j}$ is simply the variance of the current estimate for non rigid shape parameter x^j. The scalar r can be manually set. The update process for each non-rigid shape parameter (*j*) now follows a normal Kalman filtering process as given by the following steps.

$$x_k^j = ax_{k-1}^j, \qquad \sigma_k^j = a\sigma_{k-1}^j a + g, \qquad \hat{x}_{k+1}^j = \hat{x}_k^j + \kappa^j \upsilon^j$$
$$where \;\; \kappa^j = \sigma_k^j \rho^j [\rho^j \sigma_k^j \rho^j) + (r^j)^2]^{-1}, \;\; and \;\; \sigma_{k+1}^j = (1 - \kappa^j \rho^j)\sigma_k^j$$

where a is normally set to 1 (or estimated using a learning process such as in [10, 25]), the process noise scalar g is set to an appropriate value (1-20). The state variance σ, the innovation υ, and the Kalman gain κ are all scalars. p^j is the element belonging to the *j*-th column eigenvector of **P** (matrix containing the *m* principal eigenvectors).

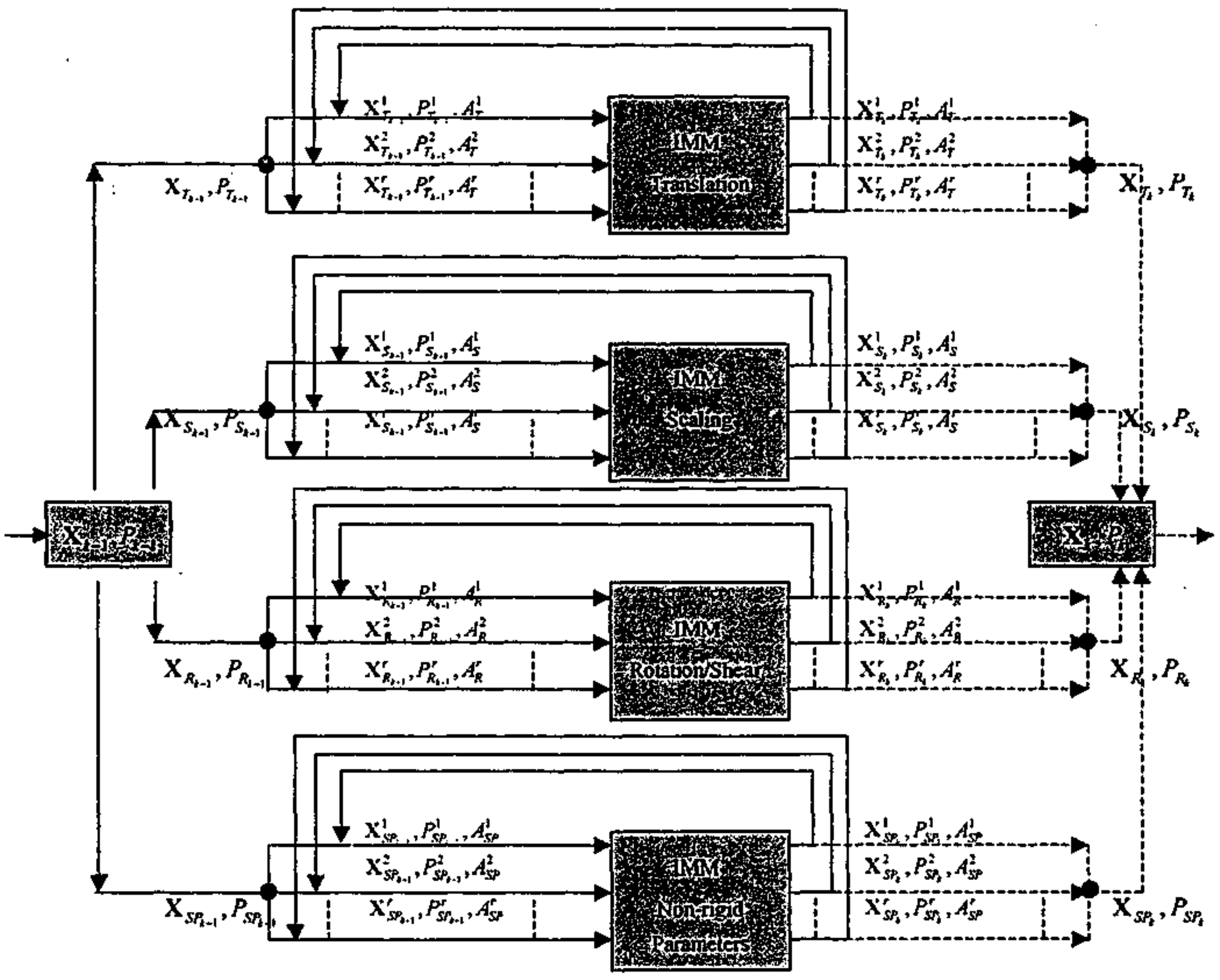
7.9 Automatic Motion Model Switching

Conventionally the tracking process is generally based on a single motion model known a priori. The disadvantage of such a tracking system is that, when the object of interest changes motion, the tracker becomes less reliable and possibly loses track eventually. For example, when tracking a pedestrian, the pedestrian can be walking, running or can suddenly halt. To accommodate all possible motions of the object, the tracker should be able to adapt (switch) between different motion models. In other words an automatic model switching capability should be included in the tracking mechanism to cope with variable motion of objects. Such a model switching process can be achieved by incorporating a multiple model filtering technique (as explained in chapter 4). Amongst the many model switching algorithms available [5], the Interacting Multiple Model (IMM) algorithm has been shown to give good model switching results [184, 185] for visual tracking applications (also demonstrated in chapter 4 for point feature tracking).

With IMM embedded in the contour tracking framework, the tracker is capable of adapting to the closest motion model (or a combination of motion models) available from the bank of models that best describes the object's actual motion. The steps involved in the IMM algorithm are given in detail in Chapter 4.

7.9.1 The application of IMM in the Contour Tracking Framework

The application of IMM for the proposed tracking system is illustrated in Fig. (7.9). Each of the filters follows a Kalman filter recursion as described in previous sections. Note that the IMM is separately applied to the translation, scaling, rotation and deformable shape parameter filters. The advantage is that each of the IMM implemented can employ different motion models. For example, the translation filter can use 3 motion models, while the scaling and rotation filters can use 2 motion models and the deformable shape parameter filter can use only 1 model. Such a freedom provided for the tracking filters give excellent tracking results while providing model switching ability for the tracker. A variety of contour tracking examples are shown in the results section to illustrate the principles discussed.



X_{k-1}, P_{k-1}
$X_{T_{k-1}}, P_{T_{k-1}}$
$X^1_{T_{k-1}}, P^1_{T_{k-1}}, A^1_T$
$X^2_{T_{k-1}}, P^2_{T_{k-1}}, A^2_T$
$X^r_{T_{k-1}}, P^r_{T_{k-1}}, A^r_T$
IMM Translation
$X^1_{T_k}, P^1_{T_k}, A^1_T$
$X^2_{T_k}, P^2_{T_k}, A^2_T$
$X^r_{T_k}, P^r_{T_k}, A^r_T$
X_{T_k}, P_{T_k}
$X_{S_{k-1}}, P_{S_{k-1}}$
$X^1_{S_{k-1}}, P^1_{S_{k-1}}, A^1_S$
$X^2_{S_{k-1}}, P^2_{S_{k-1}}, A^2_S$
$X^r_{S_{k-1}}, P^r_{S_{k-1}}, A^r_S$
IMM Scaling
$X^1_{S_k}, P^1_{S_k}, A^1_S$
$X^2_{S_k}, P^2_{S_k}, A^2_S$
$X^r_{S_k}, P^r_{S_k}, A^r_S$
X_{S_k}, P_{S_k}
X_k, P_k
$X_{R_{k-1}}, P_{R_{k-1}}$
$X^1_{R_{k-1}}, P^1_{R_{k-1}}, A^1_R$
$X^2_{R_{k-1}}, P^2_{R_{k-1}}, A^2_R$
$X^r_{R_{k-1}}, P^r_{R_{k-1}}, A^r_R$
IMM Rotation/Shear
$X^1_{R_k}, P^1_{R_k}, A^1_R$
$X^2_{R_k}, P^2_{R_k}, A^2_R$
$X^r_{R_k}, P^r_{R_k}, A^r_R$
X_{R_k}, P_{R_k}
$X_{SP_{k-1}}, P_{SP_{k-1}}$
$X^1_{SP_{k-1}}, P^1_{SP_{k-1}}, A^1_{SP}$
$X^2_{SP_{k-1}}, P^2_{SP_{k-1}}, A^2_{SP}$
$X^r_{SP_{k-1}}, P^r_{SP_{k-1}}, A^r_{SP}$
IMM Non-rigid Parameters
$X^1_{SP_k}, P^1_{SP_k}, A^1_{SP}$
$X^2_{SP_k}, P^2_{SP_k}, A^2_{SP}$
$X^r_{SP_k}, P^r_{SP_k}, A^r_{SP}$
X_{SP_k}, P_{SP_k}

Figure 7.9: One iteration of the tracking system for updating the state vectors and state covariance matrices based on multiple motion models. The system shows that r motion models are utilized for each IMM algorithm. The tracking system has automatic motion model switching capability. Note that the superscript of variables indicate the model identity of the filter (the identity ranges from 1 - r).

7.10 Object Searching and Separation

Moving objects are identified by background subtraction method. The foreground objects are labeled separately (if more than 1 object is detected). To search for an object in the subsequent frame, the estimated object centroid is used. Once the object centroid is estimated for the following frame, then an enlarged bounding box (say 110%-120% of the current bounding box size) is created centered at the estimated bounding box center (see Fig. 7.10). A search for the moving object is now carried out within this box, by looking for the foreground grouped-pixels (white in our case as opposed to black for the background) of reasonable size (not considering noisy elements). If an object exists, then measurements are obtained for that object contour. This process will not be applicable when there is occlusion. To detect occlusion, one has to impose more stringent model based constraints. Some possible examples of constraints for a walking person sequence will be to introduce a height/width ratio or/(and) imposing a range for possible human height in the image plane (assuming the scene is some known meters from the camera) etc. If an extracted foreground region (representing the moving object) violates such conditions, then one can reasonably conclude that there is occlusion (eg: foreground region is unusually large). Another method is to monitor the object centroid values. If there is an unusual change in object centroid (the difference between the measured object centroid and the estimated object centroid), then a possible occlusion might have occurred. During occlusion measurements are abandoned and the last estimated contour is placed at the estimated object centroid. Such a process identifies the number of objects involved in an occlusion, and keeps track of the objects separately. In situations like this, the contour shapes might not fully reflect the objects of interest. When occlusion disappears, the correct measurements are applied once again and the tracker is able to recover from occlusion.

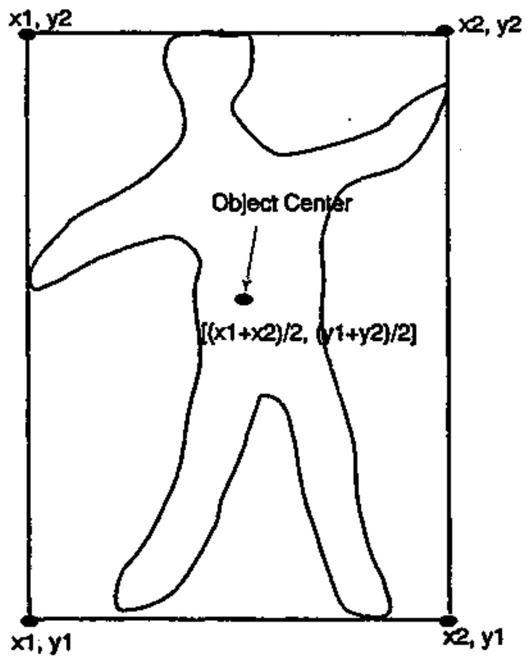


Figure 7.10: A bounding box encompassing the contour.

7.11 Result and Discussion

The tracking algorithm introduced in this chapter (we name it CONT-IMM) has been applied to a variety of image sequences to test the tracker's ability to copy with rigid and non-rigid shape changes. The image sequences were captured at 10-15 frames/sec. mainly to make the inter-frame shape changes of some significance. For each sequence, the number of deformable parameters for non rigid shape changes were selected between 10-12, which account for about 90 % of shape variations of the training sequence considered. For each of the sequence considered, we employed 3 motion models for the rigid part of shape filter (CPM, CVM and CAM) and a single model for the non-rigid shape filter (CPM). The initial mode probability (μ) required by the IMM was set to $1/r$ (r is the number of models in operation) for each filter, thus eliminating any bias between the filters. For a 2 and a 3 model IMM filter, the mode probability and mode transition probability (Eq. 7.33) were set as follows respectively.

$$\mu = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}, \quad p = \begin{bmatrix} 0.95 & 0.05 \\ 0.05 & 0.95 \end{bmatrix}; \qquad \mu = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \quad p = \begin{bmatrix} 0.95 & 0.05 & 0 \\ 0.05 & 0.90 & 0.05 \\ 0 & 0.05 & 0.95 \end{bmatrix}$$

2 Model IMM *3 Model IMM*

As illustrated in the following sections, the tracker has been demonstrated to track well in a variety of conditions.

7.11.1 Sean Sequence

This is an indoor sequence to demonstrate the tracker's ability to track significant scaling changes. As the person approaches the camera, large scaling changes are observed, and as Figs. (7.11) & (7.12) shows the tracker is able to cope with the scaling changes well. Referring to Figs. (7.13 c,d), we see that a CVM is adequate for the first third of the sequence to capture the slowly increasing scale changes, then around half way through the sequence an automatic model switching occurs selecting a CAM to cope with larger scale changes. Such changes cannot be effectively tracked using a single model based tracker [187].

The translation filter tracks the position of the person very well (Fig. 7.13a,b). It should be noted that the translation values plotted in Fig. (7.13a) are the positions of the object centroid. Non-rigid shape changes are also tracked well apart for the last few frames where the legs of the person are not fully recovered (Fig. 7.11). This is partly because of inadequate number of control points selected to represent the object. 32 points were selected for this sequence, but a larger number would have given better shape tracking results at the expense of computational cost. Fig. (7.13e) shows the first 10 tracked non-rigid (deformable) shape parameters, while Fig. (7.13f) shows the absolute error between the actual and tracked (estimated) non-rigid shape parameters.

7.11.2 Outdoor pedestrian sequence

This is a sequence captured by a static camera (15 frames/sec.) at the entrance to a building. We represented the objects using 32 control points (seemed adequate for this application). As can be seen from Figs. (7.14 & 7.15), the tracker tracks both pedestrians with good tracking results (see Figs. 7.16a,b). In some frames the head of the pedestrian is missing, this is because the threshold (fixed for these experiments) at the image differencing and thresholding stage was set too high, hence part of the head disappeared. Fig. (7.16c) shows that the pedestrian on the left is moving fast, thus a CAM is chosen as the most appropriate motion model for the translation filter. Since there is not much scaling of the object (very little change) a CVM is chosen as the best model for the scaling filter (Fig. 7.16d). Rotational effects were not taken into account for the human tracking experiments. Any rotational changes were included in the deformable shape tracking filter. For the human tracking experiments the best 10 principal components (PC) were selected to represent the variation in non-rigid shapes (the best 10 PCs accounted for about 93% of shape variations of the training sequence). As can be seen from Fig. (7.14), most deformable shape variations have been tracked well. Figure (7.16c) shows the non-rigid shape parameters ($\mathbf{b}_i$) varying with time. The first few (particularly the first) parameter variations are more significant than the less significant ones. Fig. (7.16d) shows the absolute error between the actual and the tracked non-rigid shape parameters. The error observed is less than 5% of the actual magnitude of the non-rigid (deformable) shape parameters, which indicates good tracking performance.

7.11.3 Outdoor pedestrians with occlusion

This pedestrian sequence is considered to illustrate tracking under occlusion (see Fig. 7.17). The 2 pedestrians occlude each other towards the end of the sequence. The occlusion is modeled as explained in section (7.10). During the occlusion period the last object shape tracked (when there's no occlusion) is placed at the estimated centroid in the following frame. Doing so is computationally attractive but does not recover the actual shapes well. However, the tracker identifies both objects separately and recovers well from occlusion. To recover the shapes better during occlusion, a method can be followed as explained in [118], [161]. It is also reasonable to consider the fact that the object that appears at the bottom of the frame occludes the ones above them, this was assumed reasonable in the research work reported in [118]. We have adapted this principle in our tracking algorithm.

7.11.4 Waving hand sequence

This is a sequence taken at 10 frames/sec. to account for larger shape variations between frames. The shape of the hand was represented using 50 control points. The hand was moving with variable motion, pausing for a few seconds at the change of direction (at each end). The waving hand was specifically maneuvered in order to test the tracker's ability to cope with large transformational changes, and to test the tracker's automatic model switching capability. Translation, scaling, rotation and non-rigid shape changes were all taken into account for this experiment.

A three-model IMM filter was employed to account for constant acceleration, constant velocity, and constant position changes. All three motion-models were in operation at some part of the duration of the sequence (especially for the translation filter). Figures (7.18-7.19) shows the qualitative tracking results for the hand sequence. As Fig. (7.21a) shows, the translation filter correctly model switches as expected, switching to the most appropriate motion model (confirmed by manual inspection). Fig. (7.20a,b) shows the accuracy of the translation filter. The tracker accuracy in scaling and rotation are displayed in Figs. (7.20c,d). As can be seen from Figs. 7.21(b, c) a combination of motion models are in operation for the scaling filter while a CAM is mostly in operation for the rotation filter (to cope with erratic rotational changes).

The best 12 principal components were selected to account for the non-rigid shape variations. This seemed adequate to capture most of the deformable shape variations. Fig. (7.20e) shows the first 12 tracked non-rigid shape parameters, while Fig. (7.20f) shows the absolute error between the actual and tracked (estimated) non-rigid shape parameters. The accuracy observed was as good as the pedestrian sequence results (~ 1% error). It is also worth comparing the waving hand contour tracking result with that obtained in Chapter 4 for point feature tracking (see Section 4.9.2).

7.11.5 Discussion

The results (quantitative and qualitative) show that the performance of the tracking system is affected by the system parameters and more importantly by the suitability of the linear shape model used. The errors observed (the differences between the true and the tracked contour) could have come from several possible sources.

- Smoothing error – due to the smoothing of the spline representation.
- Truncation error – cased by ignoring the less significant deformable shape modes.
- Modeling error – due to an inaccurate a priori probability distribution (due to segmentation errors and spline approximation errors in the training shapes considered). Also due to inaccurate a priori assumptions in the stochastic model (eg: unexpected large shape changes).
- Filtering error – due to ignoring the off diagonal elements of the deformable shape parameter covariance matrix. Also prior assumptions in fixing the initial state covariance and measurement noise matrices for the rigid transformation filters.
- Poor correspondence – even in the absence of image noise the measurement process is prone to errors as the contour can lock onto the wrong part of the image feature (due to image plane noise or occlusion). This can also be due to the pre-selected value of threshold used at the image differencing and thresholding stage. A too high or a too low threshold value can result in poor quality image measurements, which can in turn lead to poor tracking results. This is a limitation of the tracker presented.
- Numerical error – numerical approximations and compromises made in tracker implementation procedure.
- Contour quality - limitation of the number of control points used in representing the object contour. The same number of control points are used to represent all the moving objects, which limits the quality of track result (particularly when the number of control points are inadequate).

Despite all the reasons mentioned above the tracker proposed gave good contour tracking results. Further performance analysis of the tracker (such as tracking under noise etc.) is discussed in the next chapter.

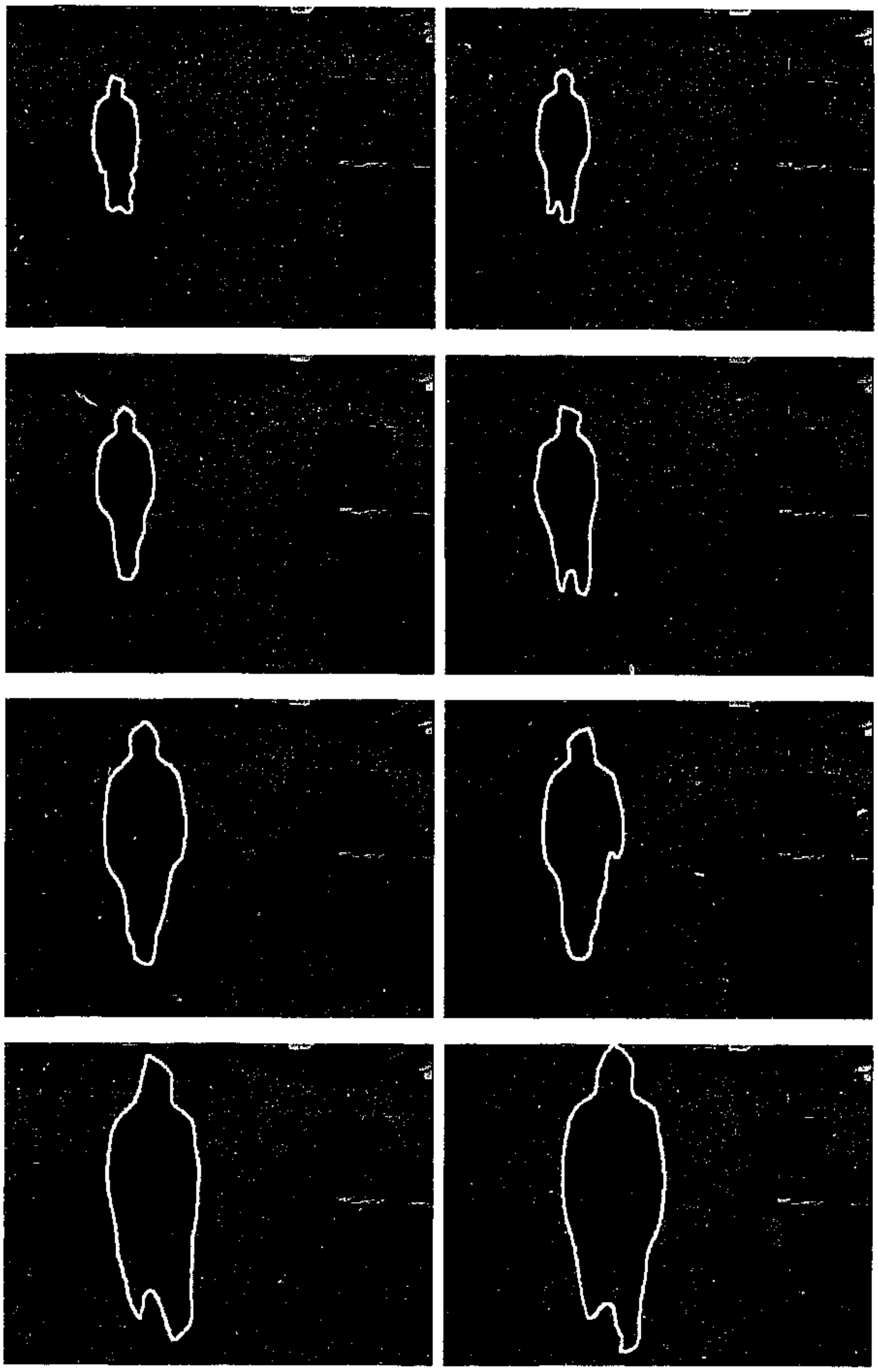


Figure 7.11: Some frames of the indoor walking man sequence (Sean sequence captured at 10frames/sec.) with the tracked contour (white) superimposed on top of the walking person. The tracker has successfully tracked the scaling changes of the person.

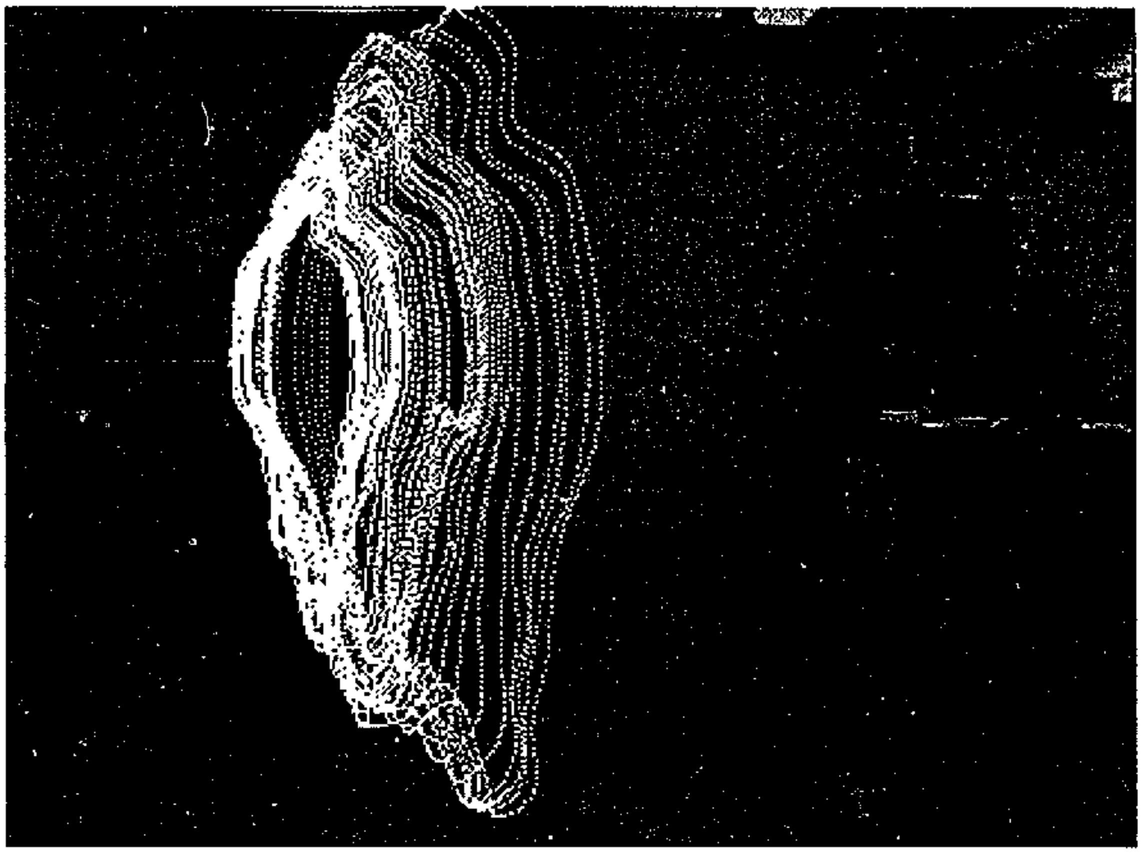


Figure 7.12: Motion of the indoor walking Man (Sean). The tracked silhouettes (white outline) are superimposed onto the first frame of the sequence.

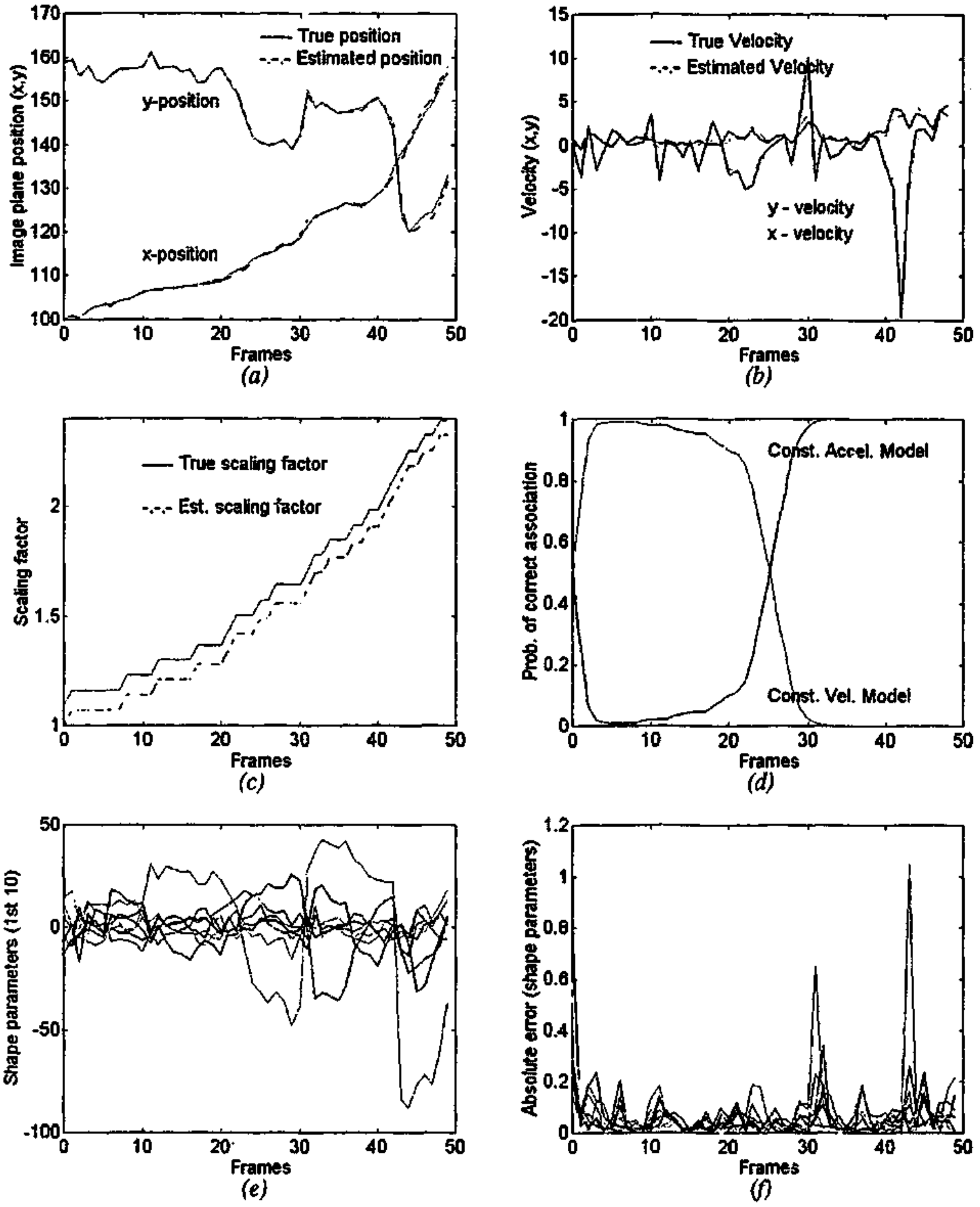


Figure 7.13: Sean sequence tracking results (captured at 10 frames/sec.). (a) Estimated and true position plotted against frame number. The position value is the position of the object centroid. (b) Corresponding estimated and true velocities plotted against frame numbers. (c) Estimated and true scaling changes of the object (in relation to the mean shape). (d) Motion model selection for the scaling filter, the scaling changes in the first third of the sequence are small, therefore a CVM is adequate for correct tracking. The scale changes are significant towards the latter part of the sequence and rightly a model switching takes place (around the 25th frame) to select a CAM for the rest of the sequence. (e) Non-rigid (deformable) shape parameters tracked for Sean sequence: the first 10 tracked non-rigid (deformable) shape parameters corresponding to the largest 10 eigenvalues,(see text for details) plotted against frame number. The parameter with the highest deviation from 0 relates to the first shape parameter (corresponding to the largest eigenvalue), the lowest deviation shape parameter corresponds to the 10th largest eigenvalue. (f) Absolute error between the tracked and the true non-rigid shape parameter values. The larger error values correspond to the first few deformable shape parameters.

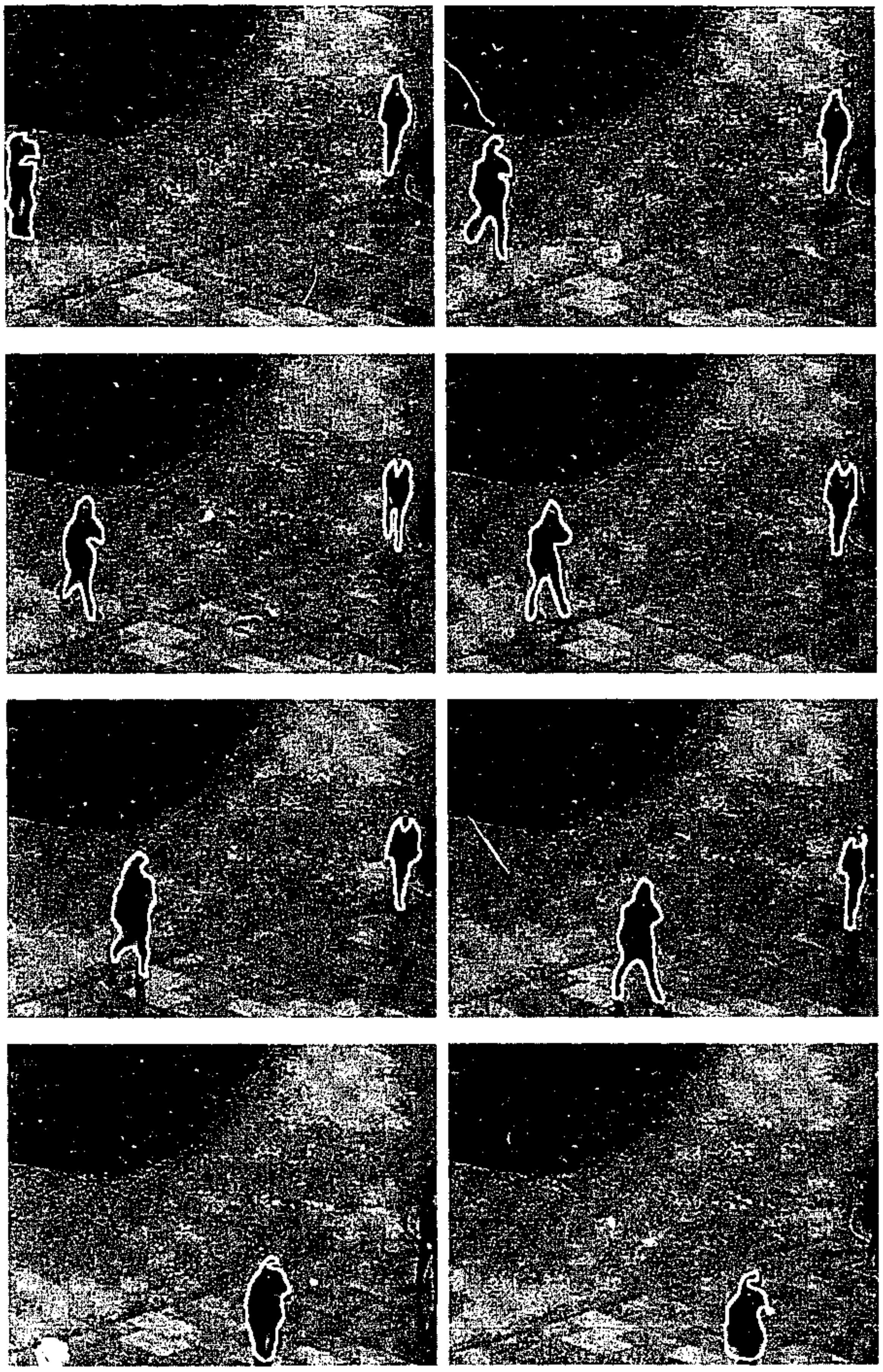


Figure 7. .: Some frames of the outdoor pedestrian sequence (captured at 15 frames/sec): Multiple objects tracked with partial occlusion. The partial occlusion is modelled by introducing a height to width ratio constraint. If this value is violated the tracker gives up on the object. Note that the tracker has abandoned tracking the pedestrian on the right while the person on the left is tracked, despite part of his legs disappearing.

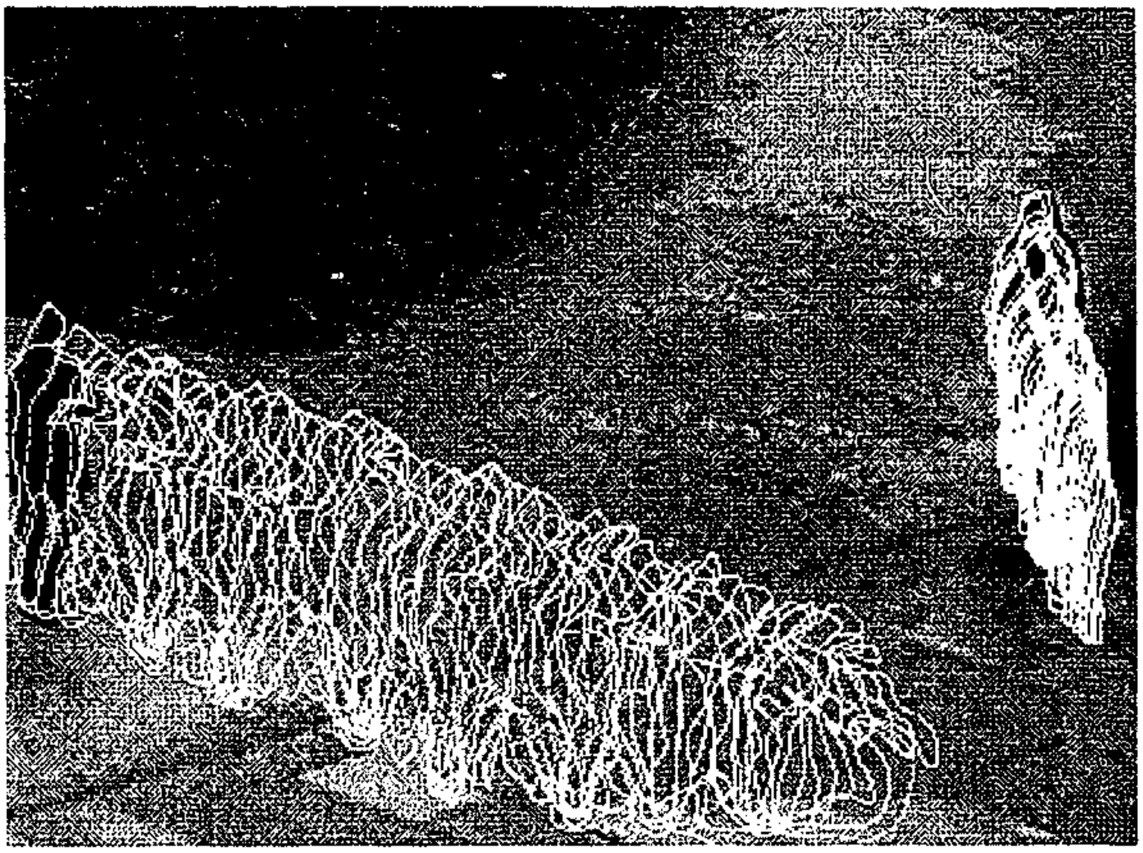


Figure 7.15: Motion of outdoor pedestrians. The tracked silhouettes (white outline) are superimposed on the initial frame.

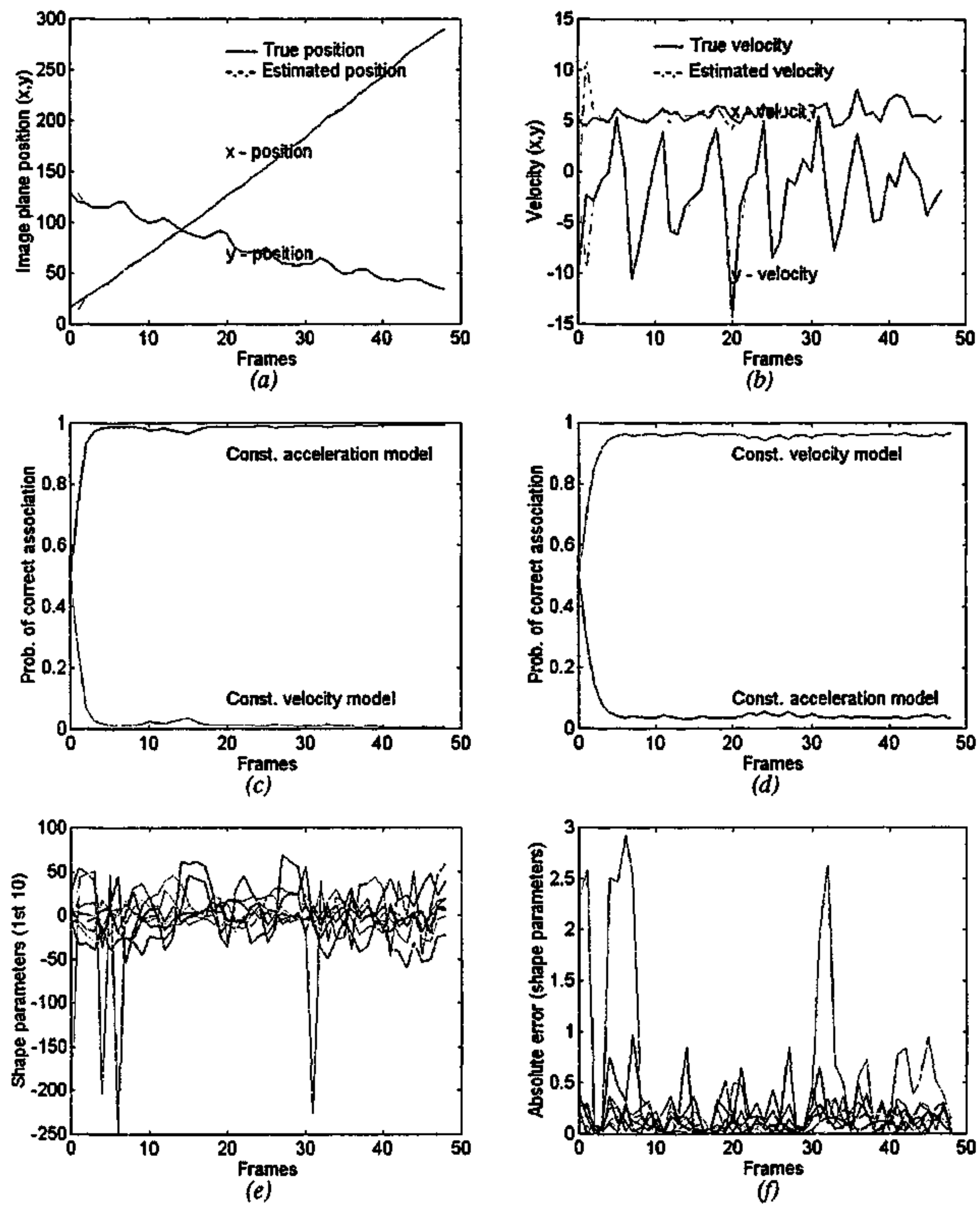


Figure 7.16: Pedestrian sequence 1 tracking results (for pedestrian on the left) captured at 15 frames/sec. (a) Estimated and true position plotted against frame number. The position value is the position of the object centroid. (b) Corresponding estimated and true velocities plotted against frame numbers. (c) Motion model selection for the translation filter: a Constant Acceleration Model (CAM) is selected as the most appropriate motion model, to cope with larger displacements. (d) For the scaling filter, a velocity motion model seems to be adequate to capture small scaling changes. (e) The best 10 tracked non-rigid shape parameters plotted against frame number. (f) Absolute error between the tracked and the true deformable (non-rigid) shape parameter values.

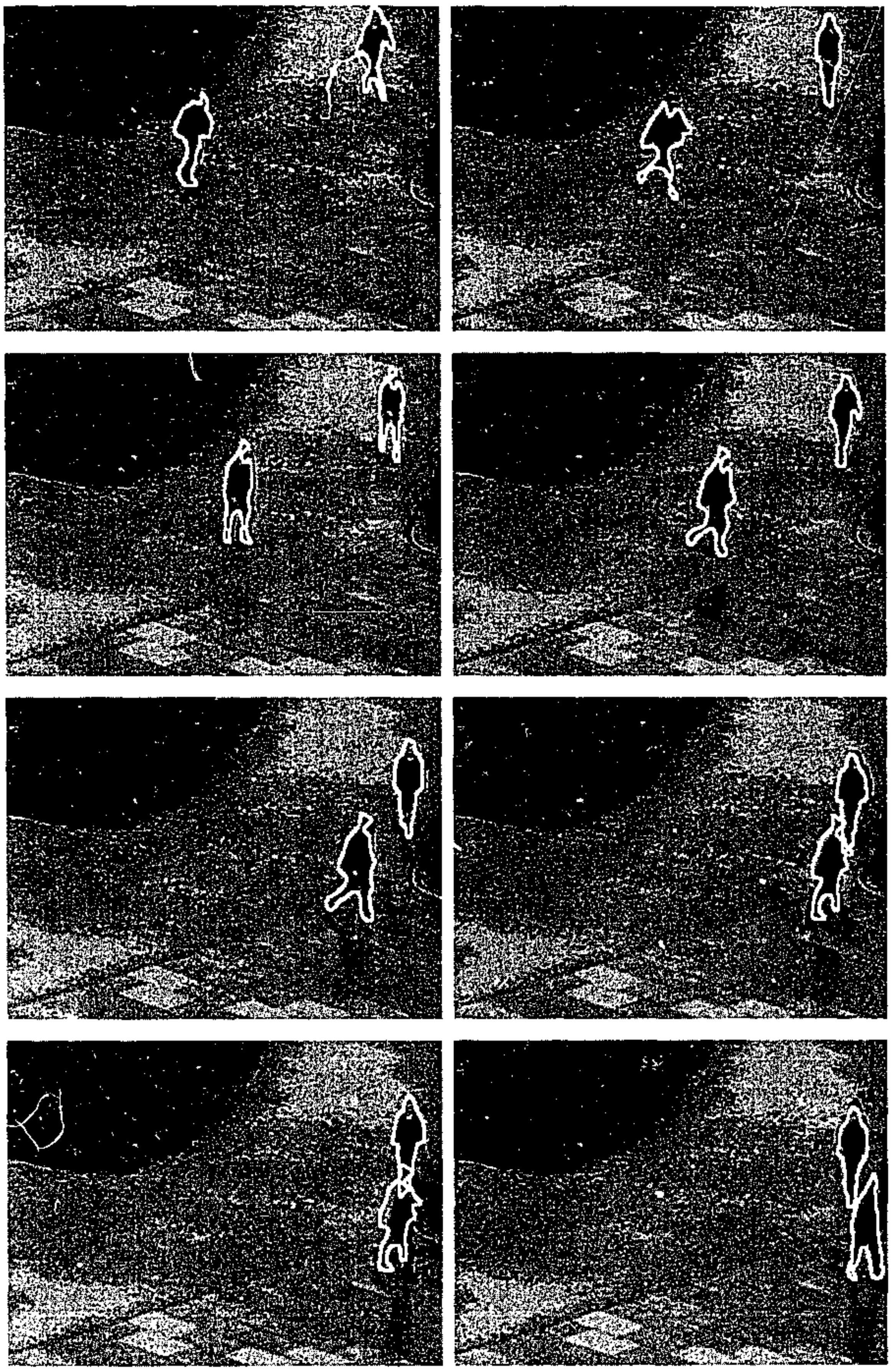


Figure 7.17: Modelling occlusion (captured at 15 frames/sec). When occlusion is detected (either by a sudden change in the object centroid or by verifying the height to width ratio of the object), the measurements are ignored and the current object shape is retained and placed on the estimated object centroid in the following frame. By doing so, the total shape of object cannot be fully recovered (image 7), but the 2 pedestrians are identified separately. As can be observed from the last frame, the tracker recovers from the occlusion well.

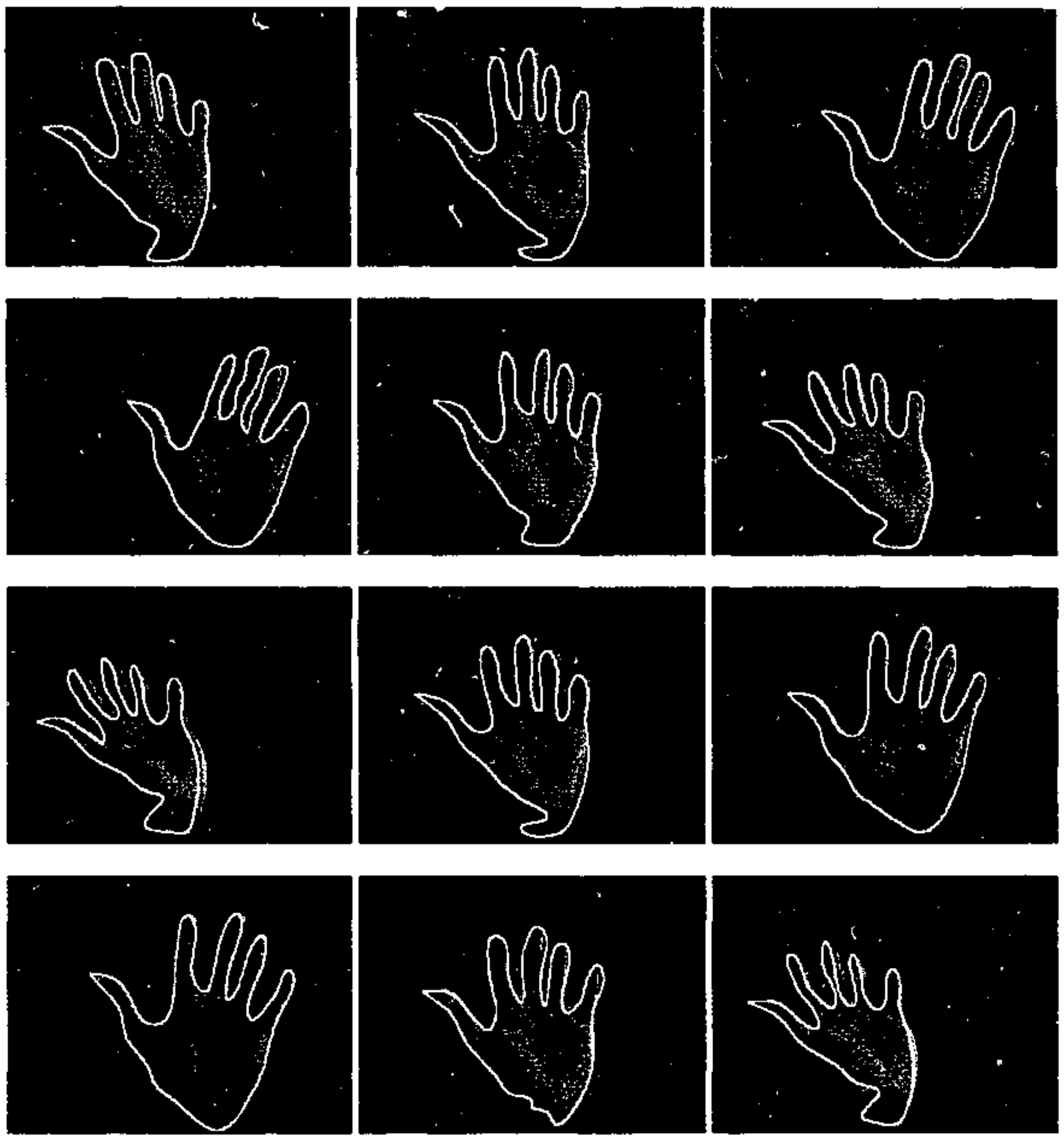


Figure 7.18: Some frames of the waving hand sequence (captured at 10 frames/sec.) showing the tracked silhouette (white outline) superimposed on the hand. The tracker automatically model switches to the most appropriate motion model (chosen from a bank of models) that best describe the waving hand's motion.

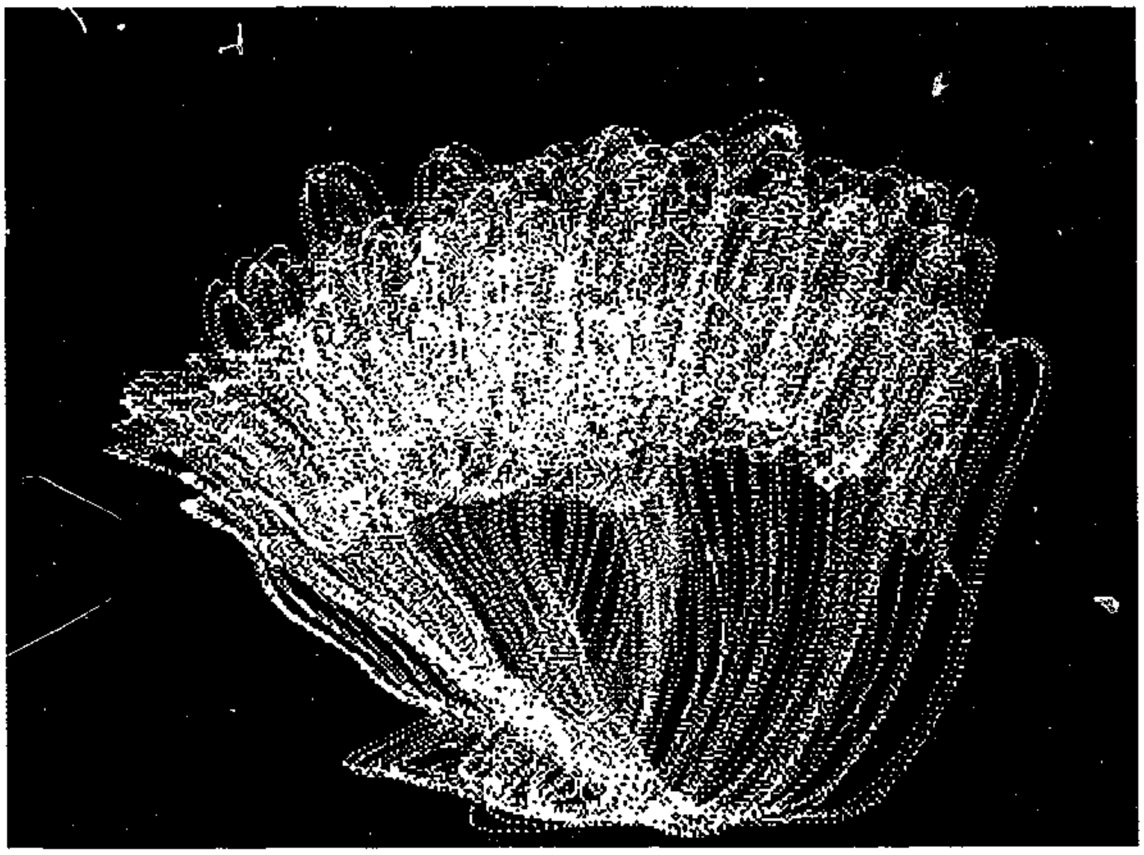


Figure 7.19: The waving hand in motion: The tracked silhouettes are superimposed on the initial frame. The hand initially moves towards the right then to the left and so forth.

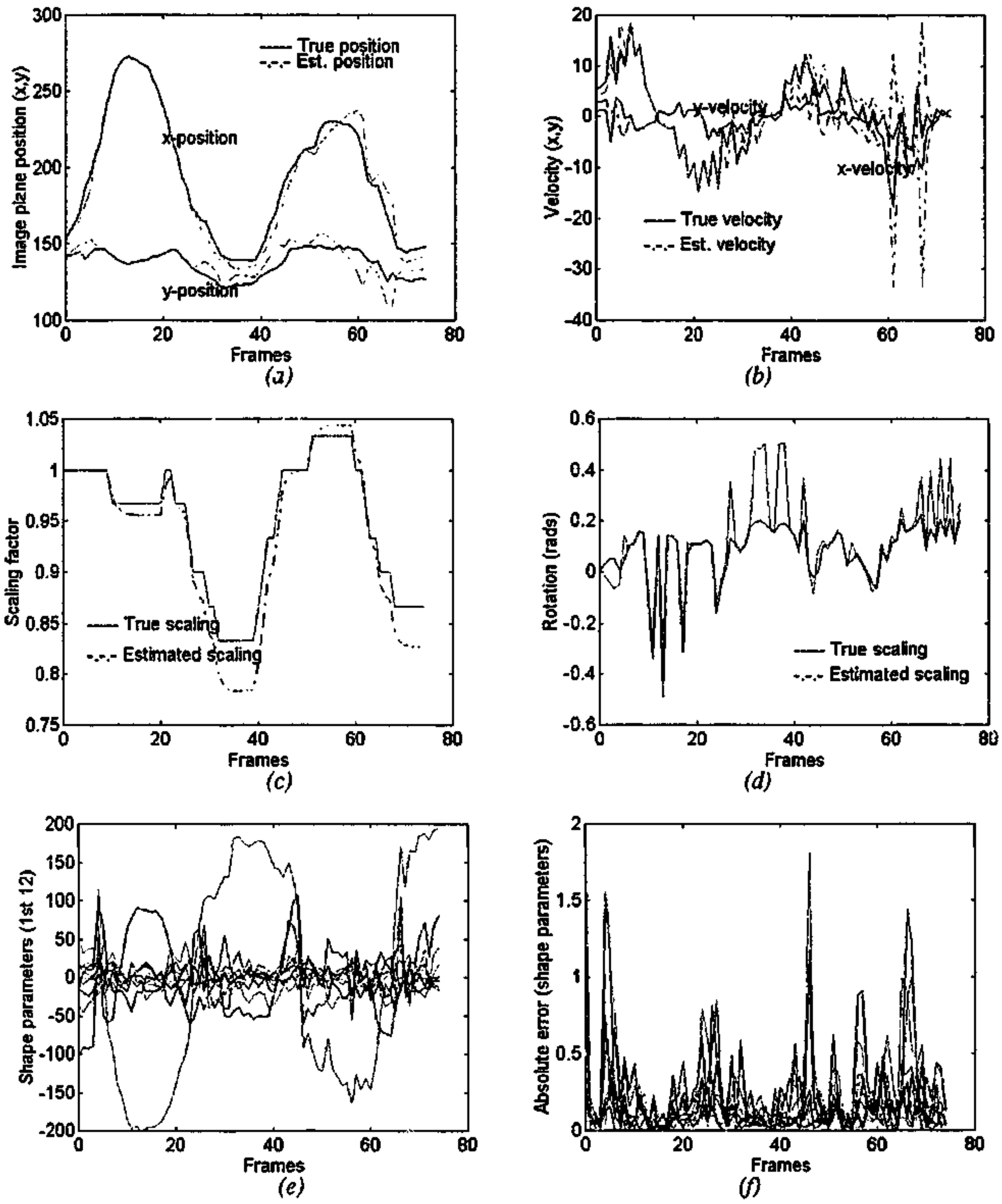
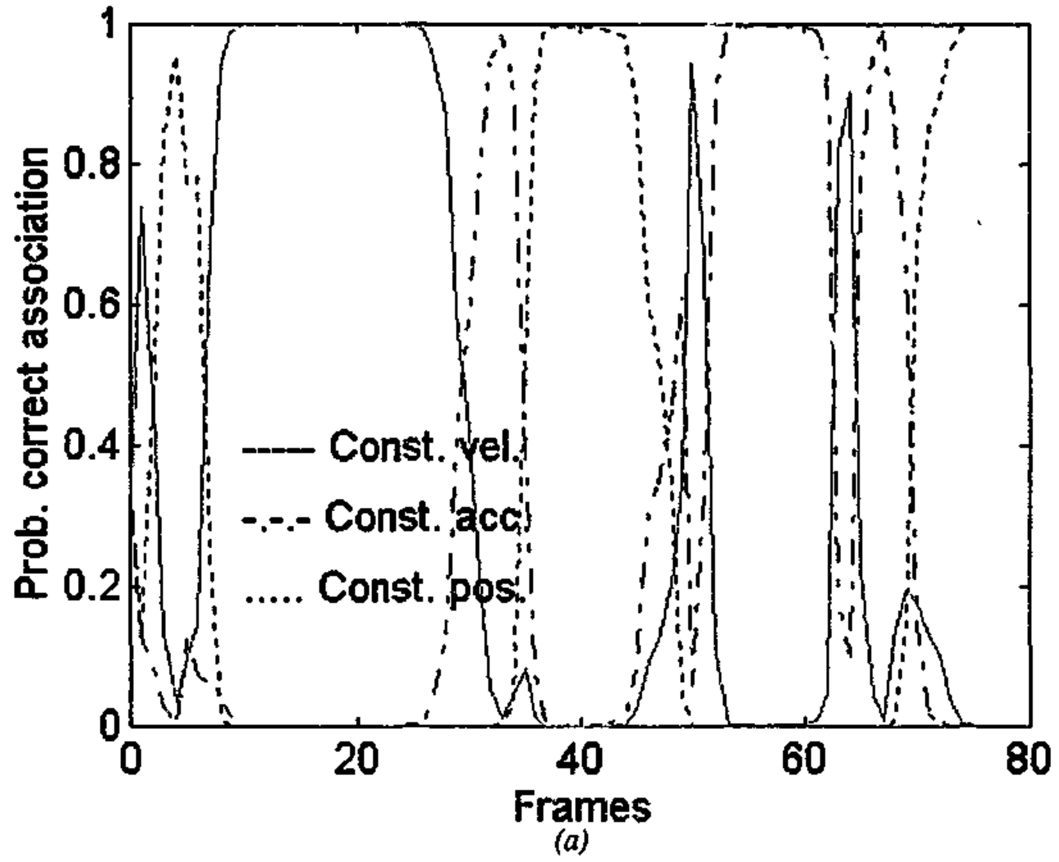
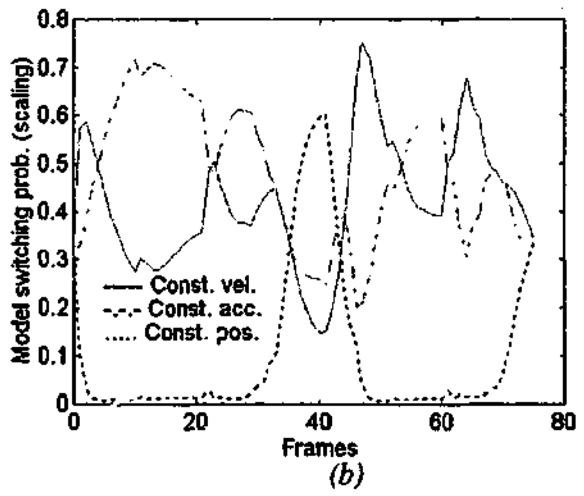


Figure 7.20: Waving hand sequence tracking results (captured at 10 frames/sec.). (a) Estimated and true position plotted against frame number. The position values shown are the position of the object centroid. (b) Corresponding estimated and true velocities plotted against frame numbers. (c) Estimated and true scaling (measured in relation to the mean shape size) changes over time. (d) Estimated and true rotational (measured in relation to the mean shape orientation) changes over time. (e) The best 12 non-rigid shape parameters tracked. (f) Absolute error between the tracked and the true non-rigid shape parameter values.





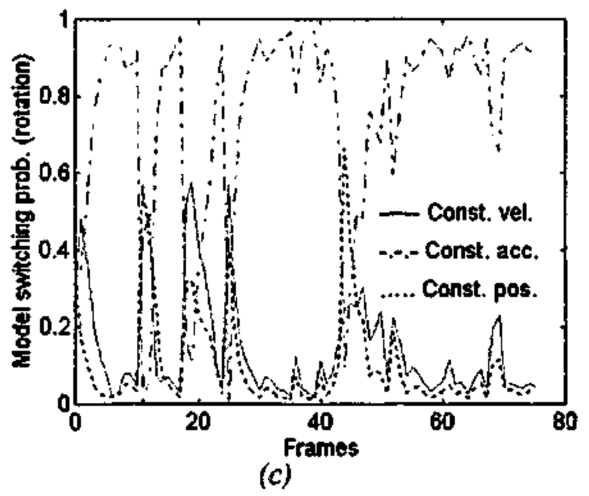


Figure 7.21: Motion model switching probabilities for waving hand sequence. (a) Model switching for translation filter: The translation filter automatically switches motion model according to the hand's motion. The hand starts from a nearly stationary position, then moves with a constant velocity (towards the right) then slightly accelerates and comes to a stationary position. The process is repeated towards the left side, and finally comes to a stationary position shortly after moving towards the right. As expected the translation filter model switches automatically at appropriate times. (b) Model switching for scaling filter: According to the changes in scaling, a combination of CAM and CVM are in operation for most of the duration, except when the hand reaches a stationary position. (c) Model switching for rotational filter: CAM is preferred over other models to cope with small but erratic rotational changes.

7.12 Conclusion

We have provided a contour tracking method (named CONT-IMM) for applications where the inter-frame displacements can no longer be considered very small, and for objects that move with variable motion. Tracking of a walking person is a classic example of such a case, and we have demonstrated the ability of the tracker to cope with rigid and non-rigid shape changes using a variety of applications. The tracking algorithm in general can be applied to any deformable object in motion, providing attractive computational advantages.

In the next chapter we give extended performance analysis of the CONT-IMM tracker comparing its performance (particularly in terms of quality) with the well-known CONDESATION algorithm of Isard and Blake [104], and the pedestrian tracker of Baumberg and Hogg [11].

Chapter 8

Performance Measures for Assessing Contour Trackers

Abstract

In this chapter we present techniques to compare the quality of tracking performances of contour trackers. Three trackers reported to give good tracking performance have been considered for our empirical evaluation. They are the CONT-IMM tracker (chapter 7, [184]), the CONDESATION tracker [24, 104] and the Baumberg tracker [11, 10]. Four different test conditions were set and for each test, the tracking performance of each tracker was measured against four performance measures. The results presented have revealed some interesting findings about the performance of the trackers.

8.1 Introduction

This chapter primarily focuses in comparing the performance of the CONT-IMM tracker (reported in chapter 7) with Baumberg and Hogg's Leeds tracking algorithm [11] and Isard and Blake's Condensation algorithm [104]. The empirical performance measures provided in this chapter is used to assess the quality of the contours tracked by the 3 tracking algorithms.

The literature survey carried out in the area of performance of contour tracking (some aspects were discussed in chapter 2) reveals that very little work has been published to compare the quality of tracker output. Most performance comparison methods presented are specific for the tracker considered, thus cannot be easily employed to compare the performances of other trackers. Examples of such methods can be found in [3, 63, 77, 108, 117]. Formulating closed form performance measures for tracking is very difficult given the complexity involved, and can give inaccurate results under varied tracking environments. Therefore, in this chapter, empirical evaluation methods have been described. The methods cater for a variety of applications and conditions under which the tracker performance can be analyzed. The results presented reveal some interesting facts about the trackers.

We use a test image sequence of a walking person to carry out various tracking performance tests. The test image sequence considered was relatively free of clutter and occlusion, so that the focus of the experiments designed was to purely assess the quality of the contour tracked. For the experiments reported in this chapter, the internal parameters of each tracker was tuned to give the best possible result, so that the observations obtained are a fair representation of the performance of the trackers.

The tests that we employ include tracking objects under varied noise conditions (using SNR test measure), tracking objects captured at varied frame rates, tracking using varied number of non-rigid shape parameters to account for contour deformation, and using varied number of control points to represent object shape. The result of each of the trackers was measured against 4 performance measures. Namely: the contour distance error, the contour origin error, the deformable shape parameter error, and the SNR. The description of test methods and performance measures employed are given in sections 8.4 and 8.5.

This chapter is organized as follows: Section 8.2 describes the CONDESATION algorithm in brief. Section 8.3 describes the Baumberg's tracker in brief. Section 8.4 describes the performance comparison methods used to compare the tracker performance. Section 8.5 gives the results obtained, and Section 8.6 gives a discussion and interpretation on the results presented. Finally section 8.7 provides the conclusion.

8.2 The Condensation Algorithm

The Condensation algorithm [104] is based on the factored sampling method [102], but extended to apply iteratively to successive images in a sequence. The following diagram (Fig. 8.1) displays the principle of the algorithm recursion, and Fig. 8.2 shows one iteration of the Condensation tracker. Details of the algorithm are given in Appendix F.1 (see also [102, 104] for complete details).

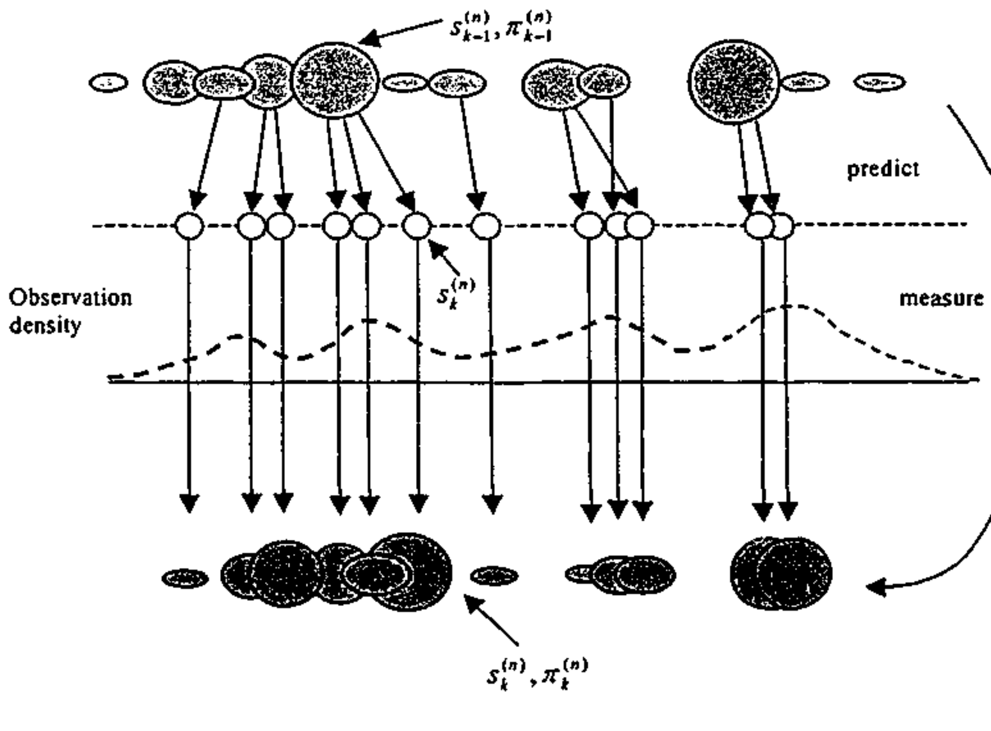


Figure 8.1: One time step in the CONDENSATION algorithm.

Iterate

From the 'old' sample-set $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}), n = 1,\ldots,N\}$ at time step (*t-1*), construct a 'new' sample set $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}), n = 1,\ldots,N\}$ for time *t*.

Construct the *n* th of *N* new samples as follows:

1. Select a sample set $\mathbf{s}_t^{\prime(n)}$ as follows:
   (a) generate a random number $r \in [0,1]$, uniformly distributed.
   (b) Find by binary subdivision, the smallest *j* for which $c_{t-1}^{(j)} \geq r$
   (c) set $\mathbf{s}_t^{\prime(n)} = \mathbf{s}_{t-1}^{(j)}$

2. Predict by sampling from

   $$p(\mathbf{x}_t \mid \mathbf{x}_{t-1} = \mathbf{s}_t^{\prime(n)})$$

   to choose each $\mathbf{s}_t^{(n)}$. For instance, in the case that the dynamics are governed by a linear stochastic differential equation, the new sample value may be generated as: $\mathbf{s}_t^{(n)} = \mathbf{A}s_t^{\prime(n)} + B\mathbf{w}_t^{(n)}$ where $\mathbf{w}_t^{(n)}$ is a vector of standard normal random variates, and BB^T is the process noise covariance.

3. Measure and weight the new position in terms of the measured features $\mathbf{z}_t$:

   $$\pi_t^{(n)} = p(\mathbf{z}_t \mid \mathbf{x}_t = \mathbf{s}_t^{(n)})$$

   then normalize so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$
   where

   $$c_t^{(0)} = 0,$$

   $$c_t^{(n)} = c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1,\ldots,N)$$

Once the *N* samples have been constructed: estimate, if desired, moments of the tracked position at time step *t* as

$$\varepsilon[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f(\mathbf{s}_t^{(n)})$$

obtaining, for instance, a mean position using $f(\mathbf{x}) = \mathbf{x}$.

Figure 8.2: One Iteration of the CONDENSATION tracker [104].

8.3 Baumberg's Tracker

Baumberg and Hogg's tracker [11, 10] was primarily devised for tracking pedestrians, but the principle of the base method can be applied to track any object contour. An overview of tracking mechanism is shown in Fig. 8.3 (The tracker details are given in Appendix F.2).

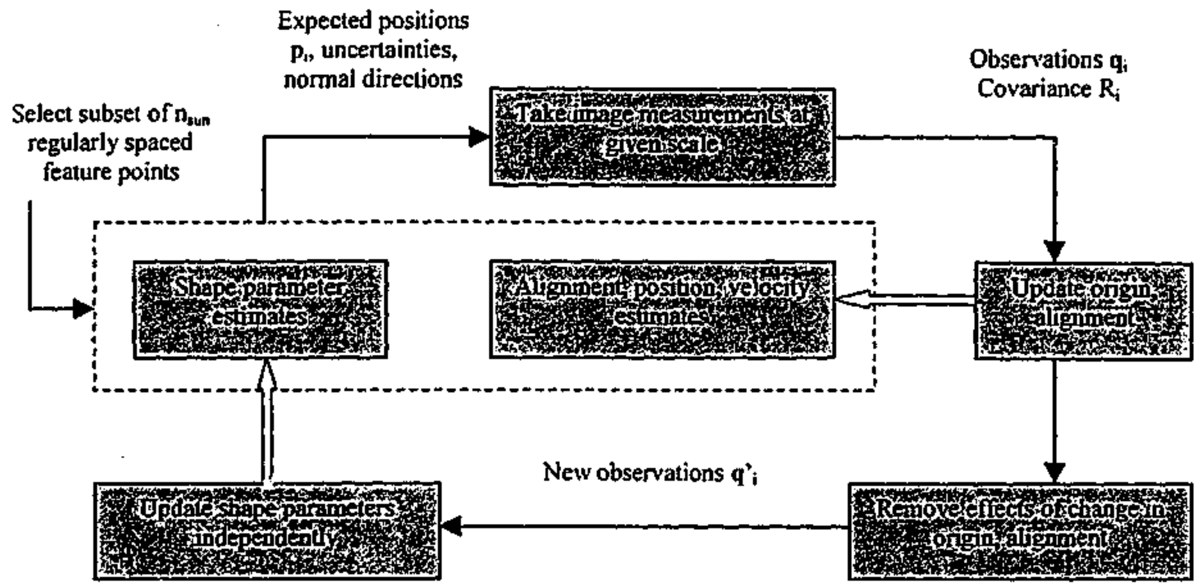

Expected positions
p_i, uncertainties,
normal directions
Select subset of n_{sub}
regularly spaced
feature points
Take image measurements at given scale
Observations q_i
Covariance R_i
Shape parameter estimates
Alignment position, velocity estimates
Update origin, alignment
New observations q'_i
Update shape parameters independently
Remove effects of change in origin, alignment

Figure 8.3: Diagram illustrating the tracking mechanism of Baumberg's tracking filter [10].

8.4 Performance Measures

8.4.1 Contour Distance Test

A simple distance metric to measure the distance between two sets of landmarks $\mathbf{x} = (x_i, y_i)$ and $\mathbf{x}' = (x'_i, y'_i)$ can be given by,

$$\begin{aligned} f(\mathbf{x}, \mathbf{x}') &= |\mathbf{x} - \mathbf{x}'| \\ &= \left(\sum_{i=0}^{N-1} (x_i - x'_i)^2 + (y_i - y'_i)^2 \right)^{1/2} \end{aligned} \tag{8.1}$$

Unfortunately for contours represented by B-splines, this measure does not take into account the B-spline metric parameters [60]. For 2 contours represented by B-splines, a better distance metric can be formulated by including the B-spline metric matrix as given in [10, 60].

Given two cubic B-splines P(s) and P'(s) defined by their N control points (x_i, y_i) and (x'_i, y'_i), a more accurate error metric d, measures the difference between corresponding points on each spline, sampled densely and uniformly over the parametric curves. The distance metric is given by,

$$\begin{aligned} d(\mathbf{x},\mathbf{x}') &= \left(\int_0^N |\mathbf{P}(s) - \mathbf{P}'(s)|^2 ds\right)^{1/2} \\ &= \left(\int_0^N \sum_{i=0}^{N-1} ((x_i - x'_i)B_i(s))^2 ds + \int_0^N \sum_{i=0}^{N-1} ((y_i - y'_i)B_i(s))^2 ds\right)^{1/2} \end{aligned} \qquad (8.2)$$

where $B_i(s)$ is the cubic B-spline basis matrix.

Equation (8.2) simplifies to the following form [10]:

$$d(\mathbf{x},\mathbf{x}') = \left[(\mathbf{x}-\mathbf{x}')^T \mathbf{J}(\mathbf{x}-\mathbf{x}')\right]^{1/2}$$

where **J** is the $2N$x$2N$ symmetric metric matrix (as described in chapter 7, section 7.4).

There is a unique inner product associated with this metric given by,

$$\langle \mathbf{x},\mathbf{x}' \rangle = \mathbf{x}^T \mathbf{J} \mathbf{x}'$$

such that

$$d(\mathbf{x},\mathbf{x}') = \langle \mathbf{x}-\mathbf{x}', \mathbf{x}-\mathbf{x}' \rangle^{1/2} = \left[(\mathbf{x}-\mathbf{x}')^T \mathbf{J}(\mathbf{x}-\mathbf{x}')\right]^{1/2}$$

We define the distance error as the average of d across the image sequence (F frames), which is given by,

$$Distance\_error = \frac{1}{F}\sum_{k=1}^{F} | d(\mathbf{x},\mathbf{x}') |_k \qquad (8.3)$$

8.4.2 Object Origin Test

The object origin is simply the center of gravity of a closed contour, which is calculated for the object of interest (actual and tracked) at each frame (k) of a sequence, then the difference of the origin error (at each frame) is averaged over the number of frames (F). The origin error is defines as,

$$Origin\_error = \frac{1}{F}\sum_{k=1}^{F}\left[(O_x^{actual} - O_x^{tracked})_k^{\ 2} + (O_y^{actual} - O_y^{tracked})_k^{\ 2}\right]^{1/2} \tag{8.4}$$

with

Actual object origin $O_k^{actual} = (O_x^{actual}, O_y^{actual}) = \left(\frac{1}{N}\sum_{i=1}^{N} X_i^{actual},\ \frac{1}{N}\sum_{i=1}^{N} Y_i^{actual}\right)$

Tracked object origin $O_k^{tracked} = (O_x^{tracked}, O_y^{tracked}) = \left(\frac{1}{N}\sum_{i=1}^{N} X_i^{tracked},\ \frac{1}{N}\sum_{i=1}^{N} Y_i^{tracked}\right)$

where $X_i^{tracked}, X_i^{actual}$ are the B-spline vectors (with *N* control points) for the tracked contour and the actual contour respectively. A low value of origin error will reveal the tracked contour-centroid is close to the original contour-centroid in terms of position. The origin error can be used as a secondary measure to the distance error measure.

8.4.3 Shape Deformation Test

The shape deformation test is a test measure to assess the deviation of non-rigid shape variation from a mean shape. The quantity reveals how much the object shape at *k*-th frame has deformed from the mean shape. In our analysis we have devised an error measure for the difference in non-rigid shape changes between the tracked shape and the actual shape in terms of the Mahalanobis Distance (MD) measure (All affine changes of shape are disregarded for this test). The non-rigid (deformable) shape parameter error is calculated for the object at every frame (*k*), then the error between the actual and tracked MD is averaged over the number of frames (*F*). This is given by,

$$NRSPE = \frac{1}{F}\sum_{k=1}^{F}\left|MD_{actual} - MD_{tracked}\right|_k \tag{8.5}$$

Where NRSPE stands for: Non Rigid Shape Parameter Error. The Mahalanobis distance measure for actual and tracked contours are given by [52, 87],

$$MD_{actual} = \left(\sum_{i=1}^{m}\frac{b_i^2}{\lambda_i}\right)^{1/2}, \quad MD_{tracked} = \left(\sum_{i=1}^{m}\frac{\hat{b}_i^2}{\lambda_i}\right)^{1/2} \text{ respectively.}$$

Where $\lambda_i, b_i, \hat{b}_i$, are the eigenvalue, deformable (non-rigid) shape parameter for the actual contour, and the deformable shape parameter for the estimated (tracked) contour, corresponding to the *i*-th principal vector respectively. *m* is the number of principal components considered for non-rigid object tracking. It should be noted that to evaluate *MD*, the object in *k*-th frame has to be translated, scaled and rotated (if required) to align with the mean shape. This process ensures that only the deformable shape changes of the object are measured (disregarding changes in translation, scaling and rotation).

8.4.4 Signal to Noise Ratio (SNR) Test

The SNR performance evaluation is a B-spline independent image based method that uses the 'SNR out' measure for tracking performance (similar to that reported in [10]). To evaluate the performance of the trackers under varied noise, a 'SNR in' measure can also be formulated. Both these measures can be determined as explained in the following sections.

8.4.4.1 Measuring the Accuracy of Tracking

An additional performance measure employed to assess the accuracy of the tracking process (ie. The accuracy of shape, position and orientation of the tracked contour) is an image processing based measure. Thus the error measure is independent of the parameterization of the contour representation [10]. The contour resulting from the tracking process is rendered flat filled in the 'foreground' color (moving object colored with white) into the image I_{track}.

The tracking process is 'local' so that the signal far from the object is never sampled. Hence, in this case, it is more appropriate to measure the signal in terms of the area of 'foreground' pixels in the ground truth image. The signal and noise are calculated using the following quantities [10].

$$\begin{aligned} signal &= 2\sum_{images}\sum_{x,y}[I_{ref}(x,y)]^2 \\ noise &= \sum_{images}\sum_{x,y}[I_{ref}(x,y) - I_{track}(x,y)]^2 \end{aligned} \tag{8.6}$$

where $I_{ref}(x,y)$ is the pixel value at (*x*, *y*) for the ground truth image. The pixel value for a 'background' pixel is 0. The scale factor of 2 in the signal value was chosen so that a SNR of 0 (ie. signal = noise) would occur if the tracker silhouette consisted of a shape of the same area as the ground truth shape but inaccurately placed so that there is no overlap between the two [10]. This is the 'worst case' scenario where the tracker has completely failed to track the object. The output SNR (in dB) denoted as SNR_{out} is calculated by using the following equation.

$$SNR_{out}(dB) = 10\log\left[\frac{signal}{noise}\right] \tag{8.7}$$

An example for finding 'SNR out' is illustrated in Fig. 8.4 for the 3 trackers considered.

8.4.4.2 Adding Noise to Input Test Sequence

Noisy images were generated by adding Gaussian noise to the test image sequence. This type of noise was chosen to test the robustness of the system, for several reasons. Firstly, the noise added (particularly at high levels) can't be thresholded out easily. Secondly, the noise process will result in significant errors in contour measurements over whole sections of the curve. Hence these noisy images are suitable for a rigorous test of the tracking system. Some corrupted images are shown in Fig. (8.5). It can be seen that the silhouette shape can be disrupted by the noise, and a conventional non-model based approach such as the 'snake' of Kass et al [113] would be unable to recover the object shape correctly.

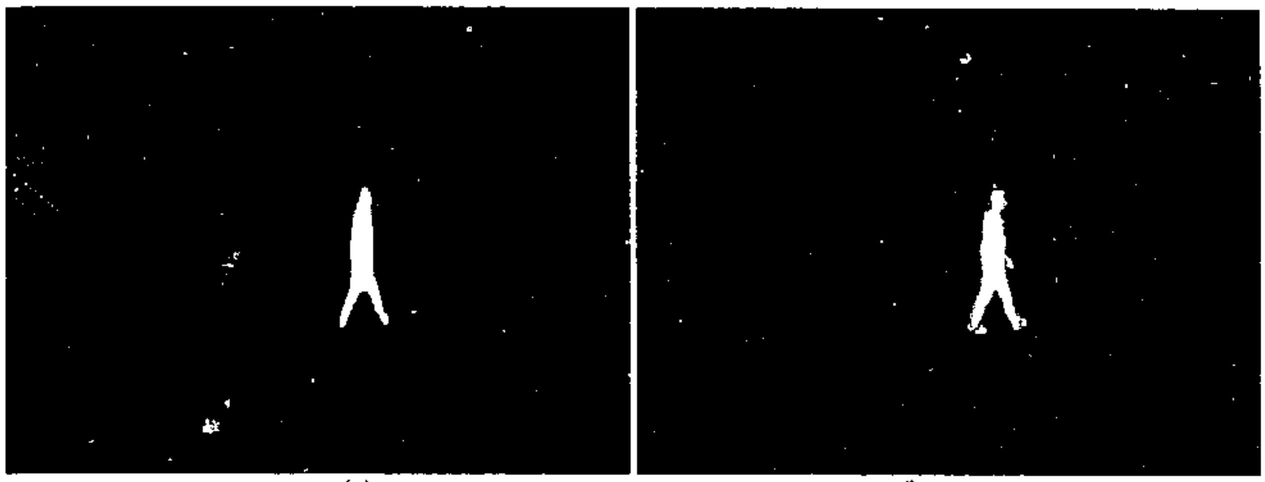
The signal to noise ratio (SNR_{in}) of the noisy images is calculated over the test image sequence using

$$SNR_{in}(dB) = 10\log\left[\frac{signal}{noise}\right] \tag{8.8}$$

with

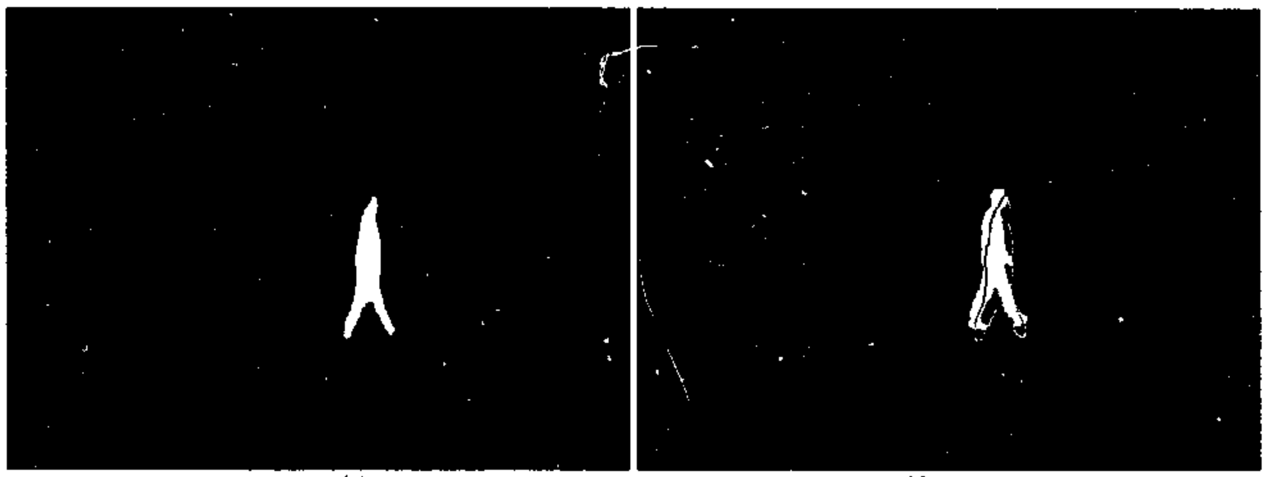
$$\begin{aligned} signal &= \sum_{images}\sum_{x,y}[I_{ref}(x,y) - I_0]^2 \\ noise &= \sum_{images}\sum_{x,y}[I_{ref}(x,y) - I'(x,y)]^2 \end{aligned} \tag{8.9}$$

where $I_{ref}(x,y)$ is the pixel value at (x, y) for the ground truth image and $I'(x,y)$ is the corresponding pixel in the corrupted image (the noisy image is binarised for 'SNR in' calculation). The constant I_0 is set to halfway between the 'background' and 'foreground' pixel values, so that a patch of foreground and a patch of background both have the same signal strength, thus ensuring the SNR is independent of the relative image and object size.



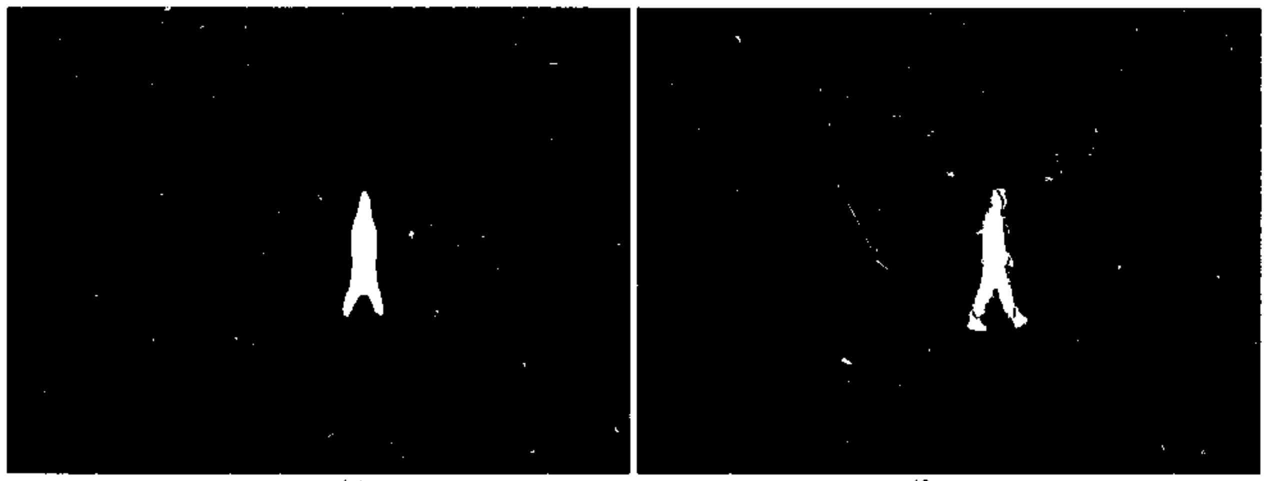
(a) (b)

Using CONT-IMM Tracker



(c) (d)

Using CONDENSATION Tracker



(e) (f)

Using Baumberg Tracker

Figure 8.4: An example of SNR output results. (a) Contour tracked by CONT-IMM flat filled (b) The true object contour flat filled with tracked contour superimposed. (c) Contour tracked by CONDENSATION flat filled (d) The true object contour flat filled with tracked contour superimposed (e) Contour tracked by Baumberg tracked flat filled (f) The true object contour flat filled with tracked contour superimposed.

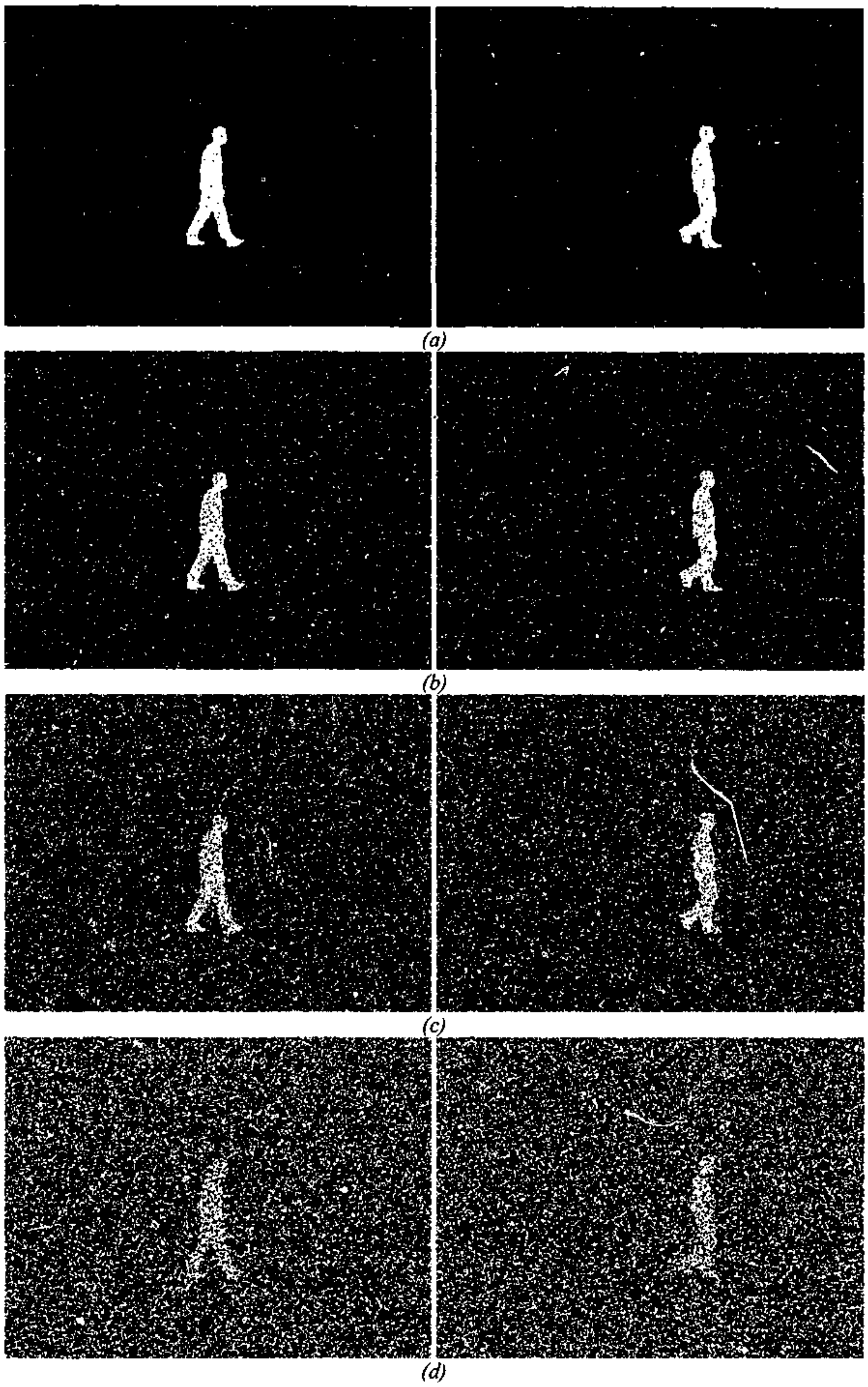

(a)
(b)
(c)
(d)

Figure 8.5: Effects of adding artificial noise (binarised for SNR input calculation). With Gaussian noise variance (a) at 50 (b) at 75 (c) at 100, and (d) at 130.

8.5 Results

8.5.1 Tracker Implementation Method

The CONT-IMM tracker was implemented as described in chapter 7. The CONDENSATION algorithm was implemented (as described in Appendix F.1) using 1000 samples per iteration. A second order dynamic motion model was applied to the CONDENSATION translation parameters. The deformable changes were assumed to follow a first order Markov process (for full details of CONDESATION implementation refer to [102]). The Baumberg's tracker was implemented as outlined in Appendix F.2 (for full details refer to [10]).

8.5.2 Frame Rate Test

The frame rate test method was devised to analyze the performance of trackers at varied frame rates. In order to carry out the experiment, test image sequences of a walking man was captured at four different frames rates: 5, 10, 20 and 30 frames/second. Each tracker was allowed to track the man independently at each frame rate. For each test, the B-spline based error measures and the 'SNR out' performance measures were calculated. The results obtained are tabulated in Table (8.1) and illustrated in Fig. (8.6).

8.5.3 Noise Test

This is a test to evaluate the trackers' capability to track objects under noisy condition. The test sequence captured at 10 frames/sec was corrupted with Gaussian noise at various levels. At each noise level, the trackers were applied to track the walking man. For each test the B-spline based error measures and the 'SNR out' values were calculated. Results obtained are tabulated in Tables (8.2)-(8.6), quantitatively illustrated in Fig. (8.7), and qualitatively displayed in Figs. (8.8)-(8.9).

8.5.4 Control Point Test

The object of interest is represented by varying number of control points. We tested and compared the performance of each tracker by employing control points ranging from 12 – 64. Since B-spline error measures are unreliable for comparing performance for this test (see details in section 8.6) only the 'SNR out' test was carried out, which is shown in Fig. (8.10). The results are tabulated in Table (8.7) and the qualitative results displayed in Fig. (8.11).

8.5.5 Shape Deformation Test

The non-rigid (deformable) shape parameter test is to assess the error in deformable shape changes of objects (between the actual and the tracked shape). The number of shape parameters (in other words the number of principal components used) employed has a direct impact on the quality of tracked shapes. We carried out experiments by using 1, 2, 3, 4, 5, 10, 15, 20, and 25 non-rigid shape parameters to account for deformable shape variations of the object contour. The results obtained are tabulated in Table 8.8 and illustrated quantitatively by Fig. 8.12 and qualitatively by Figs. (8.13). It should be *noted* that for this experiment the deformable shape parameter error quantity (equation (8.5)) was assessed by averaging the Mahalanobis distance by the number of shape parameters used (for other tests discussed in this chapter, this process is not required).

8.6 Discussion

In this section we discuss the results obtained in section 8.5. We interpret the results under the four different performance test carried out.

8.6.1 Tracker Performance Under Varying Frame Rates

All 3 trackers were employed to track an indoor walking person, where the moving person was captured at different frame rates. The purpose of the test was to analyze the robustness of the trackers when the inter- frame shape differences are varied.

The distance, the origin, and the shape parameter error measures clearly show that the CONT-IMM and the BAUMBERG trackers are less sensitive to frame rate changes (though the CONT-IMM gives much smaller errors, Fig. (8.6)). The CONDENSATION tracker is observed to be sensitive to changing frame rates. Particularly at lower frame rates, the CONDENSATION gives poor quality results, but at higher rates (at video rates) the performance is remarkable, and does approach the performance of CONT-IMM tracker. The reason for poor quality results for CONDENSATION is that, one of the assumptions for this algorithm is to have small inter-frame shape changes (particularly for the measurement process to be effective [24]), which is a reasonable assumption at high frame rates (eg: 30 frames/sec.).

Focussing on the SNR test results, the CONT-IMM provides an average 'SNR out' of around 9.5 – 9.75 (db) for the range of frame rates considered (5 – 30 frames/sec), where as the 'SNR out' for the BAUMBERG tracker varied between 7.75 – 8.00 (db). The CONDENSATION SNR output varied from about 4.5 (db) at 5 frames/sec to around 9.5 (db) at rates of 30 frames/sec. The empirical observations suggest that the CONT-IMM method give the best frame rate results followed by

CONSENSATION (at high frame rates) and BAUMBERG trackers. It should be noted that the observations obtained from B-spline based error measures are consistent with the SNR output results.

8.6.2 Tracker Performance Under Varying Noise Condition

This test is a method to evaluate the performance of the trackers under noisy environment. Uncorrelated noise is added (Gaussian distributed) to each frame of a sequence prior to tracking. The performances of the trackers are assessed at varied noise levels using the performance measures described. The results again show that the CONT-IMM tracker gives the best result under noise followed by CONDENSATION and the BAUMBERG trackers. Remarkably all 3 trackers perform well up to a noise variance level of around 50. At very high noise levels, the performance of all 3 trackers starts to deteriorate. This is because each tracker has its own mechanism to eliminate spurious measurements by employing some noise thresholding (filtering) techniques, but such techniques break down at high noise levels as evident from the results. The poor performances at high noise levels are directly attributed to obtaining erronious measurements (for all 3 trackers), which in turn leads to poor quality track results.

An important tracking performance test not covered in this chapter is the ability of the trackers to track objects in cluttered environments. Unfortunately clutter level cannot be measured witl reasonable precision, and therefore was not considered in the series of experiments that we carried out. However, as Blake et. al. [24] demonstrated, the CONDENSATION has been shown to track well in cluttered background. This is because CONDENSATION supports multiple hypothesis of pdfs for its observation process (as discussed in Appendix F.1), and as a result is able to disregard false measurements efficiently. Baumberg tracker was also shown to be agile enough to track under short periods of clutter [10], but was prone to heavy background clutter because of high false contour measurements. CONT-IMM tracker is prone to heavy clutter due to its contour measurement process. Since CONT-IMM uses background subtraction for contour measurements, heavy clutter results in poor quality measurements being obtained, despite having mechanism to reduce noise. Incorrect measurement in turn leads to poor tracking results.

8.6.3 Tracker Performance by Varying the Number of Object Control Points

For this particular test, the spline based performance measures are not useful, because the tracked contour can only be compared with the actual contour, provided both object contours have the same number of control points. Varying the number of control points on the actual and the tracked contours gives rise to an approximation error (particularly at lower number of control points). Therefore, spline based performance measures do not reveal the true quality of the trackers' output.

In this case, only the SNR output performance was measured, which is an ideal test for this experiment. The number of control points to represent the object is varied from 16 – 64 to test the tracker robustness to control point variation. The tracked 'SNR out' is compared with the theoretical maximum 'SNR out' possible. This value (Max SNR out) is calculated by taking the actual object and approximating the contour by the number of control points considered, and then flat filling the contour with white, while the background remains black. This foreground flat filled area is then used to calculate the maximum SNR output (using Eq. (8.7)).

As can be seen from Fig. (8.10) the trackers achieve their best performance level when the number of control points are around 30 (for this particular object). By extending the control points beyond 30 brings little improvement. Therefore, striking a balance between speed of the tracker and the accuracy of the tracker, it is best to maintain the number of control points to around 30.

Observation of the result shows that the Baumberg tracker is very sensitive to the number of control points used, particularly at lower values. The CONDENATION tracker is the least sensitive among the trackers, which maintains an 'SNR out' value of around 8 db for the range of control points considered. The CONT-IMM gives the best result reaching an output SNR of around 10.5 db between 28-40 control points and 11db with 64 control points, but at lower number of control points (< 16) the performance is observed to be rather poor.

The theoretical maximum possible SNR output is a guide to show how well the trackers perform in relation to optimum expectation for the range of control points considered (Fig. 8.10). It is almost impossible for a tracker to get an SNR output anywhere near the theoretical mark. This is because, a 1 pixel displacement between the tracked object and the actual object can cause about 3-4 % of the flat filled area (object area) to be mis-aligned. This mismatch alone accounts for about 3.5 - 4 db of 'SNR out' (for the object size we considered). It is also worth noting that the SNR output is dependent on the object size, therefore only the relative SNR output values ought to be taken into account when comparing the results.

8.6.4 Tracker Performance by Varying the Number of Deformable Shape Parameters

This test is used purely for measuring the deformable shape changes, and therefore, does not take into account any affine contour shape changes (disregarding changes in translation, scaling and rotation).

Varying the number of shape parameters directly corresponds to the number of principal components (PCs) employed in tracking deformable shape changes. The training sequence that we used comprised about 750 different object shapes of moving pedestrians. Our off line analysis showed that about 90% of the shape changes can be accounted for, by using the 10 most significant principal components.

For the experiments reported here, we tested by using 1, 2, 3, 4, 5, 10, 15, 20 and 25 PCs. As can be seen from the results (Fig. 8.12), increasing the number of shape parameters beyond 10 results in very little improvement. Considering the tracker speed into account, using beyond 10 deformable shape parameters can also be computationally expensive. In terms of quality of results, the CONT-IMM provides better quality results at all levels compared with the other 2 trackers. It should be noted that the shape deformation test is model dependent, and therefore, the number of deformable parameters used for tracking can vary from object to object depending on the object shape size and complexity.

CONT-IMM (Error measures)

Frame rate	5 frames/sec	10 frames/sec	20 frames/sec	30 frames/sec
Dist. Error - μ	2.73	2.63	2.61	2.51
Dist. Error - σ	0.62	0.27	0.32	0.25
Origin Error - μ	0.93	0.95	1.07	1.11
Origin Error - σ	0.29	0.74	0.57	0.38
NRSPE - μ	6.61	5.96	5.89	5.26
NRSPE - σ	4.17	3.39	2.92	2.87
SNR out (db)	9.46	9.40	9.43	9.64

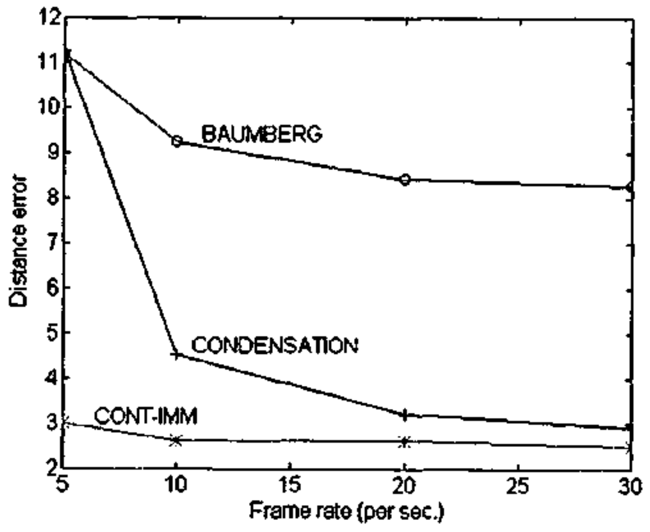
CONDENSATION (Error measures)

Frame rate	5 frames/sec	10 frames/sec	20 frames/sec	30 frames/sec
Dist. Error - μ	11.25	4.54	3.20	2.92
Dist. Error - σ	23.80	6.02	5.21	3.50
Origin Error - μ	10.15	3.66	2.35	1.90
Origin Error - σ	26.06	7.61	6.71	4.93
NRSPE - μ	11.84	7.77	6.93	6.47
NRSPE - σ	8.10	6.17	4.28	3.45
SNR out (db)	4.51	7.76	9.11	9.42

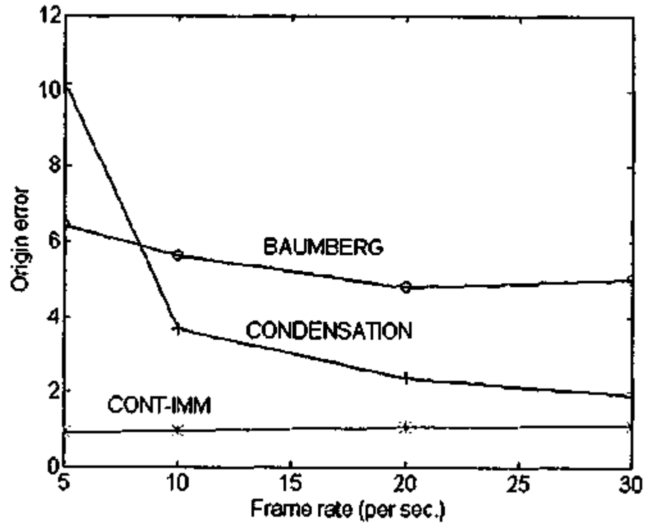
BAUMBERG (Error measures)

Frame rate	5 frames/sec	10 frames/sec	20 frames/sec	30 frames/sec
Dist. Error - μ	11.18	9.25	8.41	8.27
Dist. Error - σ	11.82	6.78	5.27	4.71
Origin Error - μ	6.42	5.60	4.80	5.04
Origin Error - σ	5.44	8.93	7.84	6.20
NRSPE - μ	18.96	18.02	17.49	16.65
NRSPE - σ	4.75	4.33	7.08	3.81
SNR out (db)	7.71	7.95	8.20	8.10

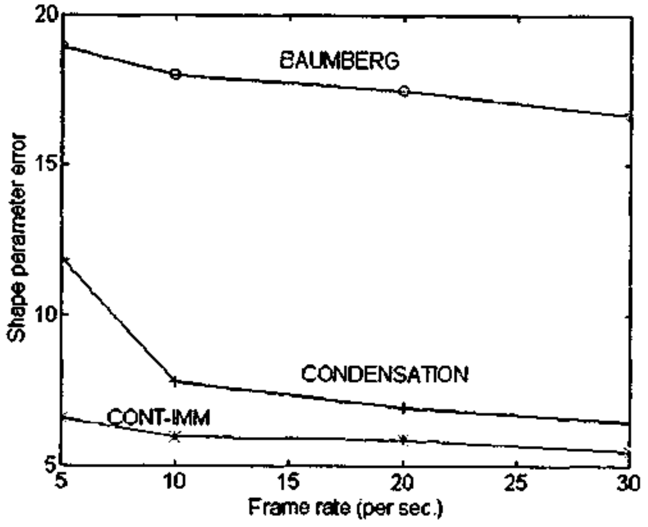
Table 8.1: Performance of the 3 trackers at varied frame rates. The symbols μ and σ indicates the mean error and variance over the frame length respectively (in pixels). NRSPE refers to Non-Rigid Shape Parameter Error.



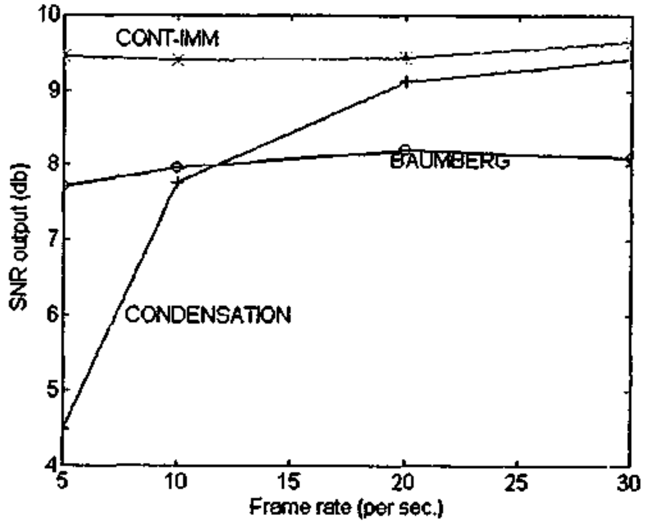
(a)



(b)



(c)



(d)

Figure 8.6: Frame rate test for the 3 trackers. The error quantities are measured in pixels. (a) Peromance using the distance error measure. (b) Performnace using the origin error test. (c) Performance using the non-rigid shape parameter error test. (d) Performance using the tracked output SNR (db). All 4 tests suggest that the CONT-IMM tracker outperforms the other two trackers.

Noise Variance	Distance error mean	Distance error var.	Origin error mean	Origin error var.	NRSPE mean	NRSPE var.
0	2.99	0.26	0.88	0.35	5.70	4.50
5	3.21	0.41	1.25	1.06	7.98	3.06
10	3.37	0.36	1.22	0.60	9.15	4.30
15	3.10	0.34	0.98	0.52	8.08	5.43
20	3.45	1.12	1.12	0.69	8.92	6.43
25	3.08	0.26	0.98	0.40	7.96	4.69
30	3.31	0.42	1.19	0.79	8.84	3.05
35	3.42	0.39	1.04	0.46	9.98	5.04
40	3.41	0.50	1.04	0.64	10.28	8.20
45	3.58	0.94	0.99	0.53	10.47	5.98
50	4.20	2.07	1.06	0.58	12.70	11.60
55	3.93	1.97	0.97	0.36	11.62	10.21
60	4.22	2.29	1.08	0.63	11.76	9.46
65	4.96	2.56	0.88	0.30	14.73	22.13
70	4.88	2.56	1.15	0.57	15.08	11.92
75	5.85	4.44	1.17	0.67	15.79	15.91
80	5.88	2.13	0.87	0.62	16.00	10.90
85	6.07	4.90	0.98	0.34	16.01	16.70
90	8.41	13.41	1.24	0.53	19.50	60.84
95	9.77	20.49	1.02	0.72	23.80	41.80
100	12.68	16.70	0.99	0.48	28.60	70.44

Table 8.2: CONT-IMM Tracker performance under varying noise. NRSPE refers to Non-Rigid Shape Parameter Error.

Noise Variance	Distance error mean	Distance error var.	Origin error mean	Origin error var.	NRSPE mean	NRSPE var.
0	5.26	6.12	3.71	7.75	9.86	8.31
5	6.48	7.28	3.68	6.48	11.07	8.97
10	6.53	7.42	3.68	6.53	11.75	10.98
15	6.53	7.37	3.76	6.53	11.33	10.06
20	6.65	8.65	3.70	6.92	11.63	15.24
25	6.46	7.66	3.67	6.81	11.10	11.18
30	6.65	7.57	3.76	6.84	11.70	11.21
35	6.75	7.00	3.67	7.00	12.40	10.21
40	6.72	8.12	3.80	7.72	12.43	13.12
45	6.83	7.56	3.67	7.17	13.39	12.48
50	7.56	9.28	3.95	7.40	14.20	18.39
55	7.36	8.74	3.90	6.76	14.41	19.47
60	7.21	7.21	3.80	5.80	14.25	14.75
65	8.09	6.52	3.94	5.06	16.60	24.75
70	8.01	8.54	4.10	6.50	16.82	17.28
75	8.68	6.91	3.92	5.41	17.75	16.82
80	8.72	6.78	4.42	7.45	18.30	13.30
85	9.39	8.92	4.33	6.47	18.87	22.85
90	11.69	37.27	4.67	6.73	22.50	60.37
95	12.98	24.73	4.83	8.84	22.57	44.53
100	16.84	19.95	6.93	10.17	30.66	75.95

Table 8.3: CONDENSATION Tracker performance under varying noise. NRSPE refers to Non-Rigid Shape Parameter Error.

Noise Variance	Distance error mean	Distance error var.	Origin error mean	Origin error var.	NRSPE mean	NRSPE var.
0	9.04	7.96	5.54	8.93	18.02	4.33
5	8.92	7.25	5.21	6.01	18.42	4.18
10	8.58	6.50	4.96	7.40	17.60	4.46
15	8.78	7.61	5.33	6.70	17.70	3.96
20	8.92	7.79	5.51	7.28	17.86	3.54
25	8.77	7.76	5.46	8.60	17.45	4.20
30	8.84	9.12	5.40	6.98	17.80	4.82
35	8.91	9.12	5.36	7.90	17.50	4.93
40	8.61	6.60	5.50	8.28	17.56	3.68
45	8.80	8.80	5.34	6.88	17.24	6.13
50	8.92	7.63	5.44	7.98	17.54	5.06
55	8.72	5.72	5.70	11.12	17.44	5.73
60	8.96	7.51	6.08	9.07	17.71	5.26
65	9.25	9.95	5.88	8.18	17.73	5.10
70	9.61	6.05	6.67	17.41	18.36	5.77
75	9.50	6.12	6.46	9.52	19.00	6.00
80	10.17	8.27	6.48	12.86	19.86	5.17
85	10.48	11.62	6.78	9.52	19.78	3.67
90	12.51	23.97	6.71	9.51	22.31	8.47
95	12.31	13.86	7.11	10.25	21.16	5.78
100	13.27	6.34	7.30	22.84	26.00	7.00

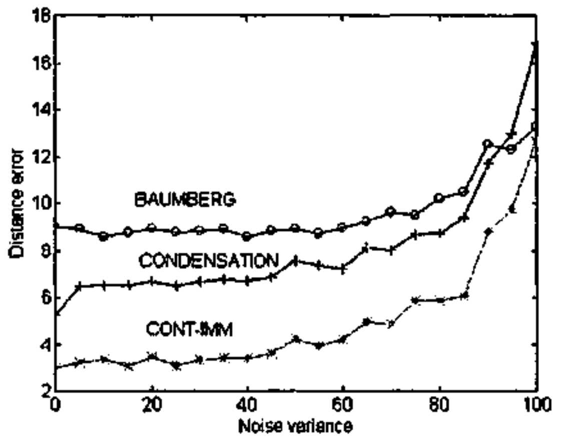
Table 8.4: BAUMBERG Tracker performance under varying noise. NRSPE refers to Non-Rigid Shape Parameter Error.

Noise variance	CONT-IMM SNR out (db)	CONDENSATION SNR out (db)	BAUMBERG SNR out (db)
0	10.17	7.90	7.98
10	10.20	8.08	8.01
20	10.55	7.97	8.05
30	10.65	8.15	7.95
40	10.27	7.82	7.88
50	9.70	7.15	7.89
60	9.74	7.70	7.52
70	9.24	7.45	7.19
80	8.60	6.97	6.80
90	7.38	6.52	5.79
100	5.90	3.40	2.87

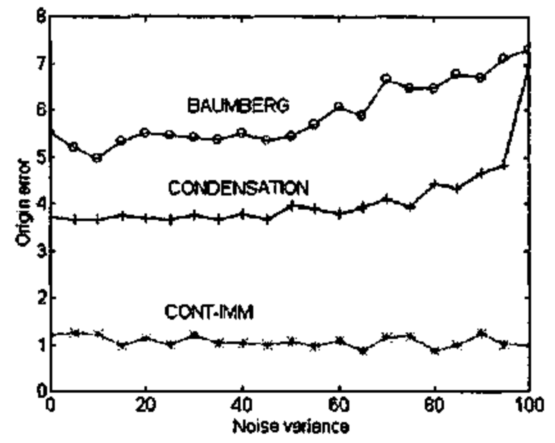
Table 8.5: SNR output achieved by the 3 trackers with varying noise.

Noise variance	SNR in (db)
10	22.47
20	16.29
30	12.71
40	10.18
50	8.23
60	6.64
70	5.29
80	4.13
90	3.14
100	2.26

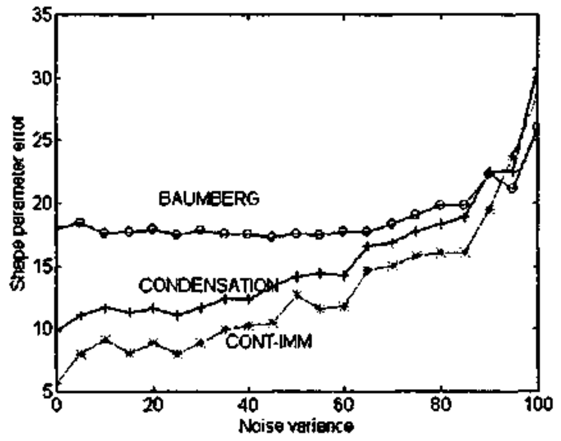
Table 8.6: Noise variance correspondence to SNR input.



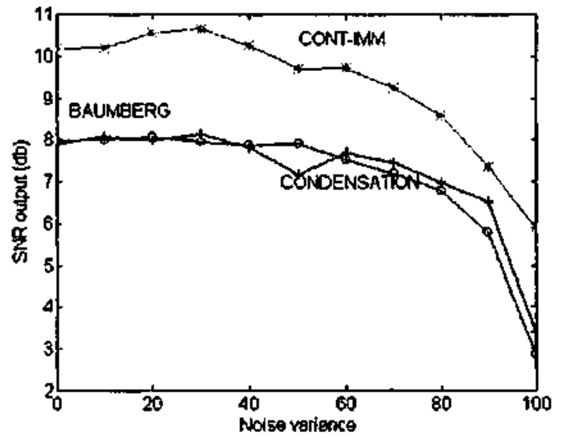
(a)



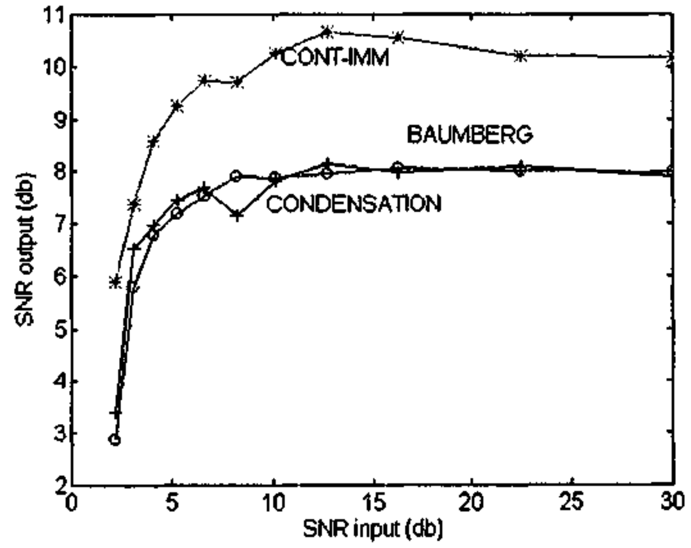
(b)



(c)



(d)



(e)

Figure 8.7: Noise performance test conducted by adding artificial noise (uncorrelated) to the test image sequence. (a) Distance error test result. (b) Origin error test result. (c) Non-rigid shape parameter error test result. (d) SNR output result. (e) SNR input versus SNR out results (see text for details). The results reveal that upto a noise variance level of 30 (~5db SNR in), the trackers produce tracking results as good as in a noise free environment. At variance levels greater than 50 (~10db SNR in), the performance of trackers deteriorates rapidly. The CONT-IMM tracker in general gives better quality results at all the noise levels considered than the other 2 trackers.

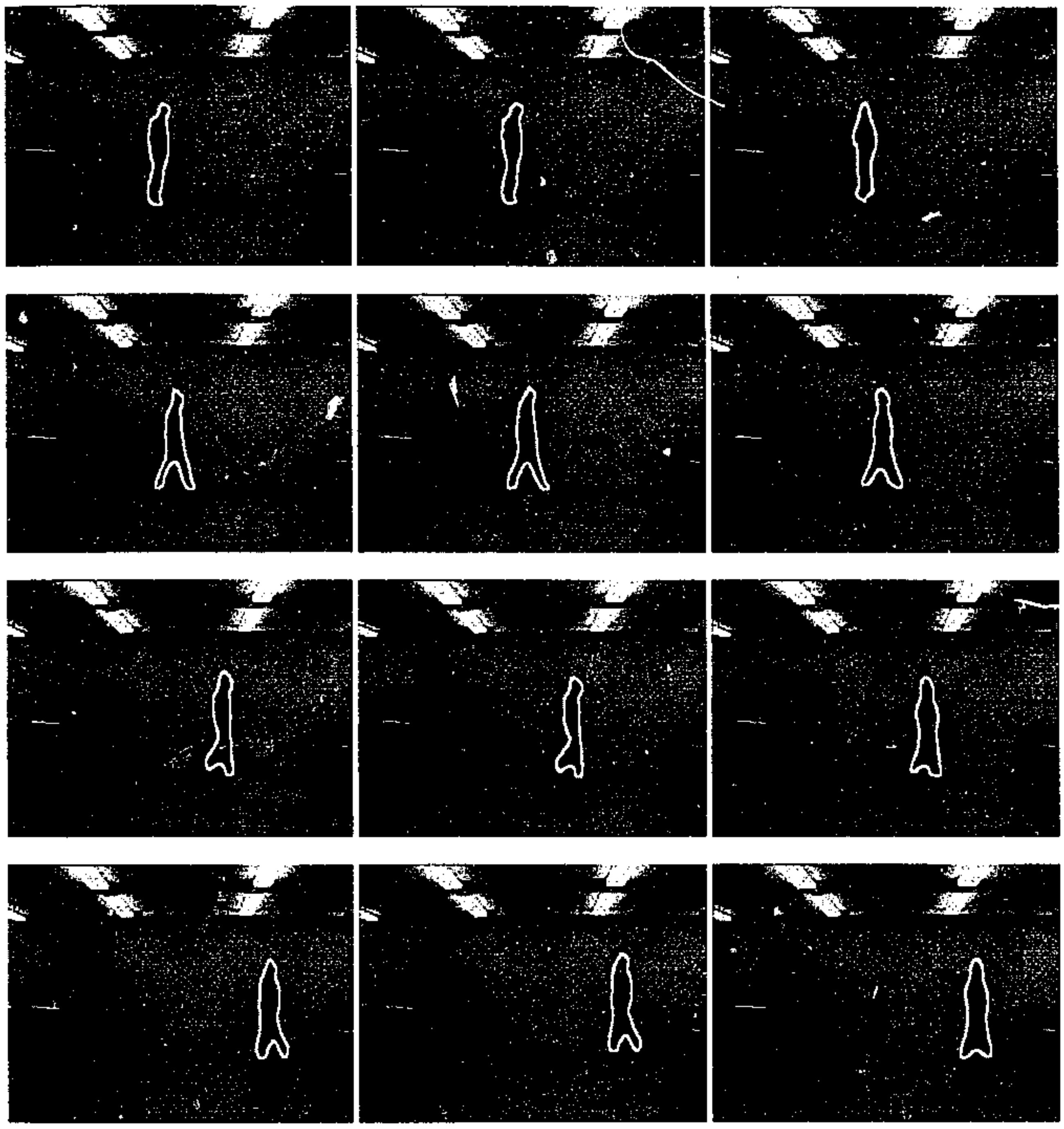


Figure 8.8: Tracking performance of the 3 trackers (with no added noise). 4 frames of a test sequence are shown with the tracked contour superimposed on top of the object (walking person). ***Figures are read column wise.*** *(a) Using CONT-IMM tracker (b) Using CONDENSATION tracker (c) Using BAUMBERG'S tracker.*

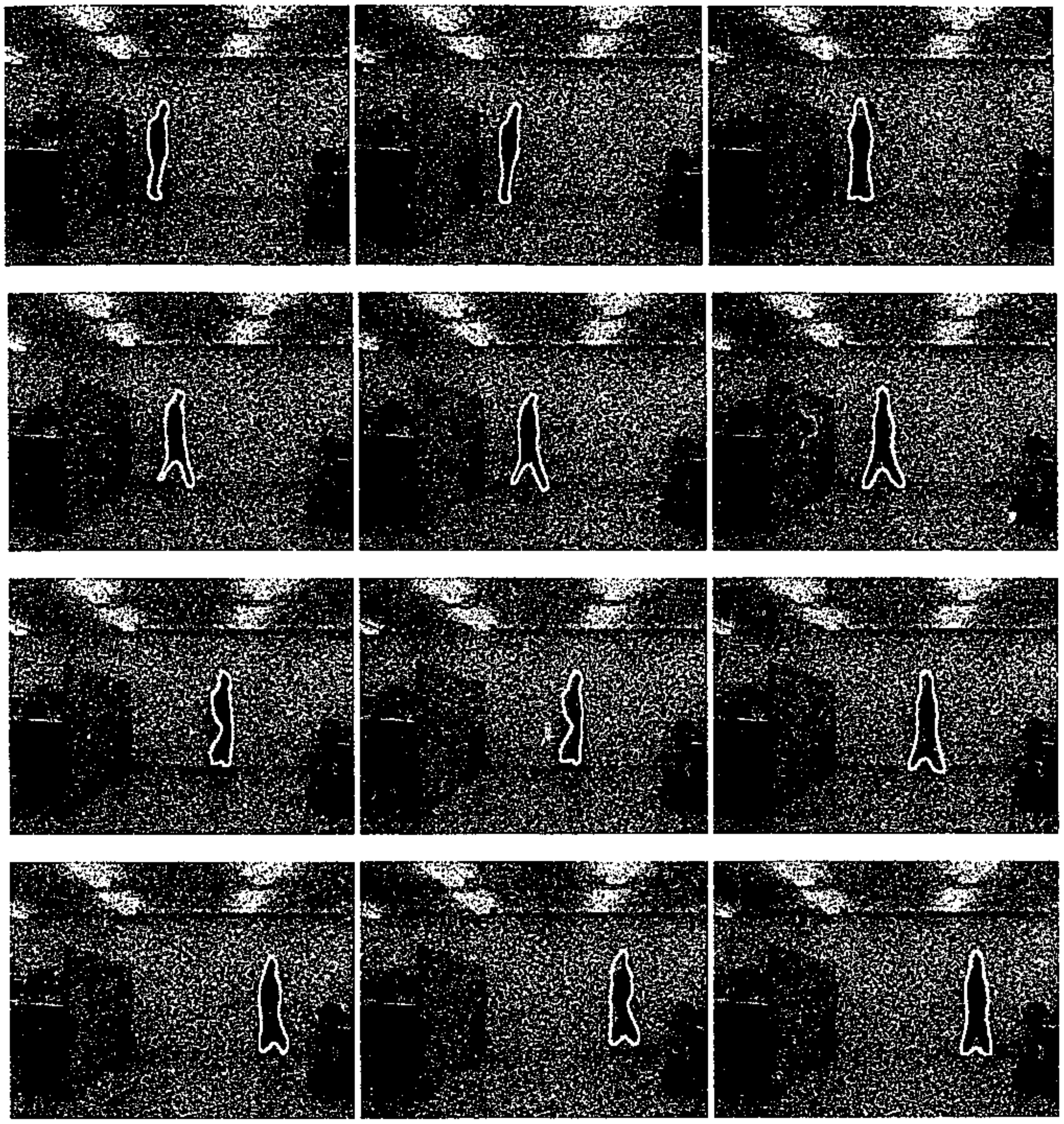


Figure 8.9: Tracking performance under noise (at variance = 80). ***Figures are read column wise.*** *(a) Using CONT-IMM tracker (b) Using CONDENSATION tracker (c) Using BAUMBERG's tracker.*

Control Points	CONT-IMM SNR out (db)	CONDENSATION SNR out (db)	BAUMBERG SNR out (db)	OPTIMUM SNR out (db)
8	7.94	7.13	3.00	8.70
12	9.21	7.64	4.64	10.41
16	9.75	7.70	5.32	11.77
20	9.86	7.77	6.96	12.70
24	10.13	7.85	7.10	13.98
28	10.47	7.80	7.50	14.51
32	10.72	7.77	7.75	15.43
40	10.50	7.88	7.77	16.90
48	11.00	7.86	7.79	17.85
56	10.54	7.86	7.80	18.60
64	11.10	7.88	7.80	19.40

Table 8.7: Performance of the 3 trackers with varying number of control point representation of the object. The optimum SNR is the maximum possible that can be achieved by any one of the trackers (see text for details).

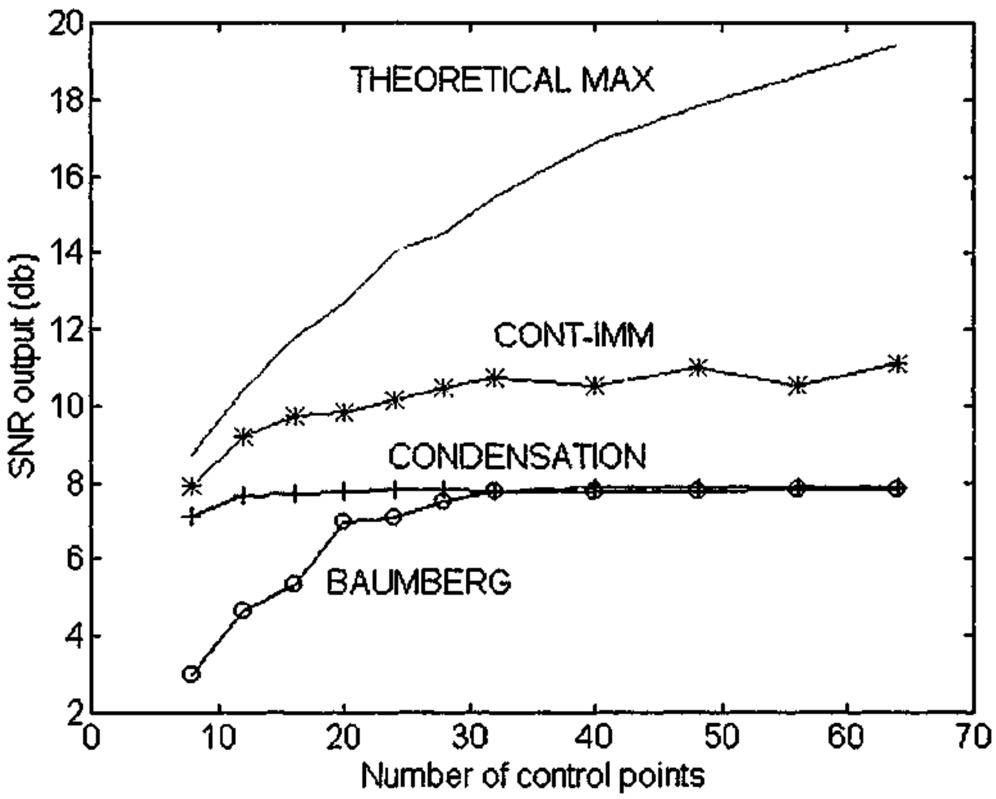
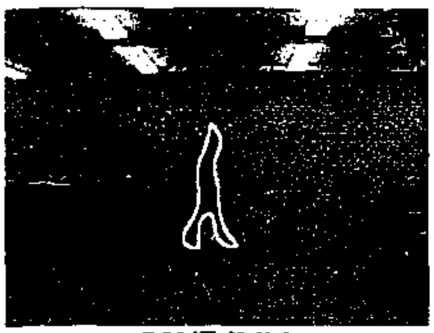
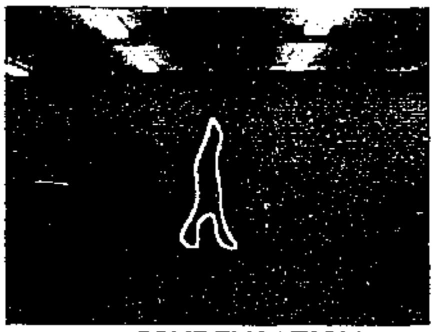
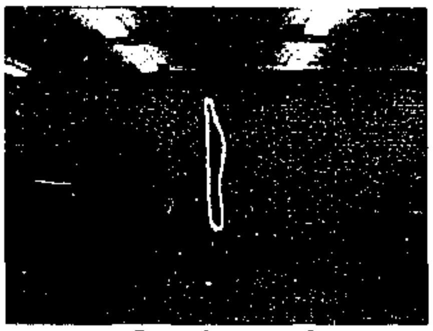
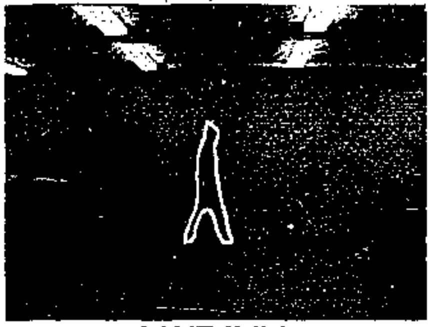


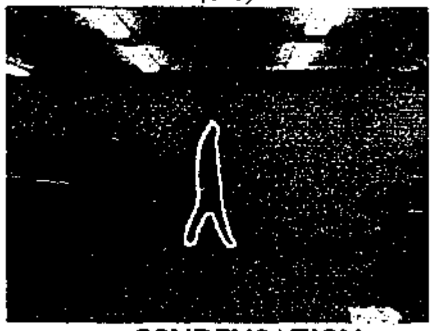
Figure 8.10: SNR output achieved by the trackers when using varied number of control points to represent the object. The theoretical maximum is the best possible tracking performance achievable (See text for detail). It is shown to indicate how well the trackers perform with varied number of control points. It is clear from the plot, that all 3 trackers achieve their best possible result when the number of contorl points are around 30. Little gain is achieved by using more than 30 control points to represent the object considered. In comparison, the CONT-IMM tracker outperforms the CONDENSATION and BAUMBERG trackers by about 2db margin.


CONT-IMM
(a-1)


CONDENSATION
(b-1)


Baumberg tracker
(c-1)


CONT-IMM
(a-2)


CONDENSATION
(b-2)

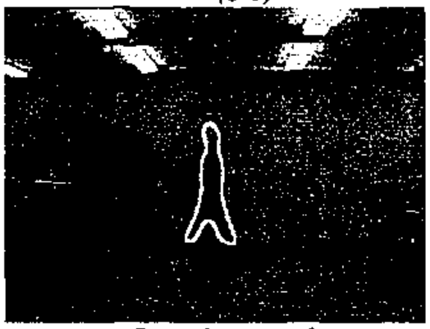

Baumberg tracker
(c-2)

Figure 8.11: Tracking performance when varying the number of control points (Frame 10 displayed). Figures are read column wise. (a1) with 16 control points using CONT-IMM, (b1) with 16 control points using CONDENSATION, (c1) with 16 control points using BAUMBERG, (a2) with 32 control points using CONT-IMM, (b2) with 32 control points using CONDENSATION, (c2) with 32 control points using BAUMBERG.

PCs	1	2	3	4	5	10	15	20	25
CONT-IMM									
Dist. Error - μ	8.38	7.78	4.38	4.07	3.67	2.81	2.14	1.88	1.61
Dist. Error - σ	1.06	1.05	0.43	0.49	0.28	0.37	0.37	0.35	0.65
Orig. Error - μ	0.87	0.94	0.85	1.00	1.24	1.11	1.03	1.01	0.91
Orig. Error - σ	0.44	0.55	0.44	0.60	0.53	0.60	0.56	0.71	0.52
NRSPE - μ	1.25	0.70	1.17	0.90	0.77	0.59	0.46	0.47	0.47
NRSPE - σ	0.80	0.20	0.21	0.14	0.08	0.02	0.01	0.02	0.01
SNR out (db)	6.79	6.93	7.10	7.60	8.22	10.10	9.98	10.76	10.80
CONDEN.									
Dist. Error - μ	5.75	5.48	5.33	5.23	5.11	4.60	4.50	4.17	4.06
Dist. Error - σ	5.64	4.61	4.78	5.01	5.25	5.74	6.28	6.54	7.10
Orig. Error - μ	3.71	3.68	3.73	3.80	3.76	3.72	3.76	3.76	3.71
Orig. Error - σ	7.22	7.38	6.91	7.42	7.64	7.14	7.36	7.22	7.75
NRSPE - μ	1.57	0.87	1.34	1.12	0.90	0.81	0.80	0.81	0.81
NRSPE - σ	1.04	0.28	0.35	0.21	0.14	0.05	0.05	0.06	0.07
SNR out (db)	6.94	6.99	7.25	7.34	7.30	7.82	7.86	8.05	8.05
BAUMBERG									
Dist. Error - μ	9.54	9.03	9.13	9.10	9.02	9.02	9.02	9.02	9.02
Dist. Error - σ	11.30	9.84	9.12	7.50	8.54	8.54	8.54	8.54	8.54
Orig. Error - μ	5.62	5.63	5.55	5.92	5.99	5.99	5.99	5.99	5.99
Orig. Error - σ	6.83	6.41	10.58	8.56	7.65	7.65	7.65	7.65	7.65
NRSPE - μ	5.23	3.35	3.22	2.67	2.41	1.66	1.51	1.47	1.47
NRSPE - σ	5.45	1.48	0.26	0.22	0.05	0.04	0.02	0.03	0.03
SNR out (db)	7.33	7.44	7.60	7.55	7.96	7.96	7.96	7.96	7.96

Table 8.8: Performance of the 3 trackers with varying number of non-rigid shape parameters (Principal Components - PCs). The symbols μ and σ indicates the mean error and variance over the frame length respectively (in pixels). NRSPE refers to Non-Rigid Shape Parameter Error.

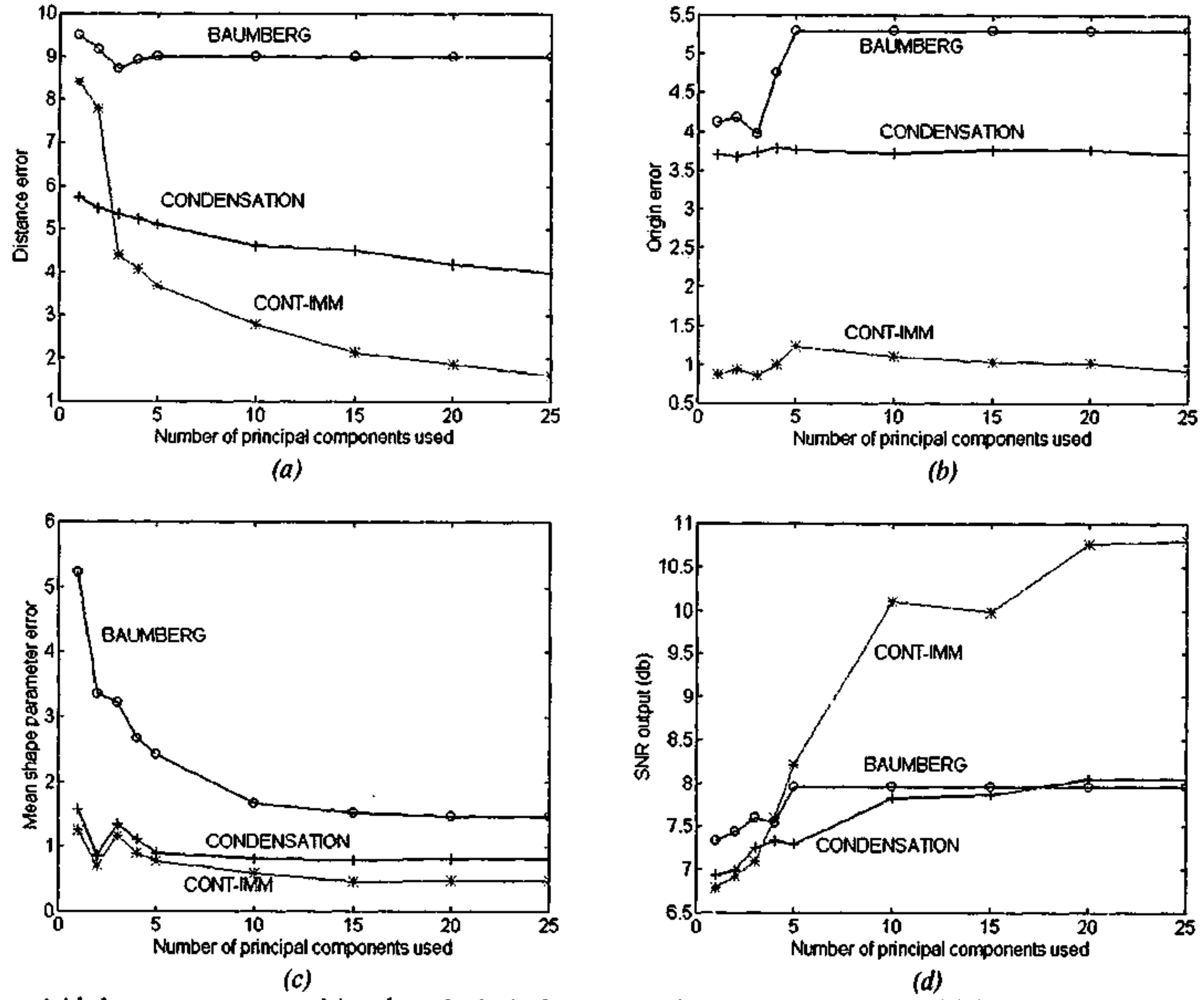


Figure 8.12: Number of non-rigid shape parameters used (number of principal components) versus error measures. (a) Performance using the distance error measure. (b) Performnace using the origin error test. (c) Performnace using the non rigid shape parameter error test. (d) Performance using the tracked output SNR (db). The results show (particularly the shape parameter error test) that by increasing the number of shape parameters, the tracking results improves (for all 3 trackers), but the relative benefit achieved by increasing the number of PCs beyond 10 is less significant.

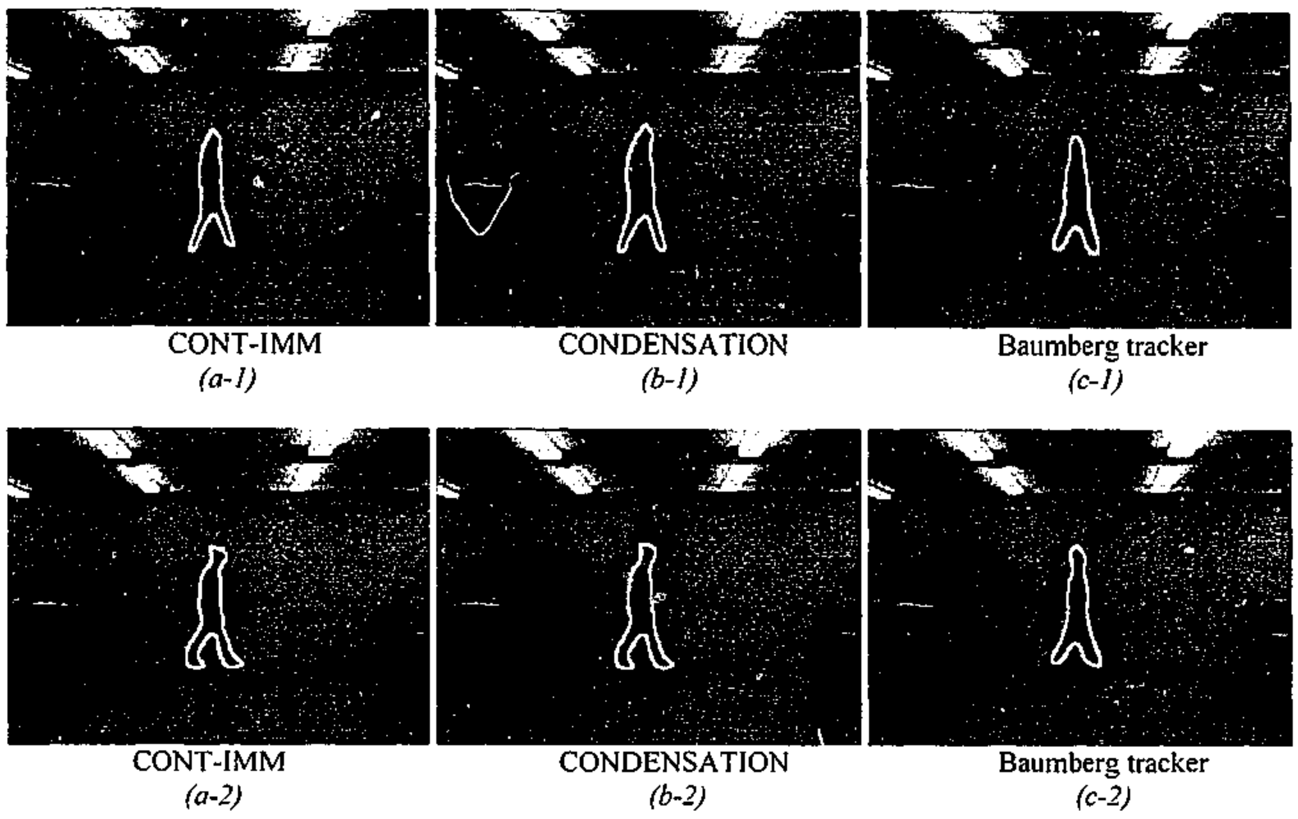


Figure 8.13: Tracking performance when varying the number of deformable shape parameters (frame 10 is displayed). Figures are read column wise (a1) with 3 parameters using CONT-IMM, (b1) with 3 parameters using CONDENSATION, (c1) with 3 parameters using BAUMBERG, (a2) with 25 parameters using CONT-IMM, (b2) with 25 parameters using CONDENSATION, (c2) with 25 parameters using BAUMBERG.

8.7 Conclusion

In this chapter we have presented empirical techniques for assessing the quality of contour tracker performance. In almost all the tests carried out, the B-spline based error measures were consistent with the SNR output results, which suggests that the performance measures are a credible representation to assess the quality of contours tracked by the three trackers concerned. The experimental methods provided can be utilized for any type of B-spline represented shape comparison test, assuming no re-parameterization of the contour control points are required. The SNR test method is a totally spline independent method, which uses only image processing techniques to evaluate performance, and therefore, can be used to analyze the output of any contour tracking algorithm with reasonable accuracy.

Though the experimental methods are restricted to the tests described in this chapter, the evaluation has revealed that the CONT-IMM method outperforms the other 2 trackers (in terms of the quality of output) in almost all the performance tests carried out.

Chapter 9

Conclusion

9.1 Summary

This thesis has examined the merits of providing a model switching ability within a visual-tracking framework. It has successfully demonstrated a model switching capability for a point feature tracker and a contour tracker, deployed on a variety of image sequences. The primary advantage of having multiple motion models is that the tracker can cope with several types of motion captured in the same image sequence, giving better quality trajectory results. The performances of the tracking algorithms (point feature tracking and contour tracking) in terms of quality and quantity have also been considered in this thesis. We have presented theoretical performance prediction methods for a point feature tracker (based on different motion models) and have provided empirical performance prediction methods for a contour tracker.

Though the performances of the tracking algorithms developed in our work have yielded promising results, there are still improvements and advancements that can be incorporated into the algorithms to enhance the tracker. In the following section we provide possible improvements that can be considered. Finally we also include possible directions for future research work.

9.2 Areas of Improvement

Although the original aim of this research was fulfilled to a great extent, there are methods presented which lack an in-depth analysis. Mainly due to the time limitation of the project, some areas have not been sufficiently addressed. In the following section we show some of the weaknesses of the methods provided, and where possible, show directions for improvement.

Chapter 3 – A fundamental issue that needs addressing when comparing the performance of corner detectors is establishing a sound ground truth. For simple objects such a task is reasonably trivial [134], but for complex scenes, as studied in this chapter, establishing ground truth is extremely difficult. Even for a human brain, to determine the best '*N*' corners from a complex scene is non-trivial. Nevertheless, a bottom line has to be drawn to decide whether a corner is 'true' or 'false' for a definite analysis. Once such a decision has been made, formulating a comparison technique becomes relatively simple. Another area that needs to be considered is the development of proper theoretical performance measures (as opposed to empirical techniques) for more stringent performance analysis. Another interesting line of work worth pursuing is to assess the performance of corner detectors under different views of the same scene. Such a process requires some form of motion model included in the analysis, which makes the comparison task much harder. For the applications we considered, a solid comparison technique was not absolutely essential, mainly because the purpose of this work was to select a good corner detector for temporal feature tracking in the subsequent phase of the project (reported in chapter 4), therefore further analysis was not considered.

Chapter 4 – The tracker (MHT-IMM) presented in this chapter was demonstrated to have the capability of automatically changing motion models. Though the results were impressive, a drawback of the tracker is the independence of corner detection and tracking processes. A coupled tracking scheme along the lines of [207, 169, 112] will enhance the quality and efficiency of the tracking algorithm. Using MHT technique for tracking has its advantages (as described in the introduction of chapter 4), but the downside is the computational cost involved. Despite Murty's algorithm [137] incorporated into the MHT framework [54, 58], the pruning of unnecessary branches in the track tree is still required. Another problem with the tracker is the use of a separate tracker for every single point feature considered. If several features (> 250) are to be tracked, then the algorithm performs poorly due to the complex management of a large number of hypothesis trees.

Chapter 5 – In this chapter we provided closed form solutions for predicting the performance of a point feature tracker under clutter using different motion models. We showed that the theoretical and empirical results presented closely matched the Mote-Carlo simulations, provided the assumptions listed were maintained. Due to the complexity of the problem, a number of assumptions were made in order to formulate the closed form representations. The assumptions taken do compromise the final results obtained to some extent as observed from the results. To minimize the number of assumptions, a deeper, rigorous theoretical study along the lines of [125, 126] would be required. An area that was not fully covered by theoretical formulations is the analysis for a tracker (that employs a constant velocity or a constant acceleration model) when recovering from a false match (we have provided a technique which is a combination of theoretical and empirical methods, but a complete theoretical formulation will be very useful). In this chapter we have considered only 3 simple linear motion models for performance prediction. It would be interesting to formulate solutions for other type of motion models (eg: a constant turn model, non-linear motion models, oscillatory motion models etc.).

Chapter 6 - The object tracking method described in this chapter is adequate only for simple objects that are not occluded. Despite the attractiveness of the algorithm, it suffers from lack of efficiency for real time purposes. The main reason for inefficiency is the application of MHT in two stages: the first for contour segmentation, and the second for temporal point tracking. A better method for object tracking would be to formulate the tracking problem along the lines of Torr et al. [192, 193] (use of geometric information criteria for motion segmentation), Reid et al. [153] (uses optic flow methods to group points belong to the same object), or Smith et. al. [172] (use of motion estimates to group point features that are of the same object). It is also useful to track objects that are far more complex than the ones that were considered in this chapter. In complex objects one would have to take into account the corner points that might appear within the object of interest, in which case the algorithm presented in this chapter might fail, because it considers only the contour (the boundary) of an object.

Chapter 7 - The CONT-IMM tracker presented in this chapter was demonstrated to track well with automatic motion model switching when applied on a variety of sequences. One area where the tracker is vulnerable is when tracking under heavy clutter. Since the tracking technique uses background subtraction to obtain contour measurements, heavy clutter can cause spurious contour measurements, thus resulting in poor quality contour tracking. A way around this problem is to use an efficient feature search method such as the methods reported in [163, 24], where suitable statistical background and foreground models are developed to reduce the effect of clutter.

For the experiments reported in this chapter we considered only 3 motion models to test the model switching ability of the tracker. For a more rigorous evaluation of the tracker, it is worth testing the CONT-IMM algorithm with other types of motion models (eg: non-linear motion models, oscillatory motion models, learned motion models that pertain to a type of object etc.). It is also worth trying the algorithm to track other objects such as animals, vehicles etc.

Chapter 8 - An important part missing in chapter 8 is a sound theoretical basis for comparing the performance of contour trackers. The methods that we have employed are simple empirical techniques to assess the output of the contour trackers. The methods provided do not give any prediction measures (as in chapter 5). Due to time constraints further theoretical performance analysis work could not be carried out. An important area of research worth considering to improve the work reported in this chapter is to provide closed form expressions for theoretically assessing the performance of contour trackers (to the author's knowledge there are no comprehensive techniques reported in the literature for comparing the performance of different types of contour trackers). One possible thought is to enhance the work presented in chapter 5 (performance prediction techniques for point feature trackers) for contour tracking algorithms. Another area that needs addressing is a sound comparison technique without using B-spline related measures. For contour trackers that do not employ splines, the B-spline performance measures presented cannot be employed. The SNR method used is also inadequate, since it depends on the foreground area of the moving object. Therefore an image processing technique that is independent of moving object size will be of valuable use.

9.3 Future Research Directions

As noted before a problem with MHT based tracking techniques is the high computational cost. One method to avoid costly computational time is to reduce the complexity of the tracking algorithm. A possible future direction of work is to consider extending the KLT tracker. As discussed in chapter 3, the KLT tracker does not employ a motion model within its tracking framework [190], but uses the corner detection process to complement the tracking task. A possible extension for this tracker is to consider embedding a motion model/s into the KLT tracker (similar to [164]). Such a tracker would be expected to perform better than the MHT-IMM algorithm in terms of efficiency.

An efficient method of point feature tracking can also be developed using similar techniques to the Condensation algorithm (with an appropriate feature detection process embedded within the tracking system). Such a tracker would be expected to run faster than a MHT based algorithm. Possible model switching can also be achieved by following similar techniques to Blake et al. [106, 156].

An extension of contour tracking method is to track not only the contour of the object but also the grey-levels contained within the object of interest. If achievable, such a technique will provide valuable information from the image sequence. Cootes et al [53, 71] have used Active Appearance Models for recovering faces using such methods, but to the author's knowledge very little work has been done in recovering the full deformable object using temporal tracking. Some success has been reported in [165], where small blobs are tracked, but further work has to be undertaken to track a complex deformable object such as a walking pedestrian.

Other possible areas of research include the following: Track 3D objects as opposed to 2D contours with motion model switching (using ideas from [91, 177, 178]). Use tracking techniques to reconstruct a scene from sequence of images (panoramic scene understanding), interpretation of scenes using tracking methods (similar to [100]), and the possibility of employing tracking concepts for the restoration of damaged sequences that contain moving objects.

Appendix A

Corner Detector Performance with Small Motion

This section gives extra information to that which was reported in chapter 3 (page 47).

A.1 Image Sequences with Very Small Motion

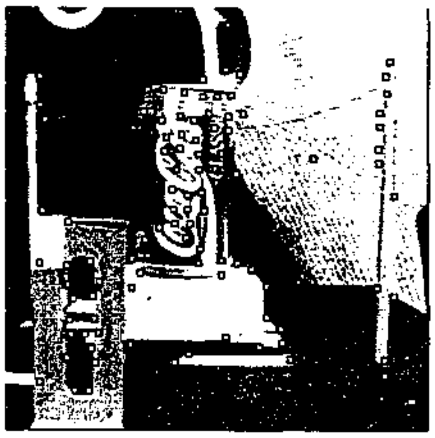
Two sequences were considered with very small motion component in them. First, a 30 frame Coke sequence was used, where there is a small camera motion towards the scene. Secondly, a 20 frame Rubic sequence is considered with a small rotational motion of the Rubic. In both sequences the maximum inter-frame displacement due to motion was around 1 pixel (see Appendix G).

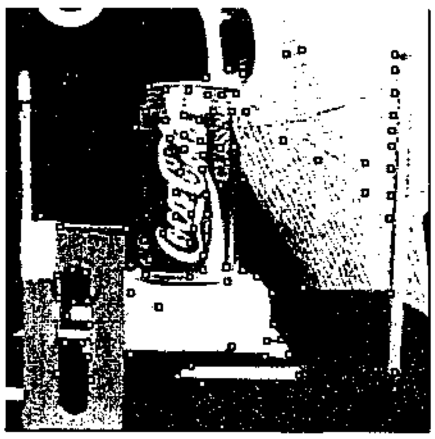
A.1.1 Test Results for the Coke Sequence

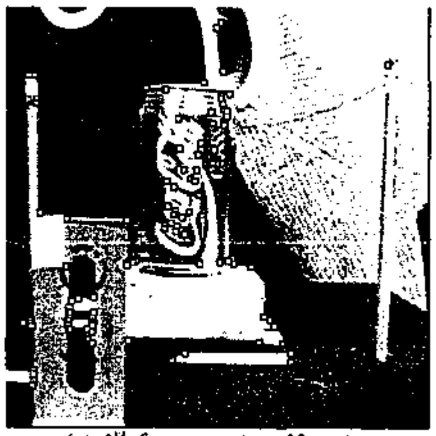
The best 100 corners extracted by each of the corner detectors are qualitatively displayed in Fig. (A.1). The quantitative results show that the KLT and Harris provide equally good number of stable corners (55% each), while SUSAN resulted with around 35% and Kitchen-Rosenfeld gave only 20% stable corners using the GVM matcher (0.004 threshold). For the same experiment using PMCM matcher (0.8 threshold), Harris (30%) gave a better result than KLT (20%), while SUSAN and Kitchen-Rosenfeld provided only around 10% stable corners. The number of first frame corner matches also suggests that Harris and KLT provide more matches than SUSAN and Kitchen-Rosenfeld detectors for both matchers (Fig. (A.2)). The mean corner displacement result shows that the corners extracted by the Harris and the KLT (with around 1 pixel displacement) detectors are more localized than the corners detected by the SUSAN (1.7 pixels) and the Kitchen-Rosenfeld (2 pixels) detectors using the GVM matcher. With the PMCM matcher, Harris gives around 0.7 pixel displacement, KLT provides 1.25 pixels, while SUSAN and Kitchen-Rosenfeld provides more than 2 pixels displacement (Fig. A.2).

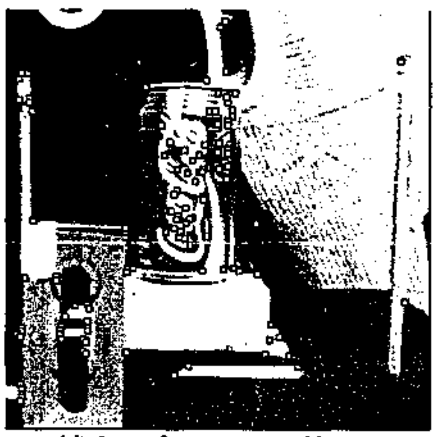
A.1.2 Test Results for the Rubic Sequence

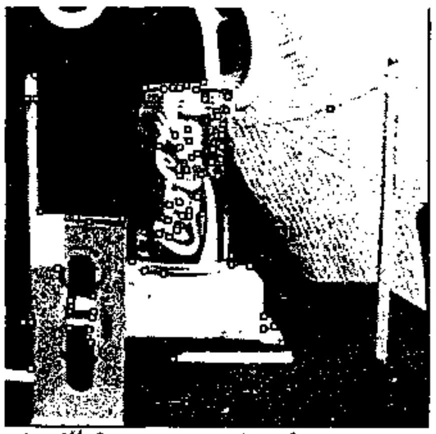
The best 100 corners extracted by each of the corner detectors are qualitatively displayed in Fig. (A.3). The result reported in Fig. (A.4) shows that KLT, Harris and SUSAN give approximately 50% stable corners using GVM (0.004 threshold), and around 10%-15% using PMCM (0.8 threshold). The Kitchen-Rosenfeld gives the least stable corners using both matchers, about 25% and 8% respectively. The number of first frame corner matches is between 60-75 for all four detectors using GVM matcher, and around 40-50 matches using PMCM matcher. The corner displacement result clearly shows that KLT with around 1.5 pixel displacement is better localized than Harris (2 pixel), SUSAN (2.5 pixel) and Kitchen-Rosenfeld (2.6 pixels) detectors using the GVM matcher. Similar observations can also be noted using the PMCM matcher (Fig. (A.4 b,d,f)).

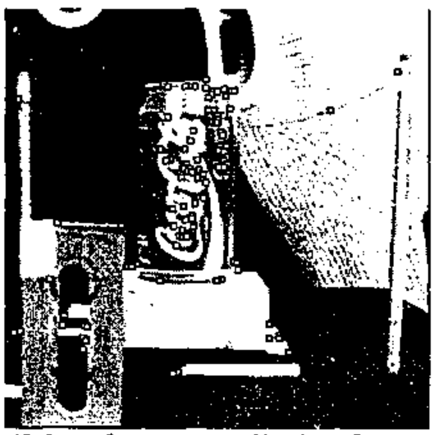

(a) 1st frame using KLT

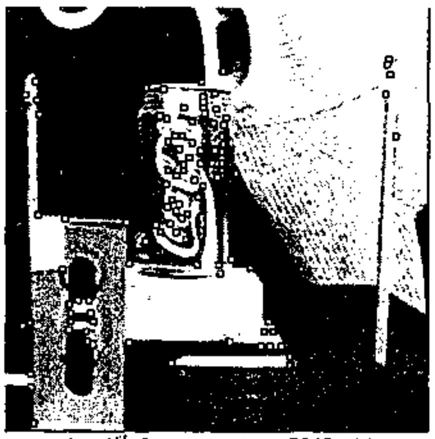

(b) last frame using KLT


(c) 1st frame using Harris


(d) Last frame using Harris


(e) 1st frame using Kitchen-Rosen


(f) Last frame using Kitchen-Rosen


(g) 1st frame using SUSAN

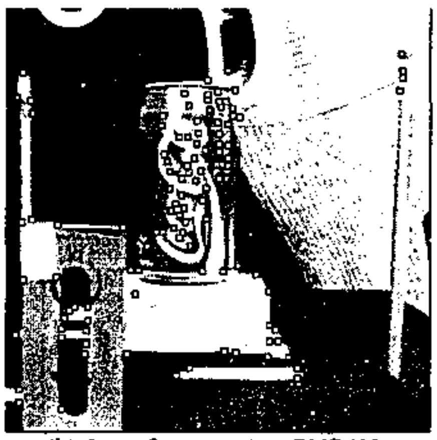
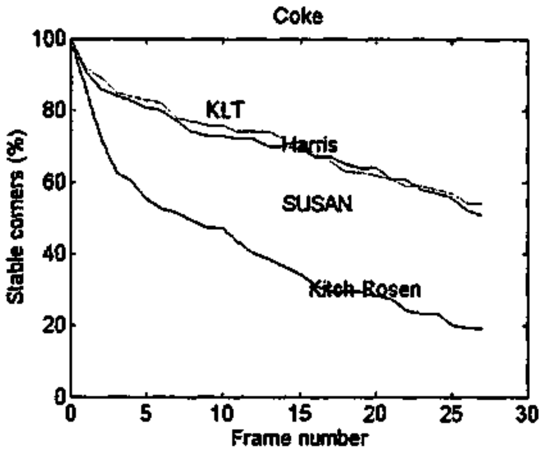
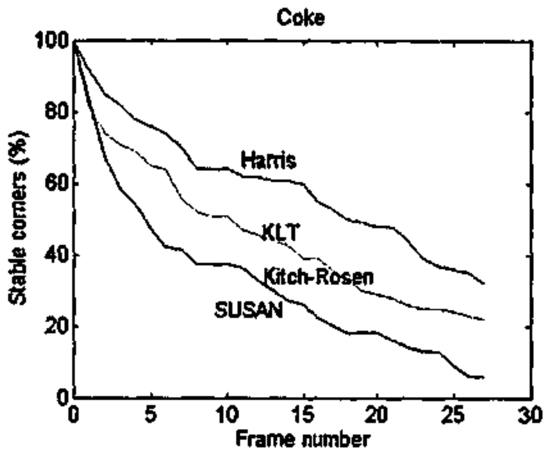

(h) Last frame using SUSAN

Figure A.1: First and last frame (row wise) of the Coke sequence showing the best 100 corners as seen by each of the 4 corner detectors.

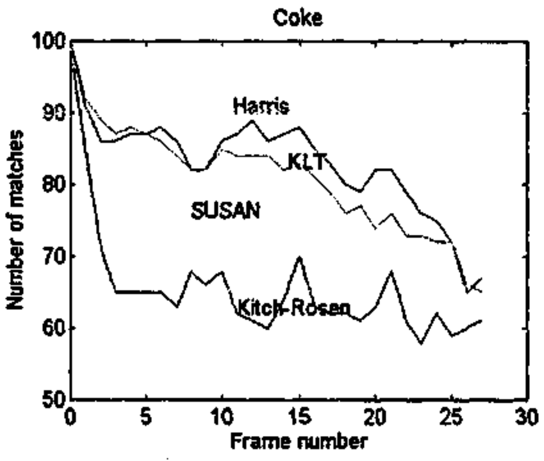
Coke Sequence Performance Results



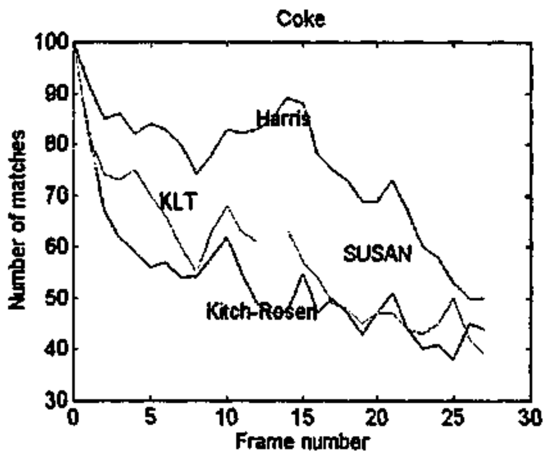
(a) Using GVM (Thresh=0.004)



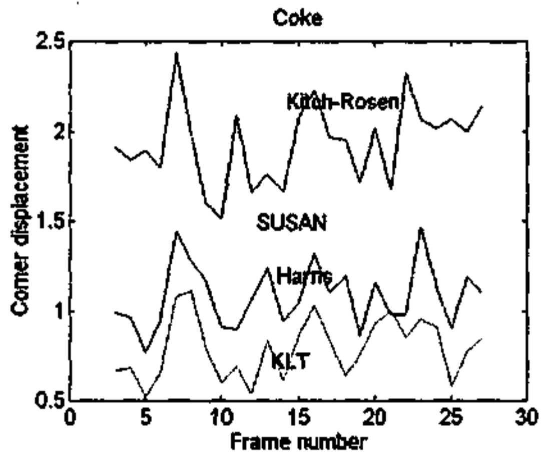
(b) Using PMCM (Thresh = 0.8)



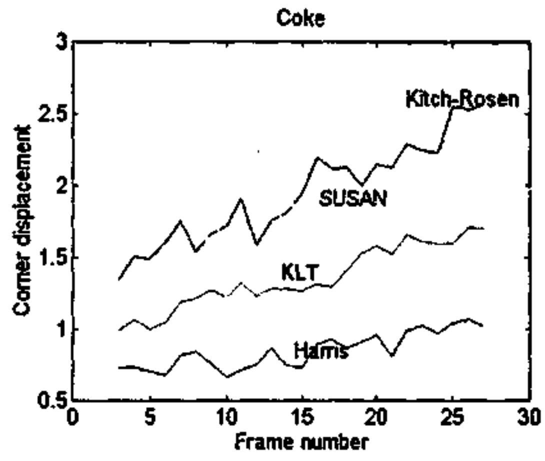
(c) Using GVM (Thresh=0.004)



(d) Using PMCM (Thresh = 0.8)

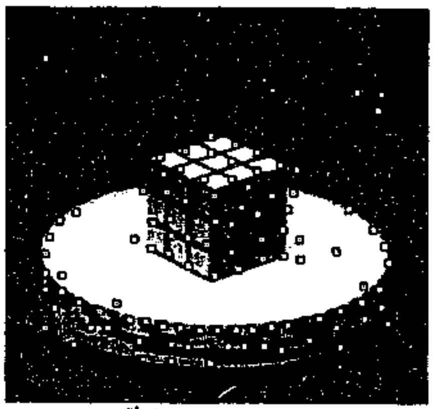


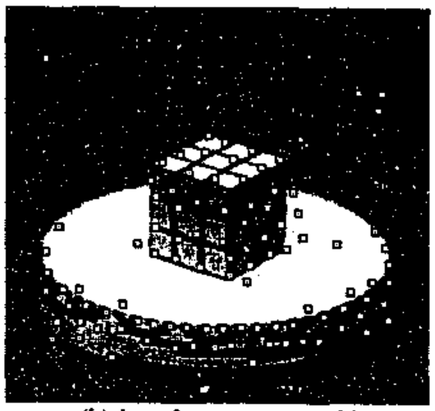
(e) Using GVM (Thresh=0.004)

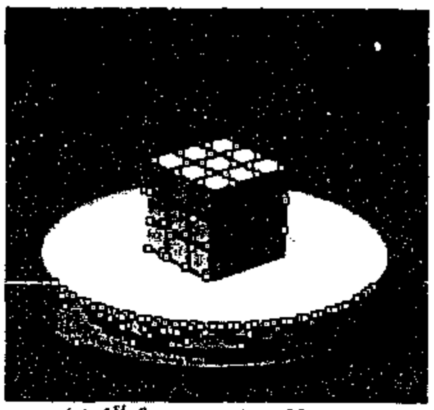


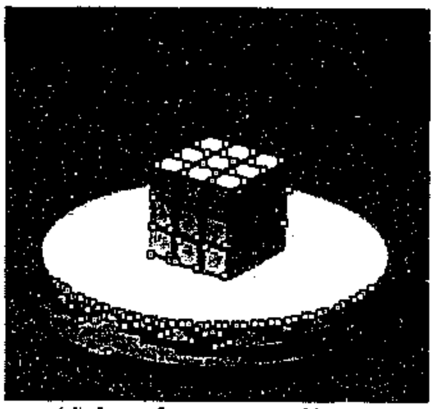
(f) Using PMCM (Thresh = 0.8)

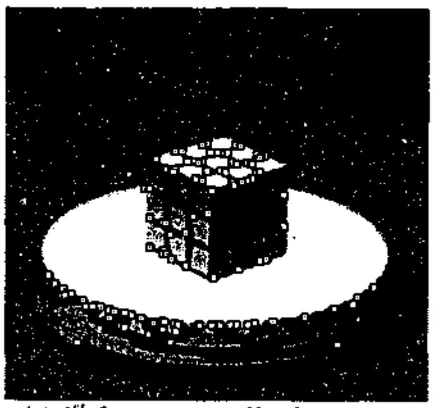
*Figure A.2: Performance of the 4 corner detectors when applied to the **Coke sequence**. (a) The percenrtage stable corners using GVM matcher. (b) The percenrtage stable corners using PMCM matcher. (c) The number of first frame corner matches using GVM macther. (d) The number of first frame corner matches using PMCM macther. (e) The corner displacment (in pixels) using GVM matcher. (f) The corner displacment (in pixels) using PMCM matcher.*

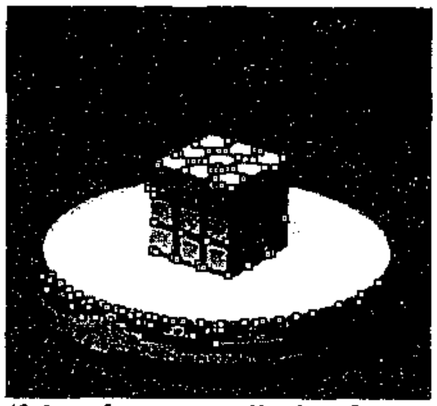

(a) 1st frame using KLT

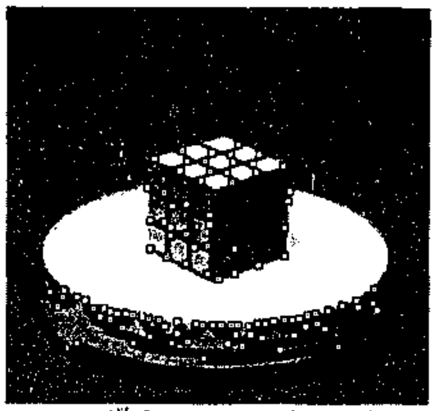

(b) last frame using KLT


(c) 1st frame using Harris


(d) Last frame using Harris


(e) 1st frame using Kitchen-Rosen


(f) Last frame using Kitchen-Rosen


(g) 1st frame using SUSAN

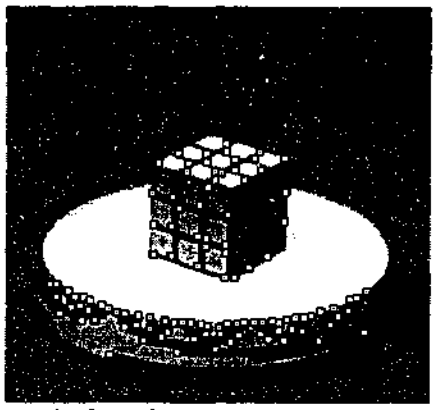
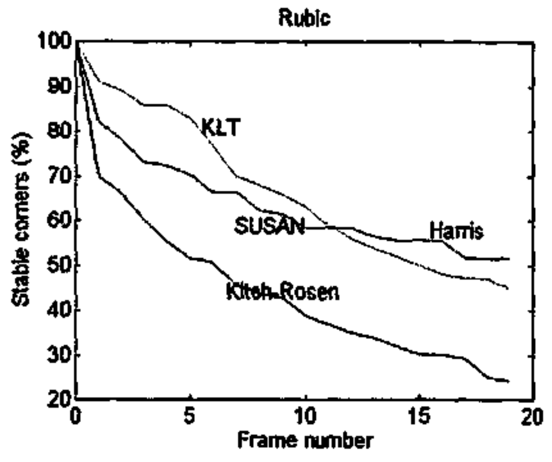
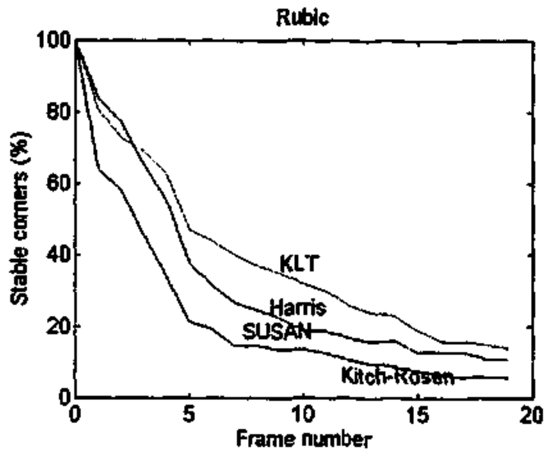

(h) Last frame using SUSAN

Figure A.3: First and last frame (row wise) of the Rubic sequence showing the best 100 corners as seen by each of the 4 corner detectors.

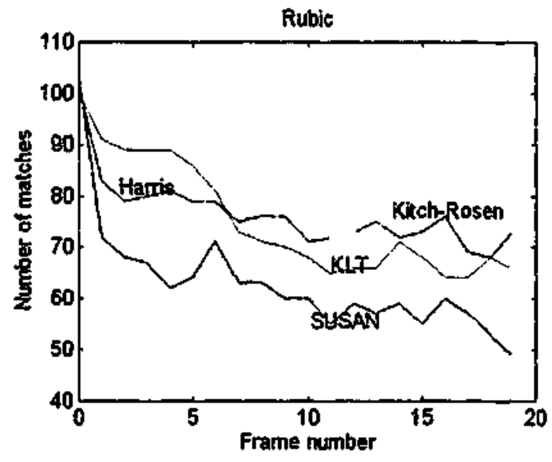
Rubic Sequence Performance Results



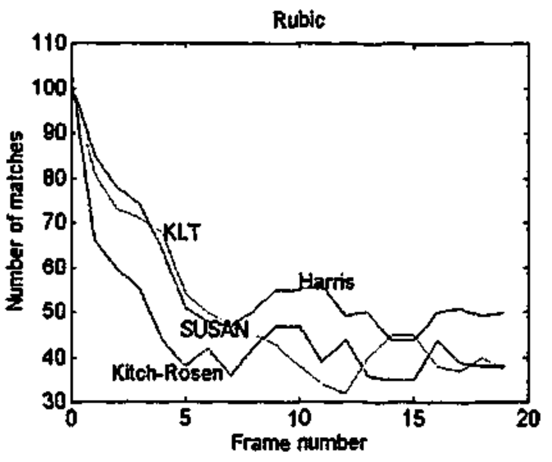
(a) Using GVM (Thresh=0.004)



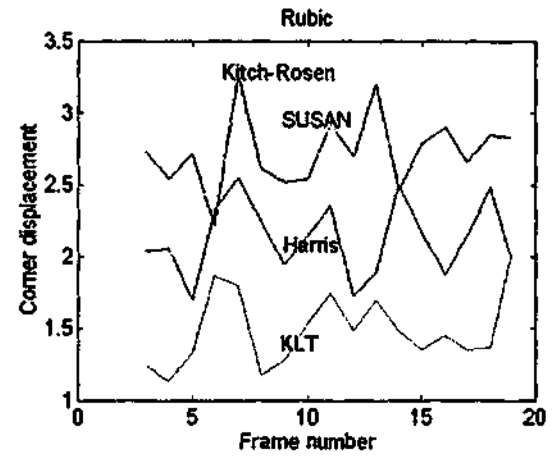
(b) Using PMCM (Thresh = 0.8)



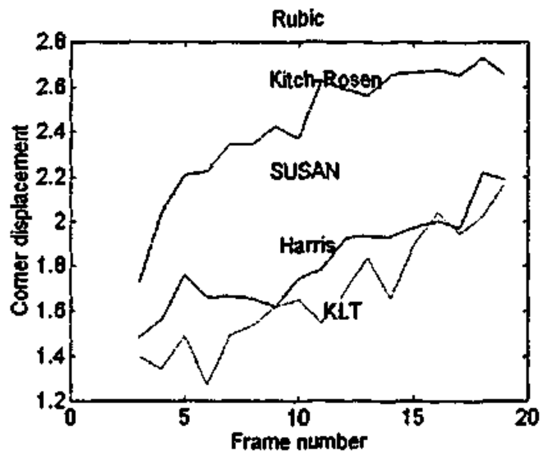
(c) Using GVM (Thresh=0.004)



(d) Using PMCM (Thresh = 0.8)



(e) Using GVM (Thresh=0.004)



(f) Using PMCM (Thresh = 0.8)

Figure A.4: Performance of the 4 corner detectors when applied to the ***Rubic sequence****. (a) The percenrtage stable corners using GVM matcher. (b) The percenrtage stable corners using PMCM matcher. (c) The number of first frame corner matches using GVM macther. (d) The number of first frame corner matches using PMCM macther. (e) The corner displacment (in pixels) using GVM matcher. (f) The corner displacment (in pixels) using PMCM matcher.*

Appendix B

Multiple Hypothesis Algorithm

B.1 MHT Overview

This section briefly describes the MHT Algorithm (the details of which are given in [152], [54], [58]). In section B.2 we look at the hypothesis generation. In section B.3 the mathematical framework for this algorithm is presented and finally in section B.4 the method of generation of the K-best hypothesis is given. The concepts presented in the whole of this section are used to predict and match features for our feature tracking analysis presented in Chapter 4.

The Multiple Hypothesis Tracking (MHT) algorithm was originally developed by Reid [152] in the context of multi-target tracking. Fig. 4.1 in chapter 4 outlined the basic operation of the MHT algorithm. An iteration begins with the set of current hypotheses from iteration (k-1). Each hypothesis represents a different set of assignments of measurements to features, i.e., it is a collection of *disjoint tracks.* A track is defined to be a sequence of measurements that are assumed to originate from the same geometric feature. A dummy track in each global hypothesis denotes spurious measurements.

Different sets of assignments expect to see different sets of measurements. Thus, each hypothesis predicts the location (in the image plane) of a set of expected geometric features (specifically corners) and these are compared with actual measurements detected in the next camera frame on the basis of their *Mahalanobis* distance [58, 6]. These comparisons are represented in the form of an *ambiguity matrix* (was defined in Chapter 4.2), which concisely models the ambiguities present in assigning measurements to features.

Each measurement may either 1) belong to a previously known geometric feature, 2) be the start of a new geometric feature, e.g., a previously unseen corner that has entered the field of view of the camera, 3) be a spurious measurement (also called a false alarm). In addition, for geometric features that are not assigned measurements, there is the possibility of 4) deletion of the geometric feature. This situation may arise when say a corner feature leaves the field of view of the camera. Alternatively, 5) there is the possibility of continuation of a geometric feature, the missed measurement perhaps being due to either noise or a temporary occlusion caused by the motions of the camera and objects in the scene.

After matching, each global hypothesis (from iteration (k-1)), has an associated ambiguity matrix from which it is necessary to generate a set of legal assignments (see Chapter 4.2). Each subsequent child hypothesis represents one possible interpretation of the new set of measurements and, together with its parent hypothesis, represents one possible interpretation of all past measurements.

Finally in order to contain the growth of the tree, it is necessary to prune unlikely branches (see [58] for further details on pruning mechanism). In order to do this intelligently, one needs to evaluate the likelihood of each hypothesis. Section B.2 and B.3 provides the mathematical framework for estimating the probability of each leaf in the tree.

B.2 Hypothesis Generation

The l-th practical global hypothesis at time k is denoted by Θ_l^k, and $Z(k)$ the set of measurements at time k. Let $\Theta_{m(l)}^{k-1}$ denote the parent hypothesis from which Θ_l^k is derived, and $\theta_m(k)$ denote the specific set of assumed assignments (events) that map $\left\{\Theta_{m(l)}^{k-1}, Z(k)\right\}$ to Θ_l^k. That is, $\theta_m(k)$ is a set of assignments of the origins of all measurements received at time k with all the geometric features postulated by the parent hypothesis, $\Theta_{m(l)}^{k-1}$ at time k. The event $\theta_l(k)$ based on the current measurements is defined to consist of τ measurements from known geometric features, ν measurements from new geometric features, ϕ spurious measurements (false alarms), and χ deleted (or obsolete) geometric features from the parent hypothesis.

A set of current assignments or events $\theta_l(k)$ can be generated by first creating an ambiguity matrix in which known geometric features are represented by the columns of the matrix and the current measurements by the rows. A non zero element at matrix position $c_{i,j}$ denotes that measurement $z_i(k)$ is contained in the validation region of geometric feature t_j. In addition to the total number, T, of known geometric features postulated by a hypothesis, the hypothesis matrix has appended to it a column 0 (T_F) denoting false alarms and a column T+1 (T_N) denoting new geometric features. The situation depicted in Fig. 4.2 (a) in chapter 4 is represented by the hypothesis matrix shown in Fig 4.2 (b).

It is desired to constrain the legal set of assignments to be disjoint so that 1) a measurement originates from only one source feature and that 2) a geometric feature has at most one associated measurement per iteration. This is equivalent to restricting an ambiguity matrix to have only a single non-zero value in any row or column, except for the first and last columns since any number of measurements might be false alarms or new geometric features. If the first and last columns of the ambiguity matrix are replicated m_k times for each of the m_k measurements, then there is only a single nonzero in any row or column and the ambiguity matrix can be thought of as a cost matrix in a linear assignment problem (or weighted bipartite graph matching [137]). Enumeration of all legal sets of assignments, $\theta_l(k)$, is straight forward [58], but impractical for anything other than a trivial example. Section B.4 describes briefly how the ambiguity matrix can be modified to represent a classical assignment matrix from which the k-best assignments (hypotheses) can be generated using an algorithm due to Murty [137].

B.3 Probability Calculations

The new hypothesis at time k, Θ_l^k is made up of the current set of assignments (also called an event), $\theta_l(k)$, and a previous hypothesis, $\Theta_{m(l)}^{k-1}$ based on measurements up to and including time k - 1, i.e.,

$$\Theta_l^k = \left\{\Theta_{m(l)}^{k-1}, \theta_l(k)\right\} \tag{B.1}$$

The probability of an hypothesis, $P\left\{\Theta_l^k \middle| Z^k\right\}$ can be calculated using Bayes' rule, so that

$$\begin{aligned} P\left\{\Theta_l^k \middle| Z^k\right\} &= P\left\{\theta_l(k), \Theta_{m(l)}^{k-1} \middle| Z(k), Z^{k-1}\right\} \\ &= \frac{1}{c} p\left[Z(k) \middle| \theta_l(k), \Theta_{m(l)}^{k-1}, Z^{k-1}\right] P\left\{\theta_l(k) \middle| \Theta_{m(l)}^{k-1}, Z^{k-1}\right\} P\left\{\Theta_{m(l)}^{k-1} \middle| Z^{k-1}\right\} \end{aligned} \tag{B.2}$$

where c is a normalisation constant. The last term of this equation, $P\left\{\Theta_{m(l)}^{k-1} \middle| Z^{k-1}\right\}$, represents the probability of the parent global hypothesis and is therefore available from the previous iteration. The remaining two terms may be evaluated as follows.

The second factor of (B.2) is obtained by combining results from [6] and [152] to yield [58],

$$P\left\{\theta_l(k) \middle| \Theta_{m(l)}^{k-1}, Z^{k-1}\right\} = \frac{\phi! \nu!}{m_k!} \mu_F(\phi) \mu_N(\nu) \prod_t \left(P_D^t\right)^{\delta_t} \left(1 - P_D^t\right)^{1-\delta_t} \left(P_\chi^t\right)^{\chi_t} \left(1 - P_\chi^t\right)^{1-\chi_t} \tag{B.3}$$

where $\mu_F(\phi)$ and $\mu_N(\nu)$ are the prior probability mass function (PMFs) of the number of spurious measurements and new geometric features (ν is the number of measurements from new geometric features, ϕ is number of spurious measurements (false alarms) and m_k is the total number of measurements at time k), P_D^t and P_χ^t are the probabilities of detection and termination (deletion) of track t and δ_t and χ_t are indicator variables defined by

$$\delta_t \equiv \begin{cases} 1 & \textit{if geometric feature } t \textit{ (in } \Theta_{m(l)}^{k-1}\textit{) is detected at time } k \\ 0 & \textit{otherwise} \end{cases}$$

$$\chi_t \equiv \begin{cases} 1 & \textit{if geometric feature } t \textit{ (in } \Theta^{k-1}_{m(l)}\textit{) is deleted at time } k \\ 0 & \textit{otherwise} \end{cases}$$

To determine the first term on the right hand side of (B.2) it is assumed that a measurement $z_i(k)$ has a Gaussian probability density function (pdf)

$$\begin{aligned} N_{t_i} &= N[\mathbf{z}_i(k)] \equiv N\left[\mathbf{z}_i(k); \hat{\mathbf{z}}_i(k|k-1), \mathbf{S}^{t_i}(k)\right] \\ &= \left|2\pi \mathbf{S}^{t_i}(k)\right|^{-\frac{1}{2}} e^{-\frac{1}{2}\left\{(\mathbf{z}(k)-\hat{\mathbf{z}}(k|k-1))^T (S^{t_i(k)})^{-1}(\mathbf{z}(k)-\hat{\mathbf{z}}(k|k-1))\right\}} \end{aligned} \tag{B.4}$$

if it is associated with geometric feature t_i, where $\hat{\mathbf{z}}_i(k|k-1)$ denotes the predicted measurement for geometric feature t_i and $\mathbf{S}^{t_i}(k)$ is the associated innovation covariance. If the measurement is spurious (a false alarm), then it's pdf is assumed uniform in the observation volume, V. The probability of a new geometric feature is also taken to be uniform with pdf of value V^{-1}. Under these assumptions, one has,

$$\begin{aligned} p\left[Z(k)|\theta_l(k), \Theta^{k-1}_{m(l)}, z^{k-1}\right] &= \prod_{i=1}^{m_k}\left[N_{t_i}[\mathbf{z}_i(k)]\right]^{\tau_i} V^{-(1-\tau_i)} \\ &= V^{-\phi-\nu}\prod_{i=1}^{m_k}\left[N_{t_i}[\mathbf{z}_i(k)]\right]^{\tau_i} \end{aligned} \tag{B.5}$$

where τ_i is an indicator variable defined as

$$\tau_i \equiv \begin{cases} 1 & z_i(k) \textit{ came from a known geometric feature} \\ 0 & \textit{otherwise} \end{cases}$$

and ν and ϕ are the total number of new geometric features and false alarms, respectively.

Substituting (B.5) and (B.3) into (B.2) yields the final expression for the conditional probability of an association hypothesis [58]

$$\begin{aligned} P\left\{\Theta^k_l|Z^k\right\} = \frac{1}{c}\frac{\phi!\nu!}{m_k}\mu_F(\phi)\mu_N(\nu)V^{-\phi-\nu}\prod_{i=1}^{m_k}\left[N_{t_i}[\mathbf{z}_i(k)]\right]^{\tau_i} \\ \left\{\prod_t (P^t_D)^{\delta_t}(1-P^t_D)^{1-\delta_t}(P^t_\chi)^{\chi_t}(1-P^t_\chi)^{1-\chi_t}\right\}P\left\{\Theta^{k-1}_{l(m)}|Z^{k-1}\right\} \end{aligned} \tag{B.6}$$

If the number of false alarms and new features are assumed to be Poisson distributed with densities λ_F and λ_N respectively, then (B.6) reduces to

$$P\left\{\Theta_m^k | Z^k\right\} = \frac{1}{c}\lambda_N^{\nu}\lambda_F^{\phi}\prod_{i=1}^{m_k}\left[N_{t_i}[\mathbf{z}_i(k)]\right]^{\tau_i}\left\{\prod_t (P_D^t)^{\delta_t}(1-P_D^t)^{1-\delta_t}(P_\chi^t)^{\chi_t}(1-P_\chi^t)^{1-\chi_t}\right\} P\left\{\Theta_{l(m)}^{k-1} | Z^{k-1}\right\} \tag{B.7}$$

The probability of each hypothesis can be used to guide a pruning strategy [58].

B.4 Generating the *k*-Best Hypothesis (Murty's Algorithm)

Because of the exponential complexity of the multiple hypothesis approach only an approximation to the MHT algorithm can be implemented. In particular, it is simply not feasible to search the entire space of hypotheses in order to determine the most likely set of assignments. Several implementation strategies were employed in order to contain the growth of the hypothesis tree and reduce the number of hypotheses that must be considered.

In order to generate the *K*-best hypotheses (from a problem size > *K*), Cox et. al. [54] used an algorithm due to Murty [137] to optimally determine the *K*-best assignments in polynomial time. The number of linear assignment problems that must then be solved is linear in *k*. In fact, "the computations required at each stage are the solving of at most (*n*-1) assignment problems, each of sizes *2,3,...,n*" [137]. The algorithm avoids solving duplicate assignment problems [54] (see later for definition), thereby eliminating the need to compare and delete duplicate hypotheses. Finally, in the average case, the dimension of the assignment problems that must be examined decreases with increasing *k*.

Consider first the problem of finding the single most probable hypothesis. This can be cast as a weighted bipartite matching problem by constructing a bipartite graph in which each node on one side represents one of the measurements, each node on the other represents one of the targets, and each arc, $< z_i, t_j, l >$, gives the log likelihood , l, that measurement z_i should be assigned to target t_j. The log of the likelihood of a given assignment can be found by summing the log likelihoods of all the arcs that it specifies. These log likelihoods can be calculated from equation B.7.

Finding the best hypothesis, then, is a matter of finding the assignment that maximizes this sum. This is an instance of the classical assignment problem from combinatorial optimization, and can be approximately solved very efficiently in polynomial time [54, 58]. Murty's algorithm is also guaranteed to find the *K*-best assignments in polynomial time. A brief description of Murty's algorithm follows:

1) The set of valid solutions for any one of the problems in the list doesn't intersect with the set of solutions for any other problem in the list. That is, there are no duplicate problems.

2) The union of the sets of valid solutions for all the problems in the list is exactly the set of solutions for problem P, minus solution S [137].

Murty gives a method for computing this partitioning in $O(N^3)$ time, where N is the dimension of the problem. For the k-best algorithm, a list of problem/solution pairs is kept. Each pair consists of an assignment problem and its best solution. The list is initialised with the initial problem to be solved. In each iteration, the best solution is found, then removed from the list, and replaced with its partitioning. So, in the first iteration, the single best solution, S_0, is found to the problem, and the list is altered so the set of possible solutions no-longer contains S_0. The next iteration gives the next best solution, S_1, and changes the list so that possible subsequent solutions no-longer include S_1 or S_0, and so on. The following steps outline the algorithm. The partitioning is performed by the loop in step 4.4. See [137], [58] for more details.

```
1) Find the best solution, $S_0$, to $P_0$ (this can be done using a standard algorithm like the Hungarian method
2) Initialize the list of problem / solution pairs with $< S_0, P_0 >$
3) Clear the list of solutions to be returned
4) For $i = 1$ to $k$, or until the list of problem / solution pairs is empty
        4.1 Search through the list of problems / solution pairs, and find the pair, $< P, S >$ that has the
        best solution value
        4.2 Remove $< P, S >$ from the list of problem / solution pairs
        4.3 Add S to the list of solutions to be returned
        4.4 For each triple, $< t, z, l >$, found in S
                4.4.1 Let $P' = P$
                4.4.2 Remove the triple $< t, z, l >$ from $P'$
                4.4.3 Look for the best solution, $S'$, to $P'$
                4.4.4 If $S'$ exists
                        4.4.4.1 Add $< P', S' >$ to the set of problem / solution pairs
                4.4.5 From P, remove all triples that include $t$, and all triples that include $z$, except
                $< t, z, l >$ itself. (This reduces the dimension of the problem by one)
```

Figure B.1: Murty's algorithm for finding the k-best solutions to an assignment problem, P_0.

NOTE: It is also worth noting that, to supplement the Mahalanobis distance in the MHT technique, Cox et al. [58] also introduced a cross correlation test [168]. This was used in order to reduce the total number of possible initial matches (and also increase the number of disjoint clusters).

B.5 The Kalman Filter Recursion

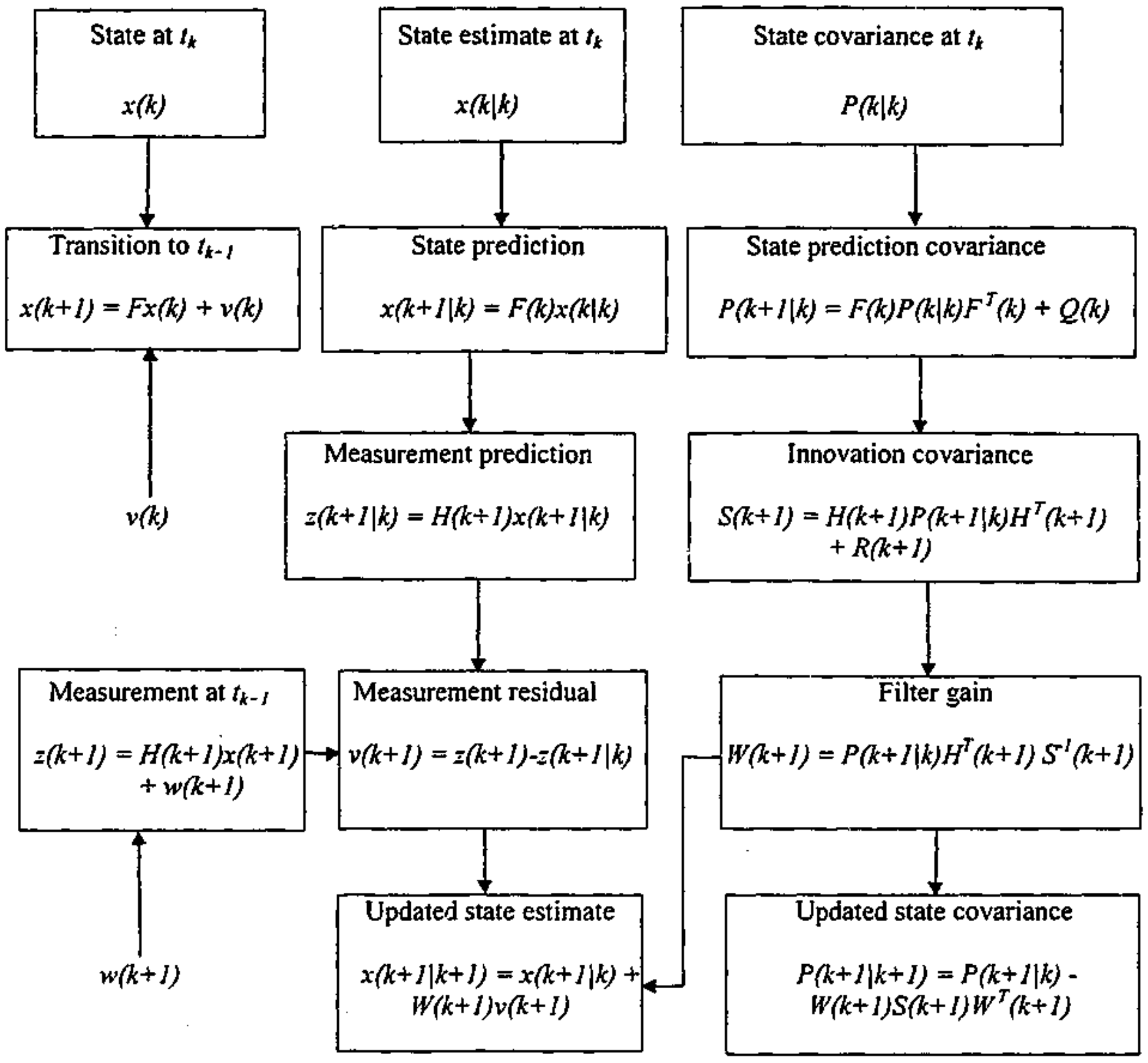

State at t_k
$x(k)$
State estimate at t_k
$x(k|k)$
State covariance at t_k
$P(k|k)$
Transition to t_{k-1}
$x(k+1) = Fx(k) + v(k)$
State prediction
$x(k+1|k) = F(k)x(k|k)$
State prediction covariance
$P(k+1|k) = F(k)P(k|k)F^T(k) + Q(k)$
$v(k)$
Measurement prediction
$z(k+1|k) = H(k+1)x(k+1|k)$
Innovation covariance
$S(k+1) = H(k+1)P(k+1|k)H^T(k+1) + R(k+1)$
Measurement at t_{k-1}
$z(k+1) = H(k+1)x(k+1) + w(k+1)$
Measurement residual
$v(k+1) = z(k+1)-z(k+1|k)$
Filter gain
$W(k+1) = P(k+1|k)H^T(k+1)\, S^{-1}(k+1)$
$w(k+1)$
Updated state estimate
$x(k+1|k+1) = x(k+1|k) + W(k+1)v(k+1)$
Updated state covariance
$P(k+1|k+1) = P(k+1|k) - W(k+1)S(k+1)W^T(k+1)$

Figure B.2: One Cycle of the Kalman Filter [5]

Appendix C

Proof for the Probability of Correct Association

This section contains proofs for the material reported in Chapter 5.

C.1 Proof for the Probability of Correct Association (PCA) for Constant Velocity Tracker (CVT)

Starting from equation (5.9) in chapter 5

$$\begin{aligned} P_{CVT}\{k+1 \mid \eta_k, \mathbf{a}\} &= P_0^{\pi\|c_{k+1}\|^2} \\ &= P_0^{\pi\|\eta_k+\mathbf{a}\|^2} \end{aligned}$$

Using the total probability theorem [5, 6], for the constant velocity model (CVT) we get,

$$P_{CVT}\{k+1\} = \int_{-\infty}^{+\infty} p\{k+1 \mid \eta_k, a\}.P(\eta_k).d\eta_k$$

Therefore the expression we require can be given by the following equation:

$$P_{CVT}\{k+1\} = \int_{-\infty}^{+\infty} P_0^{\pi\|\eta_k+\mathbf{a}\|^2}.P(\eta_k).d\eta_k \tag{C.1}$$

where $P(\eta_k) = \dfrac{1}{\sqrt{2\pi}\sigma_k}\exp\left\{-\dfrac{1}{2}\left(\dfrac{\eta_k}{\sigma_k}\right)^2\right\}$

Expanding equation C.1 using the total probability theorem gives

$$\begin{aligned} P_{CVT}\{k+1\} &= \int_{-\infty}^{+\infty} P_0^{\pi\|\eta_k+a\|^2}.\frac{1}{\sqrt{2\pi}\sigma_k}\exp\left\{-\frac{1}{2}\left(\frac{\eta_k}{\sigma_k}\right)^2\right\}d\eta_k \\ &= \frac{1}{\sqrt{2\pi}\sigma_k}\int_{-\infty}^{+\infty}\exp\left\{\ln P_0\pi\|\eta_k+a\|^2\right\}\exp\left\{-\frac{1}{2}\left(\frac{\eta_k}{\sigma_k}\right)^2\right\}d\eta_k \qquad \text{(C.2)} \\ &= \frac{1}{\sqrt{2\pi}\sigma_k}\int_{-\infty}^{+\infty}\exp\left\{-\alpha\eta_k^2+\beta\eta_k+\gamma\right\}d\eta_k \end{aligned}$$

where the following definitions are used for α, β, γ.

$$\alpha = -\left(\ln P_0\pi - \frac{1}{2\sigma_k^2}\right), \beta = 2a\ln P_0\pi, \gamma = a^2 \ln P_0\pi$$

η_k, a (That is noise and acceleration as given before) are 2 dimensional vectors (They have a x and y component). Therefore equation C.2 can be expressed in terms of the x and y components.

$$P_{CVT}\{k+1\} = \frac{1}{2\pi\sigma_x\sigma_y}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\exp\{-\alpha_x\eta_x^2 + \beta_x\eta_x + \gamma_x\}\exp\{-\alpha_y\eta_y^2 + \beta_y\eta_y + \gamma_y\}d\eta_x d\eta_y$$

where η_x, η_y are the x and y elements of the random variable η, and σ_x^2, σ_y^2 are the x and y variances of the Gaussian distribution.

Lets evaluate the integrand associated with x component of the above expression (say I_x),

$$I_x = \frac{e^{\gamma_x}}{\sqrt{2\pi}\sigma_x}\int_{-\infty}^{+\infty}\exp\{-\alpha_x\eta_x^2\}.\exp\{\beta_x\eta_x\}d\eta_x$$

For small values a_x (x component of the error acceleration) the second exp{.} in the above equation can be approximated up to 2nd order terms. Therefore the above equation becomes,

$$I_x = \frac{e^{\gamma_x}}{\sqrt{2\pi}\sigma_x}\int_{-\infty}^{+\infty}\exp\{-\alpha_x\eta_x^2\}.\left[1 + \beta_x\eta_x + \frac{\beta_x^2\eta_x^2}{2!} + higher \quad order \quad terms\right]d\eta_x$$

With suitable substitution and ignoring higher order terms for small a_x, the above integral reduces to,

$$I_x = \frac{2e^{\gamma_x}}{\sqrt{2\pi}\sigma_x}\left[\frac{\Gamma(0.5)}{2\sqrt{\alpha_x}} + \frac{\beta_x\Gamma(1)}{2\alpha_x} + \frac{\beta_x^2\Gamma(1.5)}{4\alpha_x^{3/2}}\right],$$

where the Γ function is define as follows:

$$\Gamma(\alpha) = \int_0^\infty e^{-t}t^{\alpha-1}dt \qquad (\alpha > 0) \qquad \text{(C.3)}$$

we also make use of the following gamma function relations.

$\Gamma(0.5) = \sqrt{\pi}, \Gamma(1.0) = 1, \Gamma(\alpha + 1) = \alpha\Gamma(\alpha)$

With these relations substituted in equation C.3, the final expression for I_x becomes:

$$I_x = \frac{e^{\gamma_x}}{4\sqrt{2\pi}\sigma_x \alpha_x^{3/2}}\left[4\alpha_x\sqrt{\pi} + 4\sqrt{\alpha_x}\beta_x + \beta_x^2\sqrt{\pi}\right]$$

A similar expression can be obtained for I_y. From these expressions we can evaluate $P_{CIT}\{k+1\}$ by:

$$P_{CIT}\{k+1\} = I_x . I_y$$

C.2 Proof for the Probability of Correct Association (PCA) for Constant Acceleration Tracker (CAT)

The Alternate Method is given:

Starting from equation (5.13) in chapter 5

$$P_{CAT}\{k+1 \mid \eta_k\} = P_0^{\pi\|\eta_k\|^2} \tag{C.4}$$

Note that this probability is conditioned on the random component η_k which can be integrated out by applying the *total probability theorem.*

$p(\eta_k)$ is the two-dimensional probability density function (pdf) of the random variable η_k.

$$P(\eta_k) = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left\{-\frac{1}{2}\left[\left(\frac{\eta_x}{\sigma_x}\right)^2 + \left(\frac{\eta_y}{\sigma_y}\right)^2\right]\right\} \tag{C.5}$$

where η_x, η_y are the x and y elements of the random variable η_k, and σ_x^2, σ_y^2 are the x and y variances of the Gaussian distribution.

With the above given equations C.4 and C.5, the total probability theorem expands to the following form.

$$\begin{aligned}
P_{CAT}\{k+1\} &= P_0^{\pi|\eta_k|^2} \int_{-\infty}^{+\infty}\!\!\int_{-\infty}^{+\infty} \frac{1}{2\pi\sigma_x\sigma_y} \exp\left\{-\frac{1}{2}\left[\left(\frac{\eta_x}{\sigma_x}\right)^2 + \left(\frac{\eta_y}{\sigma_y}\right)^2\right]\right\} d\eta_x d\eta_y \\
&= \exp\left\{\pi\left(\eta_x^2 + \eta_y^2\right)\ln P_0\right\} \frac{1}{2\pi\sigma_x\sigma_y} \int_{-\infty}^{+\infty} \exp\left\{-\frac{1}{2}\left(\frac{\eta_x}{\sigma_x}\right)^2\right\} d\eta_x \int_{-\infty}^{+\infty} \exp\left\{-\frac{1}{2}\left(\frac{\eta_y}{\sigma_y}\right)^2\right\} d\eta_y \qquad \text{(C.6)} \\
&= \frac{1}{2\pi\sigma_x\sigma_y} \int_{-\infty}^{+\infty} \exp\left\{\pi\eta_x^2 \ln P_0 - \frac{1}{2}\left(\frac{\eta_x}{\sigma_x}\right)^2\right\} d\eta_x \int_{-\infty}^{+\infty} \exp\left\{\pi\eta_y^2 \ln P_0 - \frac{1}{2}\left(\frac{\eta_y}{\sigma_y}\right)^2\right\} d\eta_y
\end{aligned}$$

The separated integrals with respect to η_x, η_y are density functions which can be evaluated to closed form solutions using the fact that the area under a density function is unity. The integral with respect to x reduces to the following equation.

$$I_x = \sqrt{\frac{\pi}{\frac{1}{2\sigma_x^2} - \pi \ln P_0}} = \sqrt{\frac{2\pi\sigma_x^2}{1 - 2\pi\sigma_x^2 \ln P_0}}$$

which when substituted into equation C.6, along with its I_y counterpart gives,

$$P_{CAT}\{k+1\} = \frac{I_x I_y}{2\pi\sigma_x\sigma_y} = \frac{1}{\sqrt{\left(1 - 2\pi\sigma_x^2 \ln P_0\right)\left(1 - 2\pi\sigma_y^2 \ln P_0\right)}}$$

C.3 Proof for the Probability of Correct Association for Recovering from a False Match (PCA-FM) for Zero Velocity Tracker (ZVT)

Starting from equation 5.17 in chapter 5:

$$\begin{aligned}
P'_{ZVT}\{k+1 \mid \mathbf{v}_{k+1}, \varepsilon_k\} &= P_0^{\pi|\tilde{v}_{k+1}|} \\
&= P_0^{\pi|\mathbf{v}_{k+1}-\varepsilon_k|^2}
\end{aligned} \qquad \text{(C.7)}$$

The probability $P'_{ZVT}\{k+1 \mid \mathbf{v}_{k+1}\}$ can be formed from equation C.7 by integrating out the random term ε_k using the total probability theorem. The probability density function of ε_k is a uniform distribution inside the disk of association A, and zero outside.

$$p(\varepsilon_k) = \begin{cases} \dfrac{1}{\pi\|\mathbf{v}_k\|^2} & if \quad \varepsilon_k \quad is \quad inside \quad A \\ 0 & otherwise \end{cases} \tag{C.8}$$

when substituting equation C.8 into the total probability theorem, the integral requiring evaluation is constructed as follows.

$$\begin{aligned} P\{k+1|\mathbf{v}_{k+1}\} &= \int_A P\{k+1|\mathbf{v}_{k+1},\varepsilon_k\} p(\varepsilon_k) d\varepsilon_k \\ &= \frac{1}{\pi\|\mathbf{v}_k\|^2} \int_A P\{k+1|\mathbf{v}_{k+1},\varepsilon_k\} d\varepsilon_k \end{aligned}$$

However, a closed form solution for this integral is difficult to determine because, although $P'_{Z|T}\{k+1 \mid \mathbf{v}_{k+1},\varepsilon_k\}$ and the region of integration A are both circularly symmetric, both have different centers. If the integrand is simplified, the expression for the domain is made more complicated, and vice versa. To evaluate this integral, the circular domain of integration is approximated by a square region S (*Fig. 5.4* in chapter 5) which is centered at $\mathbf{p}_{k-1}$ and has sides of length $2v_{\max,k}$ where $v_{\max,k}$ is defined as follows:

$$v_{\max} = \max\left(|v_x|,|v_y|\right)$$

The pdf for ε_k over the square region S is:

$$p(\varepsilon_k) = \begin{cases} \dfrac{1}{(2v_{\max})^2} & if \quad \varepsilon_k \quad is \quad inside \quad S \\ 0 & otherwise \end{cases}$$

The total probability integral is therefore transformed to,

$$P'_{Z|T}\{k+1;\mathbf{v}_{k+1}\} = \frac{1}{2(v_{\max})^2}\int_S P_0^{\pi\|\mathbf{v}_{k+1}-\varepsilon_k\|^2} d\varepsilon_k \tag{A9}$$

This integral is still not in a form that can be evaluated because the variable ε_k has its origin at $\mathbf{p}_k$ while the region of integration is defined with respect to an origin at $\mathbf{p}_{k+1}$. However, if $\mathbf{v}_k = \mathbf{v}_{k+1} = \mathbf{v}_c$ is assumed, that is a constant velocity of $\mathbf{v}_c$ between frames, a change of variable from ε_k to τ_k is possible and doing so yields a tractable integral. From *Fig. 5.3* it can be seen that ε_k and τ_k are related by $\varepsilon_k = \tau_k - \mathbf{v}_c$ and also $d\tau_k = d\varepsilon_k$.

Changing the variable of integration of equation C.9 from ε_k to τ_k results in an integral that can be evaluated to a closed form solution.

$$
\begin{aligned}
P'_{Z|T}\{k+1;\mathbf{v}_{k+1}\} &= \frac{1}{(2v_{\max})^2}\int_S P_0^{\pi\|2\mathbf{v}_c-\tau_k\|^2}\,d\tau_k \\
&= \frac{1}{(2v_{\max})^2}\int_{-v_{\max}}^{+v_{\max}} \exp\{\pi\ln P_0(4v_x^2-4v_x\tau_x+\tau_x^2)\}d\tau_x \cdot \\
&\qquad \int_{-v_{\max}}^{+v_{\max}} \exp\{\pi\ln P_0(4v_y^2-4v_y\tau_y+\tau_y^2)\}d\tau_y \\
&= \frac{1}{(2v_{\max})^2}P_0^{\pi\|2\mathbf{v}_c\|^2}\int_{-v_{\max}}^{+v_{\max}} \exp\{\pi\ln P_0(-4v_x\tau_x+\tau_x^2)\}d\tau_x \cdot \\
&\qquad \int_{-v_{\max}}^{+v_{\max}} \exp\{\pi\ln P_0(-4v_y\tau_y+\tau_y^2)\}d\tau_y
\end{aligned}
$$

The integral with respect to x can be evaluates to the following form.

$$I_x = \frac{\sqrt{\pi}}{2\sqrt{-a}}\exp\{-ab^2\}\left[erf\left(\sqrt{-a}(v_{\max}+b)\right)+erf\left(\sqrt{-a}(v_{\max}-b)\right)\right]$$

where $a=\ln P_0\pi$ and $b=-2v_x$.

A similar expression can be found for I_y.

Therefore the final expression for the total probability is given as follows:

$$P'_{Z|T}\{k+1\mid\mathbf{v}_c\} = \frac{1}{(2v_{\max})^2}P_0^{\pi\|2\mathbf{v}_c\|^2}I_xI_y$$

Note that this expression is only valid for a small constant velocity v.

C.4 Proof for the Probability of Correct Association for Recovering from a False Match (PCA-FM) for Constant Acceleration and Constant Velocity Trackers (CAT and CVT)

Using *Fig. 5.4* in chapter 5:

$$\begin{aligned} -\mathbf{e}_k &= \mathbf{p}_k - \mathbf{p}_{k|k-1} \\ -\mathbf{e}_{k+1} &= \mathbf{p}_{k+1} - \mathbf{p}_{k+1|k} = \mathbf{p}_{k+1} - \left(\tilde{\mathbf{p}}_k + \mathbf{v}_k + \mathbf{a}_k + \eta_k\right) \\ &= \tilde{\mathbf{v}}_{k+1} - 2\mathbf{v}_k + \mathbf{v}_{k-1} - \eta_k \end{aligned} \tag{C.10}$$

From vector graph in *Fig. 5.5* (chapter 5), it can be shown that,

$$\tilde{\mathbf{v}}_{k+1} = -\varepsilon_k + \mathbf{v}_{k+1} + \mathbf{e}_{k+1} \tag{C.11}$$

Substituting equation C.11 in C.10 gives,

$$\begin{aligned} -\mathbf{e}_{k+1} &= -\varepsilon_k + \mathbf{v}_{k+1} + \mathbf{e}_{k+1} - 2\mathbf{v}_k + \mathbf{v}_{k-1} + \eta_k \\ 2\mathbf{e}_{k+1} &= \varepsilon_k + \eta_k - \left[\mathbf{a}_{k+1} - \mathbf{a}_k\right] \\ \mathbf{e}_{k+1} &\approx (\varepsilon_k + \eta_k)/2 \end{aligned}$$

Appendix D

A modified Principal Component Model

This section contains extra information for the material presented in Chapter 7 and 8.

D.1 Principal Component Analysis

PCA aims to transform a correlated set of observed shape-vectors to a basis of linearly uncorrelated parameters. This is equivalent to diagonalizing the shape vector covariance matrix using a similarity transform. The vector $\mathbf{dx} = (\mathbf{x} - \overline{\mathbf{x}})$ is transformed to a new basis using [10]

$$\begin{aligned}\mathbf{dx} &= \sum_{i=0}^{2N-1} b_i \mathbf{e}_i \\ &= \mathbf{Pb}\end{aligned}$$

where $\mathbf{b} = (b_0, \ldots, b_{2N-1})$ and $\mathbf{P}_{jk} = [\mathbf{e}_k]_j$.

Assuming **P** is invertible, the covariance matrix for **b** is simply

$$E(\mathbf{bb}^T) = \mathbf{P}^{-1} E(\mathbf{dxdx}^T) \mathbf{P}^{-T}$$

In order to enforce linear independence, the above covariance matrix for **b** is diagonalized by appropriate choice of $\mathbf{P}^{-1}$. This does not uniquely define **P**. A further orthogonality condition is required, namely [10]

$$\mathbf{e}_i . \mathbf{e}_j = \delta_{ij} \tag{D.1}$$

which is equivalent to $\mathbf{P}^{-1} = \mathbf{P}^T$.

D.2 Distance Metric for Splines

Equation (D.1) represents only one possible orthogonality condition. The scalar product corresponds to a choice of a standard Euclidean distance metric f(...,...) to measure the error between two sets of landmarks (x_i, y_i) and (x'_i, y'_i) where,

$$f(\mathbf{x},\mathbf{x}') = |\mathbf{x}-\mathbf{x}'|$$
$$= \left(\sum_{i=0}^{N-1} (x_i - x'_i)^2 + (y_i - y'_i)^2 \right)^{1/2}$$

Given two cubic B-splines P(s) and P'(s) defined by their N control points (x_i, y_i) and (x'_i, y'_i), a more accurate error metric d, measures the difference between corresponding points on each spline, sampled densely and uniformly over the parametric curves. The distance metric is given by (also given in chapter 8, [10]),

$$d(\mathbf{x},\mathbf{x}') = \left(\int_0^N |\mathbf{P}(s) - \mathbf{P}'(s)|^2 ds \right)^{1/2}$$
$$= \left(\int_0^N \sum_{i=0}^{N-1} ((x_i - x'_i)B_i(s))^2 ds + \int_0^N \sum_{i=0}^{N-1} ((y_i - y'_i)B_i(s))^2 ds \right)^{1/2} \quad \text{(D.2)}$$

where the $B_i(s)$ is the B-spline basis matrix elements. Eq. (D.2) simplifies to the following form:

$$d(\mathbf{x},\mathbf{x}') = \left[(\mathbf{x}-\mathbf{x}')^T \mathbf{J} (\mathbf{x}-\mathbf{x}') \right]^{1/2} \quad \text{(D.3)}$$

where the 2$N$x2$N$ symmetric metric matrix $\mathbf{J}$ is defined by (see chapter 7 for details)

$$\begin{pmatrix} J_{2i,2j} & J_{2i,2j+1} \\ J_{2i+1,2j} & J_{2i+1,2j+1} \end{pmatrix} = \begin{pmatrix} H_{i,j} & 0 \\ 0 & H_{i,j} \end{pmatrix}$$

and the NxN symmetric matrix H is given by,

$$H_{i,j} = \int_0^N B_i(s) B_j(s) ds$$

There is a unique inner product associated with this metric given by,

$$\langle \mathbf{x}, \mathbf{x}' \rangle = \mathbf{x}^T \mathbf{J} \mathbf{x}'$$

such that

$$
\begin{aligned}
d(\mathbf{x},\mathbf{x}') &= \langle \mathbf{x}-\mathbf{x}', \mathbf{x}-\mathbf{x}' \rangle^{1/2} \\
&= \left[(\mathbf{x}-\mathbf{x}')^T \mathbf{J}(\mathbf{x}-\mathbf{x}')\right]^{1/2}
\end{aligned}
$$

The inner product is used in place of the scalar product in equation (D.1) to give a more suitable orthogonality condition.

D.3 Eigenshape Analysis

The desired transformation to a set of linearly independent J-orthogonal eigenvectors is found by solving the eigenproblem (see [10] for details)

$$\mathbf{SJe}_i = \lambda_i \mathbf{e}_i \qquad \text{(D.4)}$$

where S is the training set covariance matrix $E(\mathbf{dxdx}^T)$.

Using the notation of Eq. (D.4) the following results can be easily verified.

- The vectors $\mathbf{e}_i$ are orthogonal with respect to the inner product <...,...>.
- Hence by suitable normalisation

$$< \mathbf{e}_i, \mathbf{e}_j >= \lambda_{ij}$$

or equivalently $\mathbf{P}^T \mathbf{JP} = I$

- Each shape coefficient b_i is given by projecting the shape-vector **dx** onto the line spanned by the *i*-th eigenvector (minimizing the square distance d^2 to the line). i.e

$$b_i =< \mathbf{dx}, \mathbf{e}_i >$$

- The shape coefficients are linearly uncorrelated over the training set.

$$
\begin{aligned}
E(b_i b_j) &= \mathbf{e}_i^T \mathbf{JSJe}_i =< \mathbf{e}_i, \lambda_j \mathbf{e}_j > \\
&= \lambda_j \delta_{ij}
\end{aligned}
$$

- Assuming an unbiased, homogeneous, isotropic Gaussian measurement noise model (with dense measurements uniformly spaced over the contour) as described by Blake et. al [24, 60], measurements for the shape parameters are uncorrelated.

Appendix E

Learning Motion Model Parameters

This section contains extra information for the material presented in Chapter 7.

E.1 Third Order AR Process

A third order system depends on 3 previous time steps. The dynamic motion equation is given by the following equation for a system where the mean shape space of the training set also has to be learnt,

$$\mathbf{T}(t_k) - \overline{\mathbf{T}} = A_3(\mathbf{T}(t_{k-3}) - \overline{\mathbf{T}}) + A_2(\mathbf{T}(t_{k-2}) - \overline{\mathbf{T}}) + A_1(\mathbf{T}(t_{k-1}) - \overline{\mathbf{T}}) + B_0\mathbf{w}_k \tag{E.1}$$

where **T** is the shape space state vector and $\overline{\mathbf{T}}$ is the mean shape space; A_3, A_2, A_1, B_0 are all matrices of size (*6* + *m*). That is, 6 degrees of freedom for the affine transformation and *m* number of principal components for the non-rigid shape variation. Eq. (E.1) can be represented compactly as follows [24, 25].

$$\mathbf{X}(t_k) - \overline{\mathbf{X}} = A(\mathbf{X}(t_{k-1}) - \overline{\mathbf{X}}) + B\mathbf{w}_k$$

where

$$\mathbf{X}(t_k) = \begin{bmatrix} \mathbf{T}(t_{k-2}) \\ \mathbf{T}(t_{k-1}) \\ \mathbf{T}(t_k) \end{bmatrix}, \quad \overline{\mathbf{X}} = \begin{bmatrix} \overline{\mathbf{T}} \\ \overline{\mathbf{T}} \\ \overline{\mathbf{T}} \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ a_3 & a_2 & a_1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ b_0 \end{bmatrix}$$

The coefficients of A are chosen to correspond to the damped exponential (in terms of frequency f and damping coefficient β) for a time step τ.

For a one dimensional problem, a_3, a_2, a_1 can be given by the following expressions.

$$\begin{aligned} a_1 &= \exp(-2\beta\tau) + \exp(-\beta\tau + 2\pi f\tau) + \exp(-\beta\tau - 2\pi f\tau) \\ a_2 &= -\exp(-2\beta\tau) - \exp(-\beta\tau + 2\pi f\tau) - \exp(-\beta\tau - 2\pi f\tau) \\ a_3 &= \exp(-2\beta\tau) \end{aligned}$$

b_0 can be selected suitably according to a method described in [24]. For a higher dimensional shape space A_3, A_2, A_1, B_0 becomes

$$A_3 = a_3 I, A_2 = a_2 I, A_1 = a_1 I, B_0 = \alpha b_0 H^{-1/2}$$

A sub case of harmonic motion, useful particularly for translational motion, as reported in this thesis, is the constant acceleration model in which $f = \beta = 0$, which gives the following motion model parameter values.

$$A_3 = I, \quad A_2 = -3I, \quad A_1 = 3I .$$

E.1.1 Learning Model Parameters from a Sequence of Images

The model parameters can be learnt from a known image sequence as described by Blake et. al. [24, 25] for a second order model. This method can be easily extended to a third order model as summarised below.

Given a training set $\{\mathbf{T}_1, \ldots, \mathbf{T}_M\}$ of shapes spaces from an image sequence, learn the parameters A_3, A_2, A_1, B_0 for a third order AR process that describes the dynamics of the moving shape.

The log-likelihood function for the multi-variate normal distribution of the 3rd order dynamic system is given by (assuming the mean $\overline{\mathbf{T}}$ is unknown),

$$L(\mathbf{T}_1, \ldots, \mathbf{T}_M | A_1, A_2, A_3, C, \overline{\mathbf{T}}) = -\frac{1}{2} \sum_{k=4}^{M} \left| B_0^{-1} (\mathbf{T}_k^{'} - A_3 \mathbf{T}_{k-3}^{'} - A_2 \mathbf{T}_{k-2}^{'} - A_1 \mathbf{T}_{k-1}^{'}) \right|^2 - (M-3)\log(\det B_0) \qquad \text{(E.2)}$$

where

$$\mathbf{T}_k^{'} = \mathbf{T}_k - \overline{\mathbf{T}}, \qquad C = B_0 B_0^{T}$$

The form of equation (E.2) is non-linear since the mean also has to be estimated. The non-linearity can be removed by the following substitution,

$$\mathbf{D} = (I - A_3 - A_2 - A_1)\overline{\mathbf{T}}$$

Now the motion parameters can be estimated by minimizing the following expression,

$$L(\mathbf{T}_1, \ldots, \mathbf{T}_M | A_1, A_2, A_3, C, \mathbf{D}) = -\frac{1}{2} \sum_{k=4}^{M} \left| B_0^{-1} (\mathbf{T}_k - A_3 \mathbf{T}_{k-3}^{'} - A_2 \mathbf{T}_{k-2} - A_1 \mathbf{T}_{k-1} - \mathbf{D}) \right|^2 - (M-3)\log(\det B_0) \qquad \text{(E.3)}$$

Minimising the log-likelihood L leads to the estimation of the dynamical parameters A_3, A_2, A_1, B_0. Maximising first with respect to A_3, A_2, A_1, it will be shown that separability holds. Maxima with respect to A_3, A_2, A_1 turn out to be independent of the values of C. Equivalently to maximising L, $tr(Z)$ can be minimised with respect to A_3, A_2, A_1.

$$\text{If } f(A_1, A_2, A_3, \mathbf{D}) = \sum_{k=4}^{M} \left| B_0^{-1}(\mathbf{T}_k - A_3\mathbf{T}_{k-3} - A_2\mathbf{T}_{k-2} - A_1\mathbf{T}_{k-1}) \right|^2$$

this can be expressed as

$$f(A_1, A_2, A_3, \mathbf{D}) = tr(ZC^{-1})$$

where

$$Z = \sum_{k=4}^{M} (\mathbf{T}_k - A_3\mathbf{T}_{k-3} - A_2\mathbf{T}_{k-2} - A_1\mathbf{T}_{k-1} - \mathbf{D})(\mathbf{T}_k - A_3\mathbf{T}_{k-3} - A_2\mathbf{T}_{k-2} - A_1\mathbf{T}_{k-1} - \mathbf{D})^T$$

For the purpose of finding A_3, A_2, A_1, B_0 can be effectively set to the identity matrix ($B_0 = I$) for minimizing $tr(Z)$; where

$$tr(Z) = \sum_{k=4}^{M} \left| (\mathbf{T}_k - A_3\mathbf{T}_{k-3} - A_2\mathbf{T}_{k-2} - A_1\mathbf{T}_{k-1} - \mathbf{D}) \right|^2$$

Setting to 0 the derivatives of $tr(Z)$ with respect to A_3, A_2, A_1 respectively shows that the minimum must satisfy the following simultaneous equations.

$$\begin{aligned} R_{03} - A_3R_{33} - A_2R_{23} - A_1R_{13} - \mathbf{D}R_3 &= 0 \\ R_{02} - A_3R_{32} - A_2R_{22} - A_1R_{12} - \mathbf{D}R_2 &= 0 \\ R_{01} - A_3R_{31} - A_2R_{21} - A_1R_{11} - \mathbf{D}R_1 &= 0 \\ R_0 - A_3R_3 - A_2R_2 - A_1R_1 - (M-3)\mathbf{D} &= 0 \end{aligned}$$

where

$$R_i = \sum_{k=4}^{M} \mathbf{T}_{k-i}, \quad R_{ij} = \sum_{k=4}^{M} \mathbf{T}_{k-i}\mathbf{T}_{k-j}^T, \quad R_{ij}' = R_{ij} - \frac{1}{(M-3)} R_i R_j^T \qquad i = 0,1,2,3$$

eliminating $\mathbf{D}$ gives the following 3 equations

$$
\begin{aligned}
R_{03}' - A_3 R_{33}' - A_2 R_{23}' - A_1 R_{13}' &= 0 \\
R_{02}' - A_3 R_{32}' - A_2 R_{22}' - A_1 R_{12}' &= 0 \\
R_{01}' - A_3 R_{31}' - A_2 R_{21}' - A_1 R_{11}' &= 0
\end{aligned}
$$

Solving the above equations gives the following estimates.

$$
\begin{aligned}
\hat{A}_1 &= \left\{ \frac{[R_{03}' - R_{02}'(R_{32}')^{-1}R_{33}'][R_{23}' - R_{21}'(R_{31}')^{-1}R_{33}'] - [R_{03}' - R_{01}'(R_{31}')^{-1}R_{33}'][R_{23}' - R_{22}'(R_{32}')^{-1}R_{33}']}{[R_{13}' - R_{12}'(R_{32}')^{-1}R_{33}'][R_{23}' - R_{21}'(R_{31}')^{-1}R_{33}'] - [R_{13}' - R_{11}'(R_{31}')^{-1}R_{33}'][R_{23}' - R_{22}'(R_{32}')^{-1}R_{33}']} \right\} \\
\hat{A}_2 &= \left\{ \frac{[R_{03}' - R_{02}'(R_{32}')^{-1}R_{33}'][R_{13}' - R_{11}'(R_{31}')^{-1}R_{33}'] - [R_{03}' - R_{01}'(R_{31}')^{-1}R_{33}'][R_{13}' - R_{12}'(R_{32}')^{-1}R_{33}']}{[R_{23}' - R_{22}'(R_{32}')^{-1}R_{33}'][R_{13}' - R_{11}'(R_{31}')^{-1}R_{33}'] - [R_{23}' - R_{21}'(R_{31}')^{-1}R_{33}'][R_{13}' - R_{12}'(R_{32}')^{-1}R_{33}']} \right\} \\
\hat{A}_3 &= \left\{ R_{03}'(R_{33}')^{-1} - \hat{A}_2 R_{23}'(R_{33}')^{-1} - \hat{A}_1 R_{13}'(R_{33}')^{-1} \right\} \\
\hat{\mathbf{D}} &= \frac{1}{M-3}\left(R_0 - \hat{A}_3 R_3 - \hat{A}_2 R_2 - \hat{A}_1 R_1 \right)
\end{aligned}
$$

If required for the standard form of the AR process, the mean $\overline{\mathbf{T}}$ is estimated from

$$\hat{\overline{\mathbf{T}}} = (I - \hat{A}_3 - \hat{A}_2 - \hat{A}_1)^{-1}\hat{\mathbf{D}}$$

It remains to estimate B_0 which is obtained as the square root of $C = B_0 B_0^T$. Rewriting (E.3) as

$$L = -\frac{1}{2}tr(ZC^{-1}) + \frac{1}{2}(M-3)\log(\det C^{-1}),$$

and fixing $A_2 = \hat{A}_2, A_1 = \hat{A}_1, A_3 = \hat{A}_3, \mathbf{D} = \hat{\mathbf{D}}$, and extremising with respect to C^{-1} (using the identity $\partial(\det M)/\partial M = (\det M)M^{-1}$) gives

$$\hat{C} = \frac{1}{M-3}\left(R_{00} - \hat{A}_3 R_{30} - \hat{A}_2 R_{20} - \hat{A}_1 R_{10} \right)$$

E.2 Second Order System AR Process

Similar to the third order process, a second order AR process is given by the following equation,

$$\mathbf{T}(t_k) - \overline{\mathbf{T}} = A_2(\mathbf{T}(t_{k-2}) - \overline{\mathbf{T}}) + A_1(\mathbf{T}(t_{k-1}) - \overline{\mathbf{T}}) + B_0 \mathbf{w}_k$$

Where the current state estimate depends only on the past 2 time steps.

By a similar procedure given above, the state transition matrix A is given by,

$$\mathbf{X}(t_k)=\begin{bmatrix}\mathbf{T}(t_{k-1})\\ \mathbf{T}(t_k)\end{bmatrix},\quad \overline{\mathbf{X}}=\begin{bmatrix}\overline{\mathbf{T}}\\ \overline{\mathbf{T}}\end{bmatrix},\quad A=\begin{bmatrix}0 & 1\\ a_2 & a_1\end{bmatrix},\quad B=\begin{bmatrix}0\\ b_0\end{bmatrix}$$

where for a one dimensional state, the motion parameters are given by,

$$a_1 = 2\exp(-\beta\tau)\cos(2\pi f\tau),$$
$$a_2 = \exp(-2\beta\tau)$$

For a higher dimensional shape space the parameters are simply,

$$A_2 = a_2 I, A_1 = a_1 I, B_0 = \alpha b_0 H^{-1/2}$$

A constant velocity model is obtained by setting $f=\beta=0$, which gives,

$$A_2 = -I, A_1 = 2I$$

where the size of the identity matrix (I) depends on the size of the state (example: 6 for an affine shape space).

E.2.1 Learning Motion Parameters for a Second Order Model

Using similar procedure as for the third order model, the second order model motion parameters can be shown to be as follows [24],

$$\hat{A}_2 = \left(R'_{02} - R'_{01}(R'_{11})^{-1}R'_{12}\right)\left(R'_{22} - R'_{21}(R'_{11})^{-1}R'_{12}\right)^{-1}$$
$$\hat{A}_1 = \left(R'_{01} - \hat{A}_2 R'_{21}\right)\left(R'_{11}\right)^{-1}$$
$$\hat{\mathbf{D}} = \frac{1}{M-2}\left(R_0 - \hat{A}_2 R_2 - \hat{A}_1 R_1\right)$$

The mean of the training set and $C = BB^T$ are estimated as follows,

$$\hat{\overline{\mathbf{T}}} = (I - \hat{A}_2 - \hat{A}_1)^{-1}\hat{\mathbf{D}}$$

$$\hat{C} = \frac{1}{M-2}\left(R_{00} - \hat{A}_2 R_{20} - \hat{A}_1 R_{10} - \hat{\mathbf{D}} R_0^T\right)$$

E.3 First Order AR Process

First order AR process is given by the following equation,

$$\mathbf{T}(t_k) - \overline{\mathbf{T}} = A_1(\mathbf{T}(t_{k-1}) - \overline{\mathbf{T}}) + B_0\mathbf{w}_k$$

Where the current state estimate depends only on the past one time step.

By a similar procedure given above, the state transition matrix A is given by,

$$\mathbf{X}(t_k) = [\mathbf{T}(t_k)], \qquad \overline{\mathbf{X}} = [\overline{\mathbf{T}}] \qquad A = [a_1], \qquad B = [b_0]$$

where for a one dimensional state, the motion parameters are given by,

$$a_1 = \exp(-\beta\tau)$$

For a higher dimensional shape space the parameters are simply (same process as for the other models),

$$A_1 = a_1 I$$

A constant position model is obtained by setting $\beta = 0$, which gives,

$$A_1 = I$$

E.3.1 Learning Motion Parameters for a First Order Model

Using similar procedure as before, it can be shown that,

$$\hat{A}_1 = \left(R_{10} - R_1R_0\right)\left(R_{11} - R_1R_1\right)^{-1}$$
$$\hat{\mathbf{D}} = \left(R_0 - \hat{A}_1R_1\right)$$

The mean of the training set and $C = BB^T$ are estimated as follows,

$$\hat{\overline{\mathbf{T}}} = (I - \hat{A}_1)^{-1}\hat{\mathbf{D}}$$

$$\hat{C} = \frac{1}{M-2}\left(R_{00} - \hat{A}_1R_{10} - \hat{\mathbf{D}}R_0^T\right)$$

Appendix F

Two Contour Trackers

We give details of the Condensation and the Baumberg's tracker in this section. A brief introduction was only given in Chapter 8 due to space constraints.

F.1 Condensation Algorithm

Given that the tracking process at each time step is a self contained iteration of factored sampling [24, 102, 104], the output of an iteration will be a weighted, time stamped sample set, denoted $\{\mathbf{s}_t^{(n)}, n = 1,...,N\}$ with weights $\pi_t^{(n)}$, representing approximately the conditional state density $p(\mathbf{x}_t \mid Z_t)$ at time *t*, where $\mathbf{x}_t, Z_t$ represent the curve state at time step *t*, and measurement history up to time step *t* respectively.

The sample set is obtained by using a prior density, and the effective prior for time step *t* is $p(\mathbf{x}_t \mid Z_{t-1})$. It is derived from the sample set representation $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}), n = 1,....N\}$ of $p(\mathbf{x}_{t-1} \mid Z_{t-1})$, the output from the previous time step, to which prediction must then be applied (see Figs. 8.1 & 8.2 in chapter 8, and [104] for details).

The iterative process begins from the output from time-step *t-1*, which is the weighted sample set $\{(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}), n = 1,..., N\}$. Appropriate initial values for sample set and weights are chosen before the iteration commences. The aim is to maintain, at successive time-steps, sample sets of fixed size *N*, so that the algorithm can be guaranteed to run within a given computational resource. The first operation therefore is to sample (with replacement) *N* times from the set $\{\mathbf{s}_{t-1}^{(n)}\}$, choosing a given element with probability $\pi_{t-1}^{(n)}$. Some elements, especially those with high weights, may be chosen several times, leading to identical copies of elements in the new set. Others with relatively low weights may not be chosen at all.

Each element chosen from the new set is now subject to the predictive steps. First, an element undergoes drift and, since this is deterministic, identical elements in the new set undergo the same drift. The second predictive step, diffusion, is random and identical elements now split because each undergoes its own independent Brownian motion step. At this stage, the sample set $\{\mathbf{s}_t^{(n)}\}$ for the new time step has been generated but, as yet, without its weights; it is approximately a fair random sample from the effective prior density $p(\mathbf{x}_t \mid Z_{t-1})$ for time step *t*. Finally, the observation step from factored sampling is applied, generating weights from the observation density $p(\mathbf{x}_t | Z_t)$ to obtain the sample set representation $\{(\mathbf{s}_t^{(n)}, \pi_t^{(n)})\}$ of state density for time t. See [24] for more details.

Figure (8.1) in chapter 8 gives a synopsis of the algorithm. Note the use of the cumulative weights $\{c_{t-1}^{(j)}\}$ (constructed in step 3) to achieve efficient sampling in step 1. After any time step, it is possible to report on the current state, for example by evaluating some moment of the state density as shown.

One of the striking properties of the Condensation algorithm is its simplicity, compared with the Kalman filter, despite its generality. Largely this is due to the absence of the Riccati equation, which appears in the Kalman filter for the propagation of covariance [5, 6]. The Riccati equation is relatively complex computationally but is not required in the Condensation algorithm, which instead deals with variability by sampling, involving the repeated computation of a relatively simple propagation formula.

F.1.1 Curve Motion

Contours are represented by using second order B-splines. A typical curve **r**(*s*,*t*) is given by

$$\mathbf{r}(s,t) = \left(\mathbf{B}(s).\mathbf{Q}_x(t), \mathbf{B}(s).\mathbf{Q}_y(t)\right) \quad for \quad 0 \le s \le L \tag{F.1}$$

where the vector B(*s*) is the vector containing the quadratic B-spline basis functions, $\mathbf{Q}_x, \mathbf{Q}_y$ are vectors of B-spline control point coordinates, and *L* is the number of spans.

The spline space is transformed into shape space using the following expression,

$$\begin{pmatrix} \mathbf{Q}_x \\ \mathbf{Q}_y \end{pmatrix} = W\mathbf{T} + \begin{pmatrix} \overline{\mathbf{Q}}_x \\ \overline{\mathbf{Q}}_y \end{pmatrix}, \tag{F.2}$$

where *W* is the shape matrix and **T** is the shape space. See [24, 25, 102] for details. $\overline{\mathbf{Q}}_x, \overline{\mathbf{Q}}_y$ are the *x* and *y* control points of a template shape. In this chapter we refer **T** as the 'shape space' including rigid and non rigid components. The space which includes *only* non-rigid (deformable) components is referred to as 'non-rigid shape space' or non-rigid (deformable) shape parameters. The space which includes *only* the rigid components is referred to as 'rigid shape space' or rigid shape parameters.

F.1.2 Dynamic Model

The dynamic model employed is a second order process, represented as

$$\mathbf{x}_t - \overline{\mathbf{x}} = A(\mathbf{x}_{t-1} - \overline{\mathbf{x}}) + B\mathbf{w}_t \tag{F.3}$$

where $\mathbf{w}_t$ are independent vectors of independent standard normal variables, the state-vector is given by,

$$\mathbf{x}_t = \begin{pmatrix} \mathbf{T}_{t-1} \\ \mathbf{T}_t \end{pmatrix}, \quad \overline{\mathbf{x}} = \begin{pmatrix} \overline{\mathbf{T}} \\ \overline{\mathbf{T}} \end{pmatrix},$$

and $\overline{\mathbf{x}}$ is the mean value of the state, and A, B are matrices representing the deterministic and stochastic components of the dynamic model respectively, and ideally they are learned from test sequences [25]. See Appendix E for learning process.

F.1.3 Observation model

In one dimension, observations reduce to a set of scalar positions $\{\mathbf{z} = (z_1, z_2, ..., z_M)\}$ and the observation density has the form $p(\mathbf{z} \mid x)$ where x is one-dimensional position. This can be given by,

$$p(\mathbf{z} \mid x) \propto 1 + \frac{1}{\sqrt{2\pi}\sigma\alpha} \sum_m \exp - \frac{\nu_m^2}{2\sigma^2} \tag{F.4}$$

where $\alpha = q\lambda$ and $\nu_m = z_m - x$. Peaks in the density function corresponds to measured features and the state density will tend to be reinforced in the Condensation algorithm at such points.

In a two dimensional image (as in contour tracking), the set of observations z is, in principle, the entire set of features visible in the image. However, an important aspect of earlier systems in achieving real time performance ([127], [84], [11], [23]) has been the restriction of measurement to a sparse set of lines normal to the tracked curve. The two apparently conflicting ideas can be resolved as follows.

The observation density $p(\mathbf{z} \mid \mathbf{x})$ in two dimensions describes the distribution of a (linearly) parameterized image curve z(s), given a hypothetical shape in the form of a curve r(s) 0 < s < 1, (with L spans) represented by a shape parameter x. The two dimensional density be derived as an extension of the one dimensional case. It is assumed that a mapping $g(s)$ is known that associates each point z(s) on the image curve with a point r($g(s)$) on the shape. In practice this mapping is set up by tracing normals from the curve **r**. Next, the one-dimensional density (F.4) is approximated in a more amenable form that neglects the possibility of more than one feature lying inside the search interval:

$$p(\mathbf{z} \mid x) \propto \exp - \frac{1}{2\sigma^2} f(\nu_1; \mu) \quad where \quad f(\nu; \mu) = \min(\nu^2, \mu^2), \tag{F.5}$$

$\mu = \sqrt{2}\sigma \log(1/\sqrt{2\pi}\alpha\sigma)$ is a spatial scale constant, and ν_1 is the ν_m (for each measurement on the curve) with smallest magnitude, representing the feature lying closest to the hypothesized position x. A natural extension to two dimension is then,

$$p(\mathbf{z} \mid \mathbf{x}) = Z \exp - \frac{1}{2r} \int_0^L f(\mathbf{z}_1(s) - \mathbf{r}(s); \mu) ds \tag{F.6}$$

in which r is a variance constant and $\mathbf{z}_1(s)$ is the closest associated feature to r(s):

$$\mathbf{z}_1(s) = \mathbf{z}(s') \quad where \quad s' = \arg\{ \min_{s' \in g^{-1}(s)} |\mathbf{r}(s) - \mathbf{z}(s')| \ \} \tag{F.7}$$

The assumption is made that the variation of Z with x is slow compared with the other term in (F.6) so that Z can be treated as constant [102].

The observation density (F.6) can be computed via a discrete approximation, the simplest being:

$$p(\mathbf{z} \mid \mathbf{x}) \propto \exp\left\{ - \sum_{m=1}^{M} \frac{1}{2rM} f(\mathbf{z}_1(s_m) - \mathbf{r}(s_m); \mu) \right\}, \tag{F.8}$$

where $s_m = m / M$. This is simply the product of one-dimensional densities (F.4) with $\sigma = \sqrt{rM}$, evaluated independently along M curve normals.

Despite the attractiveness of the Condensation algorithm, there are factors that limit the performance of this algorithm. The reader is referred to [115, 148] for details.

F.2 Baumberg's and Hogg's Tracker

F.2.1 Spline Representation

The tracking framework for Baumberg's tracker was given in Fig. 8.3 in chapter 8. The contour is represented by a cubic B-spline with N control points (equally spaced around the contour) as follows:

$$\mathbf{Q} = P\mathbf{b} + \overline{\mathbf{Q}} \tag{F.9}$$

where P is an $2N$ x m matrix of eigenvectors (see chapter 7 for details) and $\overline{\mathbf{Q}}$ is the mean contour shape of a training sequence. The non-rigid shape parameters are given by $\mathbf{b} = (b_0, ..., b_{m-1})^T$.

A contour in the model frame is projected into the image frame by rotation, scaling and translation using the expression

$$\mathbf{Q}_i = \begin{pmatrix} X_i \\ Y_i \end{pmatrix} = \Phi \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} o_x \\ o_y \end{pmatrix} \tag{F.10}$$

where the 2 x 2 alignment matrix Φ is given by,

$$\Phi = \begin{pmatrix} a_x & -a_y \\ a_y & a_x \end{pmatrix} = \begin{pmatrix} f\cos\theta & -f\sin\theta \\ f\sin\theta & f\cos\theta \end{pmatrix} \tag{F.11}$$

where f, θ are the scaling and rotation factor in relation to the mean shape respectively.

The contour is given by (in spline space, represented by control points)

$$\mathbf{Q} = (X_0, Y_0, \ldots, X_{N-1}, Y_{N-1})^T \tag{F.12}$$

which represents the 2D control points of the B spline contour in the image frame. Hence the state space consists of m non-rigid shape parameters b_i, the origin of the object (o_x, o_y) and the alignment parameters a_x, a_y incorporating rotation and scaling. The state parameters are related to the spline vector $\mathbf{Q}$ by

$$\mathbf{Q} = \mathbf{D}(a_x, a_y)(P\mathbf{b} + \overline{\mathbf{Q}}) + \mathbf{o} \tag{F.13}$$

where

$$\mathbf{o} = \underbrace{(o_x, o_y, \ldots, o_x, o_y)^T}_{N times} \quad \text{and} \quad \mathbf{D} = \begin{pmatrix} \Phi & & 0 \\ & \ddots & \\ 0 & & \Phi \end{pmatrix}$$

D is a *2N* x *2N* rotation and scaling matrix.

F.2.2 Stochastic Model

F.2.2.1 Non-Rigid Shape Parameters

The shape part of the state vector is modeled as a simple discrete stochastic process as follows:

$$b_i^{(k)} = b_i^{(k-1)} + w_i^{(k-1)} \tag{F.14}$$

where $w_i^k \sim N(0, \mu_i)$ and b_i^k models the *i*'th deformable (non-rigid) parameter value at frame *k* and the noise term w_i^k is a zero mean, normally distributed random variable with variance μ_i.

F.2.2.2 Origin (Object Centroid) Model

The origin of the object is assumed to follow the dynamic equation (a simple constant velocity model)

$$\frac{d}{dt}\begin{pmatrix} o_x \\ \dot{o}_x \end{pmatrix} = \begin{pmatrix} \dot{o}_x \\ 0 \end{pmatrix} + \begin{pmatrix} v_x \\ w_x \end{pmatrix} \tag{F.15}$$

where $v_x \sim N(0, q_v)$ and $w_x \sim N(0, q_w)$. A corresponding model is used for o_y.

F.2.2.3 Alignment Parameters

The alignment parameters (scaling and rotation) follow the following motion model.

$$\begin{pmatrix} a_x^{(k+1)} \\ a_y^{(k+1)} \end{pmatrix} = \begin{pmatrix} a_x^{(k)} \\ a_y^{(k)} \end{pmatrix} + \begin{pmatrix} w_{ax} \\ w_{ay} \end{pmatrix} \tag{F.16}$$

where $w_{ax}, w_{ay} \sim N(0, q_a)$.

F.2.3 Filter Update Process

F.2.3.1 Non-Rigid (Deformable) Shape Filter Covariance Update

The following recursive equation is used to update the covariance matrix for the non-rigid shape filter. See [10] for details.

$$P_k^{-1}(+) = P_k^{-1}(-) + [\mathbf{D}P]^T [r\mathbf{J}^{-1}]^{-1} [\mathbf{D}P] \tag{F.17}$$

where **J** is the metric matrix (discussed in chapter 7). Using appropriate assumptions (refer [10]), the above equation reduces to,

$$P_k^{-1}(+) = P_k^{-1}(-) + f^2 r^{-1} I \tag{F.18}$$

where *f* is the scaling factor and *r* is the measurement variance constant.

Assuming $P_k^{-1}(-)$, P_0 are diagonal, the system can be decoupled into *m* independent 1D Kalman filters. The covariance update equation for the *i*'th filter now becomes

$$[\sigma_i(+)]^{-1} = [\sigma_i(-)]^{-1} + r_i^{-1} \tag{F.19}$$

where $r_i^{-1} = f^2 r^{-1}$, and $\sigma_i = [P_k]_{i,i}$ is simply the variance of the current estimate for b_i.

The corresponding shape parameter update equation is given by,

$$\hat{b}_i(+) = \hat{b}_i(-) + \left(\frac{\sigma_i(-)}{r_i + \sigma_i(-)}\right) db_i \tag{F.20}$$

where $db_i = [P^T \mathbf{D}^T \mathbf{Q}]_i - \hat{b}_i(-)$ is the observed change in the i'th non-rigid shape parameter.

F.2.3.2 Updating the Origin

The *x* and *y* component of the origin are filtered independently. The measurement model for the *x* component of the origin, assuming all other parameters are fixed at their current estimates, is given by

$$p'_i = o_x + (\nu_k)_{2i} \tag{F.21}$$

where the noise term $\nu_k \sim N(0, R_k)$, and R_k is the contour measurement noise matrix.
Similarly for the *y* component,

$$q'_i = o_y + (\nu_k)_{2i+1}. \tag{F.22}$$

The measurements $\mathbf{p}'$ are calculated from the observed contour points $\mathbf{p}$ (these are obtained by casting normals to the estimated contour and then selecting the best point available, or alternatively obtained by some other suitable process) using

$$\mathbf{p}' = \mathbf{p} - \mathbf{D}(\hat{a}_x, \hat{a}_y) G (P\hat{\mathbf{b}} + \overline{\mathbf{Q}}) \tag{F.23}$$

where G is a *2n* x *2N* sparse matrix mapping the control points to regularly spaced points (*n*) on the curve [11]. The update equations for the origin follow a standard Kalman Filter.

F.2.3.3 Updating the Alignment

If the origin and the shape parameters are fixed at their current estimates, the measurement model for the alignment parameters is given by,

$$\mathbf{p} - G\hat{\mathbf{o}} = H\begin{pmatrix} a_x \\ a_y \end{pmatrix} + \mathbf{v}_k \tag{F.24}$$

where H is the $2n$ x 2 measurement matrix defined by

$$\begin{pmatrix} H_{2i,0} & H_{2i,1} \\ H_{2i+1,0} & H_{2i+1,1} \end{pmatrix} = \begin{pmatrix} s_{2i} & -s_{2i+1} \\ s_{2i+1} & s_{2i} \end{pmatrix}, \text{ where } s = G(P\hat{\mathbf{b}} + \overline{\mathbf{Q}})$$

The estimates $\hat{a}_x, \hat{a}_y$ and the 2 x 2 covariance matrix are updated with the corresponding Kalman filter equations. The alignment parameters are not assumed to be independent although for simplicity the system noise is assumed isotropic.

F.2.3.4 Updating Non-Rigid Shape Parameters

Each shape parameter is filtered independently for computational convenience [11].

Writing $\Delta\mathbf{p} = \mathbf{p} - \hat{\mathbf{p}}$, the measurement model for the i'th non-rigid shape filter is given by

$$\Delta\mathbf{p} = \mathbf{h}^{(i)}(b_i - \hat{b}_i) + \mathbf{v}_k \tag{F.25}$$

where the vector $\mathbf{h}^{(i)}$ is an $2n$ x 1 measurement matrix given by

$$[\mathbf{h}^{(i)}]_j = [\mathbf{D}(\hat{a}_x, \hat{a}_y)GP]_{ji}$$

The covariance update equation for each filter is given by equation (F.19), where the measurement variance for the i-th shape parameter, r_i, is now defined by,

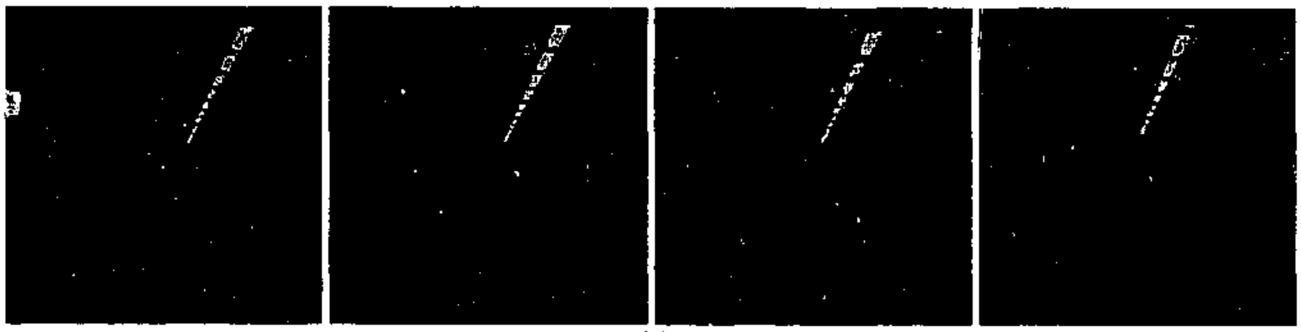
$$r_i^{-1} = (\mathbf{h}^{(i)})^T R_k^{-1}\mathbf{h}^{(i)} \tag{F.26}$$

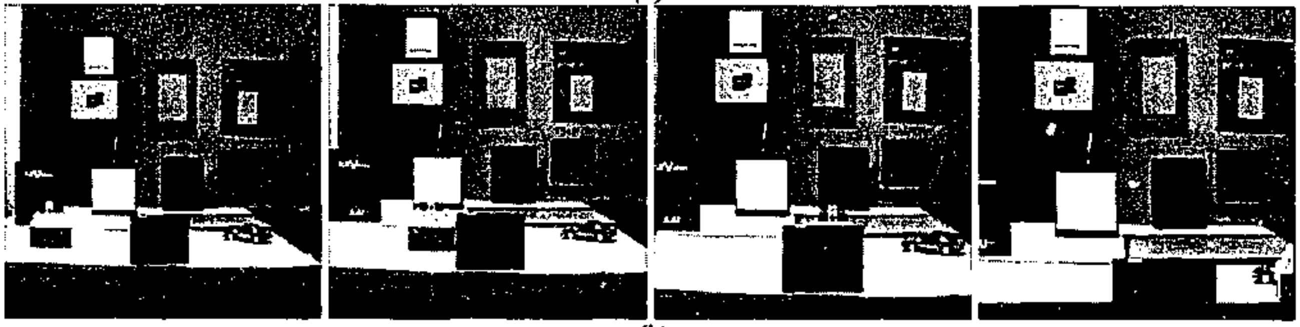
The state update equation for each filter is given by

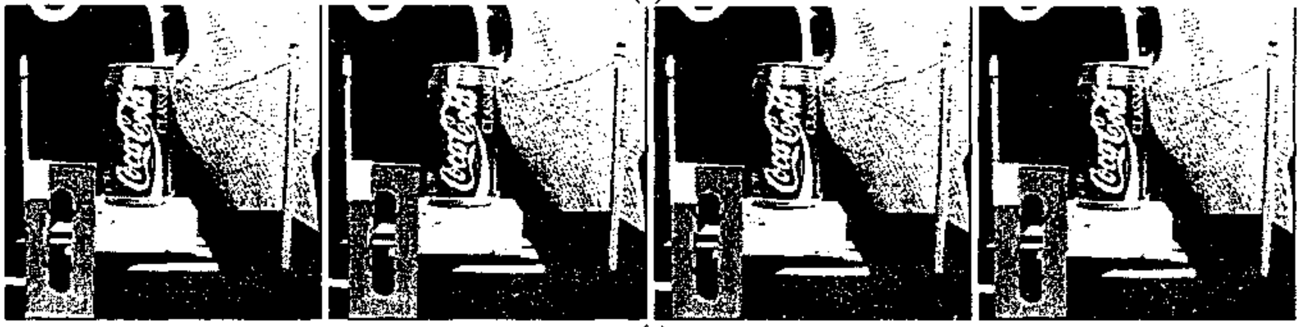
$$\hat{b}_i(+) = \hat{b}_i(-) + \sigma_i(+)((\mathbf{h}^{(i)})^T R_k^{-1}(\Delta\mathbf{p})) \tag{F.27}$$

where R_k is the contour measurement noise matrix [60]. Further details of the tracker can be found in [10].

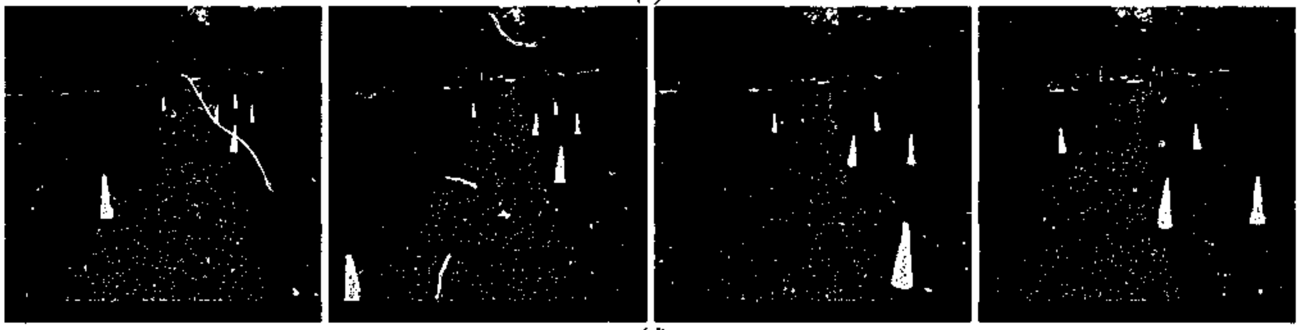
Appendix G

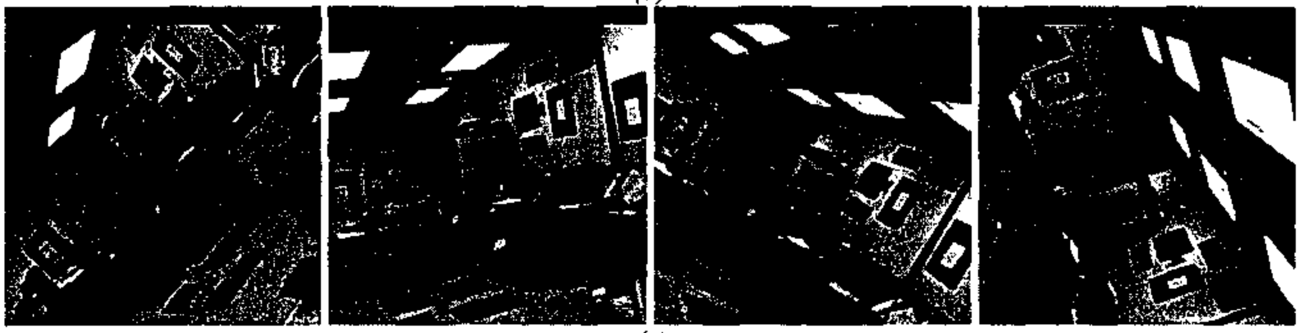

(a)


(b)

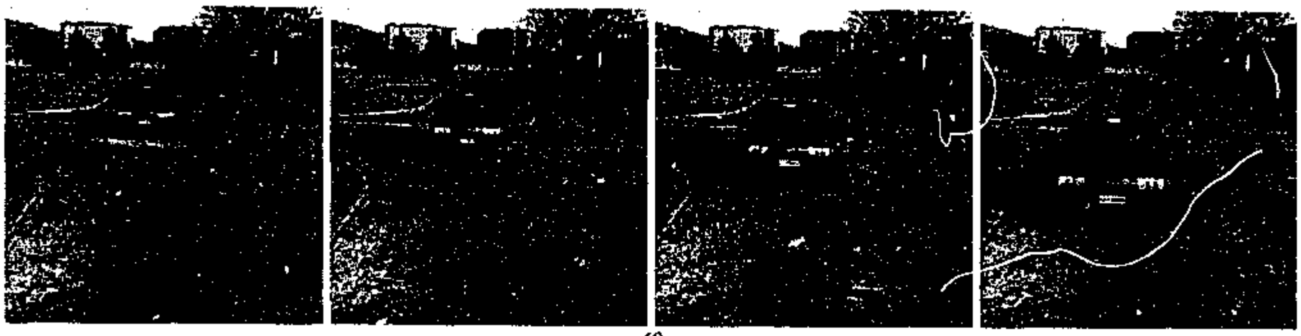


(c)

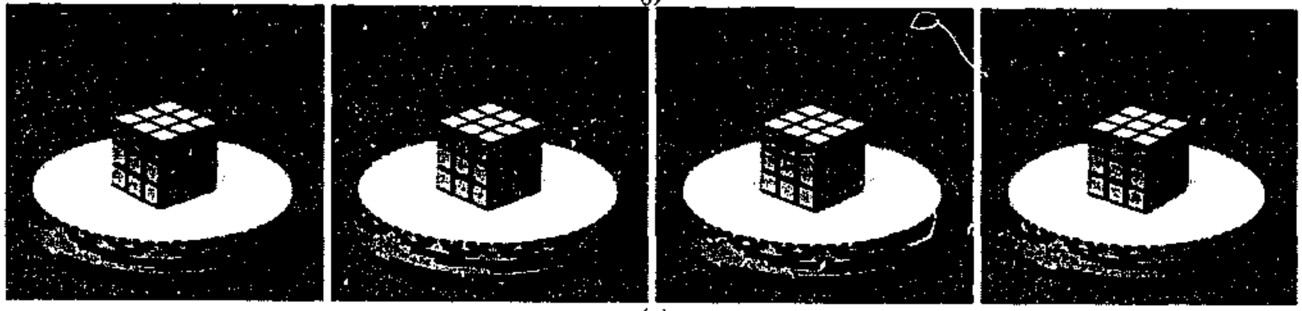

(d)


(e)

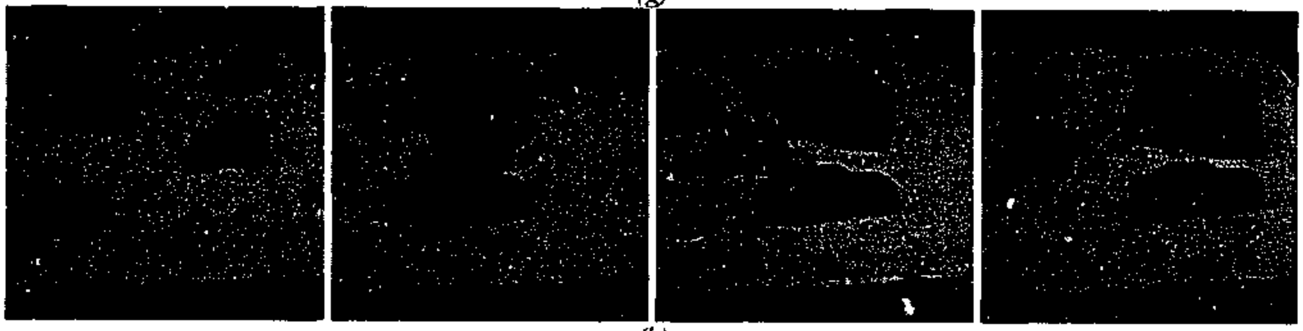
Continued on the next page.



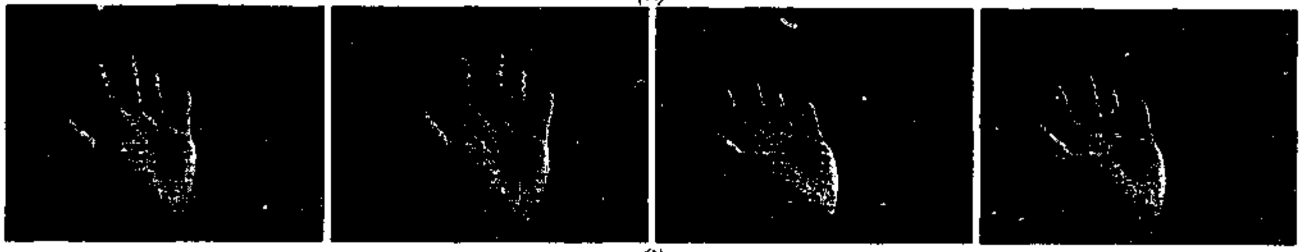
(f)



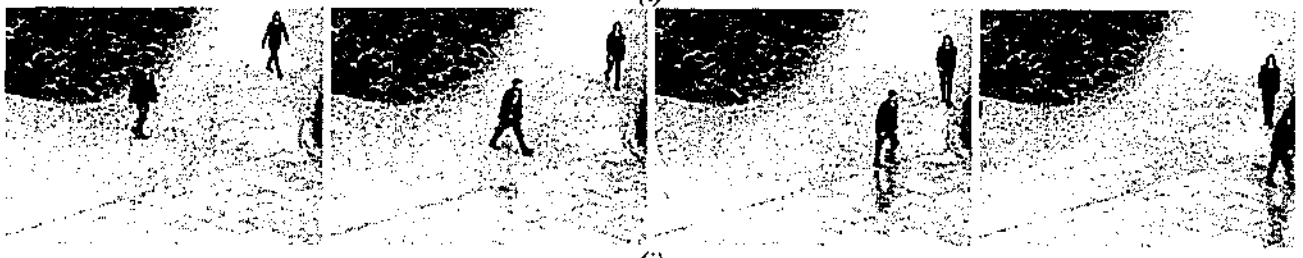
(g)



(h)



(i)



(j)

Figure G.1: Some frames of image sequences considered for the project: (a) Indoor cone (8 frames), (b) UMASS Lab (11 frames), (c) Coke (20 frames), (d) Outdoor cone (20 frames), and (e) PUMA (30 frames), (f) Road (50 frames), (g) Rubic (20 frames), (h) Toy car (9 frames), (i) Hand (75 frames), and (j) Walking man (50 frames) sequences.

Bibliography

[1] N. Ayache, I. Cohen and I. Herlin, "Medical image tracking", In A. Blake and A. Yuille, editors, *Active Vision*, pp.285-302, MIT, 1992

[2] A. Bab-Hadiashar and D. Suter, "Robust Optic Flow Computation", *International Journal of Computer Vision*, Vol.29(1), pp.59-77, 1998

[3] K. D. Baker and G. D. Sullivan, "Performance assessment of model based tracking", *Workshop on Applications of Computer Vision*, pp.28-35, 1992

[4] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of Optical Flow Techniques", *IJCV*, Vol(12), No. 1, pp. 43-77, 1994

[5] Y. Bar-Shalom and X. R. Li, Estimation and Tracking: principles, techniques, and software, *Artech House*, Boston, MA, 1993

[6] Y. Bar-Shalom and Y. Fortmann, Tracking and Data Association, volume 179 of *Mathematics in Science and Engineering*, AP, Boston, 1988

[7] R. Bartels, J. Beatty, and B. Barsky, An Introduction to Splines for use in Computer Graphics and Geometric Modeling, *Morgan Kaufmann*, 1987

[8] B. Bascle, P. Bouthemy, R. Deriche and F. Meyer, "Tracking complex primitives in an image sequence", In *Proc. International Conference on Pattern Recognition*, pp.426-431, Oct. 1994.

[9] B. Bascle and R. Deriche, "Region tracking through image sequences", In *Proc. 5th Int. Conf. on Computer Vision*, pp.302-307, 1995

[10] A. M. Baumberg, "Learning deformable models for tracking human motion", PhD thesis, *School of Computer Studies, University of Leeds*, 1995

[11] A. M. Baumberg and D. C. Hogg, "An efficient method for contour tracking using active shape models", *Proc. of the IEEE Workshop on Motion of Non-Rigid and Articulate Objects*, pp.194-199, 1994

[12] A. M. Baumberg and D. C. Hogg, "Learning flexible models from image sequences", *ECCV-94*, pp.299-308. 1994

[13] A. M. Baumberg and D. C. Hogg, "An adaptive eigenshape model", *British Machine Vision Conference*, Vol.1, pp.87-96, 1995

[14] A. M. Baumberg and D. C. Hogg, "Generating spatio-temporal models from examples", *Image and Vision Computing*, vol.14, pp.525-532, 1996

[15] S. S. Beauchemin and J. L. Barron, "The computation of optical flow", *ACM computing surveys*, Vol.27, No. 3, Sept. 1995, pp.433-467.

[16] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms", *CVPR-98*, pp.232-237

[17] M. J. Black and P. Anandan, "A framework for the robust estimation of optical flow", In *Proc. 4th Int. Conf. on Computer Vision*, pp.231-236, 1993

[18] M. J. Black and A. D. Jepson, "EigenTracking: Robust matching and tracking of articulated objects using a view based representation", *Int. Jour. of Computer Vision*, pp.63-84, 26(1), 1998

[19] M. J. Black and A. D. Jepson, "A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions", In *Proc. 5th European Conf. Computer Vision* , 1, pp.909-924, 1998

[20] M. J. Black, Y. Yacoob, A. Jepson and D. Fleet, "Learning parameterized models of image motion", In *Proc. Conf. Computer Vision and Pattern Recognition*, pp.561-567, San Juan, Puerto Rico, 1997

[21] M. J. Black and Y. Yacoob, "Recognizing facial expressions in image sequences using local parameterized models of image motion", *IJCV*, Vol.25(1), pp.23-48, 1997

[22] A. Blake, R. Curwen, and A. Zisserman, "Affine-invarient contour tracking with automatic control of spatiotemporal scale", In *Proc. 4th Int. Conf. on Computer Vision*, pp.66-75, 1993

[23] A. Blake, R. Curwen, and A. Zisserman, "A framework for spatio-temporal control in the tracking of visual contours", *Int. Journal of Computer Vision*, 11(2), pp.127-145, 1993

[24] A. Blake and M. Isard, Active contours, *Springer*, 1998

[25] A. Blake, M. Isard, and D. Reynard, "Learning to track the visual motion of contours", *Artificial Intelligence*, Vol.78, pp.101-134, 1995

[26] A. Blake, B. North and M. Isard, "Learning multi-class dynamics", *Advances in Neural Information Processing Systems, 11*, in press, MIT Press, 1999

[27] A. Blake and A. Yuille, Active Vision, *The MIT Press*, 1992

[28] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients", *IEEE Transactions on Automatic Control*, Vol.33(8), pp.780-783, 1988

[29] F. Bobick, "Video annotation: Computers watching video", *ACCV-95*, pp.19-23

[30] A. Bobick and A. Wilson, "A state-based technique for the summarisation and recognition of gesture", In *Proc. 5th Int. Conf. on Computer Vision*, pp.382-388, 1995

[31] R. A. Boie and I. J. Cox, "Two dimensional optimum edge detection using matched and wiener filters for machine vision", In *IEEE First ICCV*, pp.450-456, June 1987

[32] C. Bregler, "Learning and recognising human dynamics in video sequences", In *Proc. Conf. Computer Vision and Pattern Recognition*, pp.568-574, 1997

[33] C. Bregler and J. Malik, "Tracking people with twists and exponential maps", In *Proc. CVPR*, pp.8-15, 1998

[34] C. Bregler and S. Omohundro, "Nonlinear manifold learning for visual speech recognition", In *Proc. 5th Int. Conf. on Computer Vision* , pp.494-499, 1995

[35] F. Bremond and M. Thonnat, "Tracking multiple non-rigid objects in a cluttered scene", In *Proc. 10th Scandinavian Conference on Image Analysis (SCIA)*, 1997

[36] T. J. Broida and R. Chellappa, "Estimating the kinematics and structure of a rigid object from a sequence of monocular images", *IEEE Trans. on PAMI*, Vol.13(6), pp.497-513, 1991

[37] M.J. Brooks, W. Chojnacki, L. Baumela, "Determining the egomotion of an uncalibrated camera from instantaneous optical flow", *Journal Optical Society of America A*,14,10, October 1997, pp.2670-2677.

[38] P. Burlina and R. Chellappa, "Temporal analysis of motion in video sequences through predictive operators", *IJCV*, Vol.28(2), pp.175-192, 1998

[39] Q. Cai and J. K. Aggarwal, "Automatic Tracking of Human Motion in Indoor Scenes Across Multiple Synchronized Video Streams", *Proc. of Intl. Conf. on Computer Vision*, pp.356-362, 1998

[40] J. Canny, "Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.8(6), pp.679-698, 1986

[41] K. C. Chang, "Multitarget tracking with adaptive detection thresholds", *IEEE Trans. on Aerospace and Electronic Systems*, Vol.32, No.1, pp.401-406, 1996

[42] K. C. Chang, S. Mori, and C. Y. Chong, "Evaluating a Multiple Hypothesis Multi-target Tracking Algorithm", *IEEE Trans. on Aerospace and Electronic Systems*, Vol.30(2), pp.578-591, April 1994

[43] K. C. Chang, S. Mori, and C. Y. Chong, "Performance evaluation of track initiation in dense target environments", *IEEE Trans. on Aerospace and Electronic Systems*, Vol.30(1), pp.213-220, Jan. 1994

[44] D. Chetverikov and J. Verestóy, "Tracking feature points: A new algorithm", In *Proc. International Conf. on Pattern Recognition*, pp.1436-1438, 1998

[45] L. D. Cohen and I. Cohen, "Finite element methods for active contour models and balloons for 2-D and 3-D images", *IEEE Trans on PAMI*, Vol.15(11), pp.1131-1147, 1993

[46] I. Cohen and G. Medioni, "Detecting and tracking moving objects for video surveillance", *CVPR-99*, pp.319-325, 1999

[47] T.F.Cootes, A.Hill, C.J.Taylor, J.Haslam, "The Use of Active Shape Models for Locating Structures in Medical Images", *Image and Vision Computing*, Vol.12, No.6, pp.355-366, 1994

[48] T.F.Cootes and C.J.Taylor, "Combining point distribution models with shape models based on finite element analysis", *Image and Vision Computing*, Vol.13, No.5, pp.403-409, 1995

[49] T. F. Cootes, C.J.Taylor, D.H.Cooper and J.Graham, "Training Models of Shape from Sets of Examples", In *Proc. British Machine Vision Conference*, Springer-Verlag, pp.9-18, 1992

[50] T. F. Cootes, C.J.Taylor, "Active Shape Models – 'Smart Snakes' " In *Proc. British Machine Vision Conference*, Springer-Verlag, pp.266-275, 1992

[51] T. F. Cootes, D. Cooper, C. J. Taylor and J. Graham, "Active Shape Models - Their Training and Application", *Computer Vision and Image Understanding*, Vol. 61, No. 1, pp. 38-59, Jan. 1995

[52] T. F. Cootes, C. J. Taylor, A. Lanitis, D. H. Cooper and J. Graham, "Building and Using Flexible Models Incorporating Grey-Level Information", *Proc. ICCV*, pp.242-246, 1993

[53] T. F. Cootes, G.J. Edwards and C.J.Taylor, "Active Appearance Models", In *Proc. European Conference on Computer Vision 1998* (H.Burkhardt & B. Neumann Ed.s). Vol. 2, pp. 484-498

[54] I. J. Cox, "A Review of Statistical Data Association Techniques for Motion Correspondence", *International Journal on Computer Vision*, vol. 10, no. 1, pp.53-66, 1993

[55] I. J. Cox and J. J. Leonard, "Modelling a Dynamic Environment using a Multiple Hypothesis Approach", *Journal of Artificial Intelligence*, vol. 66, no.1, pp.311-344, 1994

[56] I. J. Cox and M. L. Miller, "On Finding Ranked Assignments with Application to Multi-Target Tracking and Motion Correspondence", *IEEE Trans. on Aerospace and Electronic Systems*, vol. 32, no. 1, pp.486-489, 1995

[57] I. J. Cox, J. M. Rehg, and S. Hingorani, "A Bayesian Multiple Hypothesis Approach to Contour Grouping and Segmentation", *International Journal of Computer Vision*, vol. 11, no. 1, pp.5-24, 1993

[58] I. J. Cox and S. L. Hingorani, "An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 2, pp.138-150, Feb. 1996

[59] I. J. Cox, J. M. Rehg, and S. Hingorani, "A Bayesian Multiple Hypothesis Approach to Contour Grouping", *European Conference on Computer Vision*, pp.72-77, 1992

[60] R. Curwen, Dynamic and Adaptive Contours, *PhD thesis, University of Oxford*, 1993

[61] R. Curwen and A. Blake, "Dynamic contours: real-time active splines", In *A. Blake and A. Yuille, editors, Active Vision*, pp.39-58, MIT, 1992

[62] J. Denzler and H. Niemann, "Active Rays: A New Approach to Contour Tracking", *International Journal of Computing and Information Technology*", Vol.4(1), pp.9-16, 1996

[63] J. Denzler and H. Niemann, "Evaluating the performance of active contour models for real-time object tracking", In *Second Asian Conference on Computer Vision*, volume 2, pages II/341-II/345, 1995

[64] R. Deriche and G. Giraudon, "A Computational Approach for Corner and Vertex Detection", *International Journal of Computer Vision*, Vol.10(2), pp.101-124, 1993

[65] R. Deriche and O. Faugeras, "Tracking Line Segments", *Proc. on European Conference on Computer Vision (ECCV)*, pp.259-268, 1990

[66] E. Dickmanns, "Expectation-based dynamic scene understanding", In *A. Blake and A. Yuille, editors, Active Vision*, pp.303-336, MIT, 1992

[67] L. Dreschler and H. H. Nagel, "On the Selection of Critical Points and Local Curvature Extrema of Region Boundaries for Interframe Matching", In *International Conference on Pattern Recognition*, pp.542-544, 1982

[68] L. Dreschler and H. H. Nagel, "Volumetric model and 3D trajectory of a moving car derived from monocular TV-frame sequence of a street scene", *Computer Vision, Graphics and Image Processing*, Vol. 20, No.3, pp.199-228, 1982

[69] M. P. Dubuisson, S. Lakshmanan, and A. K. Jain, "Vehicle segmentation and classification using deformable templates", *IEEE Trans. on PAMI*, Vol.18, No.3, pp.293-308, 1996

[70] M. Etoh and Y. Shirai, "Segmentation and 2D Motion Estimation by Region Fragments", ICCV-93, May 1993, pp.192-199

[71] G. J. Edwards, T. F. Cootes and C. J. Taylor, "Face Recognition Using Active Appearance Models", in *Proc. European Conference on Computer Vision*, Vol. 2, pp.581-595, 1998

[72] N. Ferrier, S. Rowe, and A. Blake, "*Real-time traffic monitoring*", In *Proc. 2nd IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, pp.81-88, Dec. 1994

[73] J. Ferryman, A. Worrall, G. Sullivan, and K. Baker, "*A generic deformable model for vehicle recognition*", *Proc. British Machine Vision Conference*, Birmingham, pp.127-136, 1995

[74] B. Friedland, Control System Design, An Introduction to State Space Methods, *McGraw Hill International Edition*, Singapore, 1987

[75] D. Gennery, "Visual tracking of known three-dimensional objects", *Int. J. Computer Vision*, Vol.7(3), pp.243-270, 1992

[76] D. Gibbins, G. Newsam, M. J. Brooks, "Detecting suspicious background changes in video surveillance of busy scenes" *Third IEEE Workshop on Applications of Computer Vision*, December 1996, Sarasota, Florida, USA, pp.22-26.

[77] S. Gil, R. Milanese and T. Pun, "Combining multiple motion estimates for vehicle tracking", *ECCV-98*, pp.307-320

[78] S. Gil, R. Milanese and T. Pun, "Feature selection for object tracking in traffic scenes", *SPIE International Symposium on Smart Highways*, pp.253-266, 1994

[79] R. Gonzales and P. Wintz, "Digital Image Processing", *Addison-Wesley*, 1987

[80] G. D. Hager and P. Belhumeur, "Efficient Region Tracking With Parametric Models of Geometry and Illumination", *IEEE PAMI*, Vol.20(10), pp.1025-1039, 1998

[81] G. D. Hager and K. Toyama, "XVision: A Portable Substrate for Real-Time Vision Applications", In *Computer Vision and Image Understanding*, Vol.69(1), pp.23-37, 1998

[82] G. D. Hager and C. Rasmussen, "Joint Probabilistic Techniques for Tracking Multi-Part Objects", In *CVPR-98*, pp.16-21

[83] I. Haritaoglu, D. Harwood, and L. Davis, "W4S: A real-time system for detecting and tracking people in 2.5D", In *Proc. 5th European Conf. Computer Vision*, pp.877-892, 1998

[84] C. G. Harris, "Tracking with rigid models", In *Blake, A. and Yuille, A., editors, Active Vision*, pp.59-74 MIT, 1992

[85] C. G. Harris and M. J. Stephens, "Combined Corner and Edge Detector", In *Proceedings of the Fourth Alvey Vision Conference*, Manchester, pp.147-151, 1988

[86] C. G. Harris and C. Stennett, "Rapid - A video-rate object tracker", In *Proc. 1st British Machine Vision Conference*, pp.73-78, 1990

[87] J. Haslam, C. J. Taylor, and T. F. Cootes, "A Probabalistic Fitness Measure for Deformable Template Models", In *Proc. British Machine Vision Conference*, BMVA Press, pp.33-42, 1994

[88] A. J. Heap, "Learning deformable shape models for object tracking", *PhD thesis, School of Computer Studies*, University of Leeds, 1997

[89] A. J. Heap and D. C. Hogg, "Wormholes in shape space: tracking through discontinuous changes in shape", *6th IEEE International Conference on Computer Vision (ICCV'98)*, pp.344-349

[90] A. J. Heap and D. C. Hogg, "Extending the point distribution method using polar coordinates", *Image and Vision Computing*, Vol.14(8), pp.589-600, 1996

[91] A. J. Heap and D. C. Hogg, "Towards 3D hand tracking using a deformable model", *Proceedings of the Second International Conference on Automatic Face & Gesture Recognition*, pp.140-145, 1996

[92] A. Hill, A. Thornham and C.J. Taylor, "Model-Based Interpretation of 3D Medical Images", *4th British machine Vision Conference*, Guilford, England, pp.339-348, Sept.1993

[93] A. Hill, and C.J. Taylor, "Automatic Landmark Generation for Point Distribution Models", In *Proc. BMVC*, pp.429-438, 1994

[94] A. Hill, T.F. Cootes, C.J. Taylor and K. Lindley, "Medical Image Interpretation: A Generic Approach using Deformable Templates", *Journal of Medical Informatics*, 19, no.1, pp.47-59, 1994

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

[96] B. Horn and B. Schunk, "Determining optical flow", *Artificial Intelligence*, Vol.17, pp.185-203, 1981

[97] 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*, pp.93-101, 1993

[98] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the Hausdorff distance", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.15, No.9, pp.850-863, 1993

[99] S. Huwer and H. Niemann, "2D-object tracking based on projective histograms", *ECCV-98*, pp.861-876

[100] S. Intille and A. Bobick, "Closed-world tracking", In *Proc. 5th Int. Conf. on Computer Vision*, pp.672-678, June 1995

[101] S. Intille, J. Davis, and A. Bobick, "Real-time closed-world tracking", In *Proc. Conf. Computer Vision and Pattern Recognition*, pp.697-703, 1997

[102] M. A. Isard, "Visual motion analysis by probabilistic propagation cf conditional density", *PhD thesis, Dept. of Engineering Science*, Oxford University, UK, 1995

[103] M. Isard and A. Blake, "Visual tracking by stochastic propagation of conditional density", In *Proc. 4th European Conf. Computer Vision* , pp.343-356, 1996

[104] M. Isard and A. Blake, "Condensation: conditional density propagation for visual tracking", *Int. J. Computer Vision*, 28(1), pp.5-28, 1998

[105] M. Isard and A. Blake, "ICondensation: Unifying low-level and high-level tracking in a stochastic framework", In *Proc. 5th European Conf. Computer Vision* , pp.893-908, 1998

[106] M. Isard and A. Blake, "A mixed-state Condensation tracker with automatic model switching", In *Proc. 6th Int. Conf. on Computer Vision*, pp.107-112, 1998

[107] M Isard and A. Blake, "A smoothing filter for Condensation", In *Proc. 5th European Conf. Computer Vision*, 1, pp.767-781, 1998

[108] G. Jacob, J. A. Noble, and A. Blake, "Robust contour tracking in echocardiographic sequences", *International Conference on Computer Vision*, pp.408-413, 1998

[109] A. Jepson and M. J. Black, "Mixture models for optical flow computation", *Proc. Conf. Computer Vision and Pattern Recognition*, pp.760-761, 1993

[110] N Johnson and D C Hogg, "Learning the distribution of object trajectories for event recognition", *Image & Vision Computing*, 14(8), pp.609-615, August 1996

[111] S. X. Ju, M. J. Black, and A. D. Jepson, "Skin and Bones: Multi-layer, locally affine, optical flow and regularization with transparency", *Proc. CVPR-96*, pp.307-314.

[112] S. B. Kang, R. Szeliski and H. Y. Shum, "A parallel feature tracker for extended image sequences", *Computer Vision and Image Understanding*, Vol. 67(3), pp.296-310, 1997

[113] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active contour models", *Int. Journal of Computer Vision*, pp.321-331, 1988

[114] R. Kaucic, B. Dalton and A. Blake, "Real-time lip-tracking for audio-visual speech recognition applications". *In Proc. 4th European Conf. Computer Vision*, pp.376-387, 1996

[115] O. King and D. Forsyth, "How does Condensation behave with a finite number of samples ?", *ECCV-2000*, pp.695-709

[116] L. Kitchen and A. Rosenfeld, "Gray level corner detection", *Pattern Recognition Letters*, pp.95-102, 1982

[117] D. Koller, D. Danilidis, and H.-H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes", *Int. Journal of Computer Vision*, Vol.10(3), pp.257-281, 1993

[118] D. Koller, J. Weber, and J. Malik, "Robust multiple car tracking with occlusion reasoning", *ECCV'94*, pp.189-196, 1994

[119] T. Kurien, "Issues in the design of practical multi-target tracking algorithms", *Y. Bar-Shalom, ed., Multitarget Multisensor Tracking: Advanced Applications*, Artech House, pp.43-83, 1990

[120] K. F. Lai, "Deformable contours: modeling, extraction, detection and classification", *PhD thesis, University of Wisconsin-Madison*", USA, 1994

[121] K. F. Lai and R. T. Chin, "Deformable contours – modeling and extraction", *IEEE Trans. PAMI*, Vol.17(11), pp.1084-1090, 1995

[122] A. Lanitis, C. J. Taylor, and T. F. Cootes, "An automatic face identification system using flexible appearance models", *Image and Vision Computing*, Vol.13 No.5, pp.393-402, 1995

[123] A. Lanitis, C. J. Taylor, and T. F. Cootes, "Automatic Interpretation and Coding of Face Images Using Flexible Models", *IEEE PAMI*, Vol.19 No.7, pp.743-756, July 1997

[124] E. J. Lund, J. G. Balchen, and B. A. Foss, "Multiple Model Estimation with Inter-Residual Distance Feedback", *9th Symposium on Identification and System Parameter Estimation*, Budapest, Hungary, Vol.2, pp.889-894, 1991

[125] X. R. Li and Y. Bar-Shalom, "Performance prediction of tracking in clutter with nearest neighbour filters", *Proceedings of the SPIE*, Vol. 2235, pp.429-440, 1994

[126] X. R. Li and Y. Bar-Shalom, "Performance prediction of the Interacting Multiple Model Algorithm", *IEEE Trans. on Aerospace and Electronic Systems*, Vol.29, No.3, pp.755-771, July 1993

[127] D. Lowe, "Robust model-based motion tracking through the integration of search and estimation", *Int. J. Computer Vision*, Vol.8(2), pp.113-122, 1992

[128] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", *Proc. Seventh International Joint Conference on Artificial Intelligence*, pp.674-679, 1981

[129] S. Maybank, A. Worrall and G. Sullivan, "A filter for visual tracking based on a stochastic model for driver behaviour", In *Proc. 4th European Conf. Computer Vision*, pp.540-549, 1996

[130] P. S. Maybeck, Stochastic Models, Estimation and Control, Vol 1 & 2, *Academic Press*, New York, 1982

[131] G. L. Mealy and W. Tang, "Application of Multiple Model Estimation to a Recursive Terrain Height Correlation System", *IEEE Trans. on Automatic Control*, Vol.28(3), pp.323-331, 1983

[132] F. Meyer and P. Bouthamy. "Region-based tracking in an image sequence", In G. Sandini (ed.), *Proc. ECCV*, pp.476-484, 1992

[133] M. Mirmehdi and T. J. Ellis, "Parallel approach to tracking edge segmentation in dynamic scenes", *Image and Vision Computing*, Vol.11(1), pp.35-48, 1993

[134] F. Mokhtarian and R. Suomela, "Robust image corner detection through curvature scale space", *IEEE Trans. on PAMI*, Vol.20(12), pp.1376-1381, 1998

[135] S. Mori, K. C. Chang, C. Y. Chong, and K. P. Dunn, "Prediction of Track Purity and Accuracy," *IEEE Trans. on Automatic Control*, Vol. 40, No. 5, pp.953-959, May, 1995

[136] S. Mori, K. C. Chang, and C. Y. Chong, "Performance Analysis of Optimal Data Association - with applications to multiple target tracking", In *Y. Bar-Shalom, ed., Multitarget-Multisensor Tracking: Applications and Advances*, Vol. II, Artech House, 1992

[137] K. G. Murty, "An Algorithm for Ranking all the assignments in order of increasing cost", *Operational Research*, Vol.16, pp.682-687,1968

[138] P. M. Ngan and A. M. McIvor, "Performance analysis of two tracking methods", *Image and Vision Computing New Zealand*, pp.197-202, 1996

[139] P. M. Ngan, "Image sequence analysis 1995-96", Tech. Report 571, *Industrial Research Limited, Auckland, New Zealand*, June 1996

[140] J. A. Noble. "Finding Corners", *Image and Vision Computing*, Vol.6(2), pp.121-128, 1988

[141] N. Oliver, A. Pentland and F. Berard, "LAFTER: Lips and Face Real Time Tracker", *CVPR-97*, pp.123-129

[142] N. Paragios and R. Deriche, "A PDE-based level-set approach for detection and tracking of moving objects", In *Proc. 6th Int. Conf. on Computer Vision*, pp.1139-1145, 1998

[143] A. Pentland and B. Horowitz, "Recovery of non-rigid motion and structure", *IEEE Trans on PAMI*, Vol.13(7), pp.730-742, 1991

[144] A. Pentland and S. Sclaroff, "Closed form solutions for physically based shape modeling and recognition", *IEEE Trans on PAMI*, Vol.13(7), pp.715-729, 1991

[145] N. Peterfreund, "The Velocity Snake: Deformable Contour for Tracking in Spatio-Velocity Space", *Computer Vision and Image Understanding*, Vol.73, No.3, pp.346-356, March 1999

[146] N. Peterfreund, " Robust Tracking of Position and Velocity with Kalman Snakes", *IEEE Trans. on PAMI*, Vol. 21, No.6, pp.564-569, June 1999

[147] N. Peterfreund, "The PDAF based Active Contour", *ICCV-99*, pp.227-233

[148] V. Philomin, R. Duraiswamy, and L. Davis, "Quasi-random sampling for Condensation", *ECCV-2000*, pp.134-149

[149] B. Rao, "Data association methods for tracking systems", In *A. Blake and A. Yuille, editors, Active Vision*, pp.91-105, MIT, 1992

[150] J. M. Rehg, "Visual analysis of high DOF articulated objects with application to hand tracking", *PhD Thesis, School of Computer Science*, Carnegie Mellon University, PA, USA, 1995

[151] J. M. Rehg and T. Kanade, "Visual tracking of high dof articulated structures: an application to human hand tracking", *European Conf. Computer Vision*, pp.35-46, 1994

[152] D. B. Reid, "An Algorithm for Tracking Multiple Targets", *IEEE Trans. on Automatic Control*, Vol.24, no. 6, pp.843-854, Dec. 1979

[153] I. Reid and D. Murray, "Active tracking of foveated feature clusters using affine structure", *Int. J. Computer Vision*, Vol.18(1), pp.41-60, 1996

[154] P.Remagnino, A. Baumberg, T. Grove, D. Hogg, T.Tan, A.Worrall, and K.Baker, "An Integrated Traffic and Pedestrian Model based Vision System", *Proceedings of BMVC-97*, pp.380-389, 1997

[155] D. Reynard, A. Wildenberg, A. Blake, and J. Marchant, "Learning dynamics of complex motions from image sequences", In *Proc. 4th European Conf. Computer Vision*, pp.357-368, 1996

[156] J. Rittscher and A. Blake, "Classification of human body motion", *ICCV-99*, pp.634-639

[157] J. M. Roberts, "Attentive Visual Tracking and Trajectory Estimation for Dynamic Scene Segmentation", *Phd Thesis, Dept. of Elect. and Comp. Science*, University of Southampton, UK, 1994

[158] S. R. Rogers, "Diffusion analysis of track loss in clutter", *IEEE Trans. on Aerospace and Electronic Systems*, Vol.27(2), pp.380-387, March 1991

[159] K. Rohr, "Towards Model-Based Recognition of Human Movements in Image Sequences", *CVGIP: Image Understanding*, Vol.59, No.1, Jan., pp.94-115, 1994

[160] R. Rosales and S. Sclaroff, "3D Trajectory Recovery for Tracking Multiple Objects and Trajectory Guided Recognition of Actions", *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, (CVPR), pp.117-123, 1999

[161] R. Rosales and S. Sclaroff, "Improved Tracking of Multiple Humans with Trajectory Prediction and Occlusion Modeling", *Proc. IEEE Workshop on the Interpretation of Visual Motion*, June, 1998

[162] P. L. Rosin, "Augmenting corner descriptors", *Graphical models and image processing*, Vol.58(3), pp.286-294, 1996

[163] S. Rowe, Robust feature search for active tracking, *PhD thesis, University of Oxford*, 1996

[164] G. I. Salama and A. L. Abbott, "Monocular and Binocular tracking", *CVPRIP-98*, pp.326-329

[165] S. Sclaroff and J. Isidoro, "Active Blobs", *Proc. International Conference on Computer Vision*, January, pp.1146-1153, 1998

[166] S. Sclaroff and A. P. Pentland, "Modal matching for correspondence and recognition", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.17, No.6, pp.545-561, 1995

[167] I. K. Sethi and R. Jain, "Finding trajectories of feature points in a monocular image sequence", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.9, pp.56-73, 1987

[168] L. S. Shapiro, Affine analysis of image sequences, *Cambridge University Press*, New York, 1995

[169] L. S. Shapiro, H. Wang, and J. M. Brady, "A Matching and Tracking Strategy for Independently-Moving, Non-Rigid Objects", In *3rd British Machine Vision Conference*, pp.306-315, 1992

[170] J. Shi and J. Malik, "Motion segmentation and tracking using normalized cuts", In *Proc. ICCV-98*, pp.1154-1160

[171] J. Shi and C. Tomasi, "Good features to track", In *CVPR-94*, pp.593-600

[172] S. Smith, "Asset-2: Real-time motion segmentation and shape tracking", *In Proc. 5th Int. Conf. on Computer Vision*, pp.237-244, 1995

[173] S.M. Smith and J.M. Brady, "SUSAN - a new approach to low level image processing", *Int. Journal of Computer Vision*, Vol.23(1), pp.45-78, May 1997

[174] B. Southall, B. Buxton, J. Marchant, "On the performance characterisation of image segmentation algorithms", ECCV-2000, pp.351-365

[175] G. Sullivan, "Visual interpretation of known objects in constrained scenes", *Phil. Trans. R. Soc. Lond. B.*, B(337), pp.109-118, 1992

[176] D. Terzopoulos and R. Szeliski, "Tracking with Kalman snakes", *In A. Blake and A. Yuille, editors, Active Vision*, pp.3-20, The MIT Press, 1992

[177] D. Terzopoulos and D. Metaxas, "Tracking non-rigid 3D objects", In *A. Blake and A. Yuille, editors, Active Vision*, pp.75-89, The MIT Press, 1992

[178] D. Terzopoulos and D. Metaxas, "Dynamic 3D models with local and global deformations: deformable superquadrics", *IEEE Trans. on PAMI*, Vol.13(7), pp.703-714, 1991

[179] P. Tissainayagam and D. Suter, "Comparison of Corner Extractors for Tracking Objects in Long Image Sequences", Proc. *International Workshop on Image Analysis and Information Fusion (IAIF '97)*, pp. 171-181, Adelaide, Australia. Oct. 1997.

[180] P. Tissainayagam and D. Suter, "Visual Tracking and Motion Determination using the IMM Algorithm", *International Conference on Pattern Recognition (ICPR '98)*, Brisbane, Australia, pp.289-291, 1998

[181] P. Tissainayagam and D. Suter, "Visual Tracking with Multiple Motion Models", *IAPR Machine Vision Applications (MVA '98)*, pp.414-417, Chiba, Japan. Nov. 1998

[182] P. Tissainayagam and D. Suter, "Object Tracking in Image Sequences using the Multiple Hypothesis Approach", *First International Workshop on Computer Vision, Pattern Recognition and Image Processing (CVPRIP '98)* ", pp. 473-475, Durham, NC, USA. Oct. 1998.

[183] P. Tissainayagam and D. Suter, "Performance Prediction Analysis for Visual Tracking Algorithms", *Irish Machine Vision and Image Processing Conference (IMVIP '99)*, Dublin, Ireland, Sept.1999, pp.117-134

[184] P. Tissainayagam and D. Suter, "Tracking multiple object contour with automatic motion model switching", *International Conference on Pattern Recognition*, 2000, Barcelona, Spain, pp.1146 - 1149

[185] P. Tissainayagam and D. Suter, "Visual Tracking with Automatic Motion Model Switching", *International Journal of Pattern Recognition*, Vol(34), pp.641-660, 2001.

[186] P. Tissainayagam and D. Suter, "Performance Prediction Analysis for Visual Tracking Algorithms", *International Journal of Computer Vision and Image Understanding*, 2001 (accepted).

[187] P. Tissainayagam and D. Suter, "Motion Model Selection for Visual Feature Tracking", *Technical Report MECSE-1997-4*, Dept. of Electrical and Computer Systems Engineering, Monash University, Clayton, Australia. 1997.

[188] P. Tissainayagam and D. Suter, "Performance analysis of Point-Feature Trackers", *Technical Report MECSE-1998-6*, Dept. of Electrical and Computer Systems Engineering, Monash University, Clayton, Australia. 1998.

[189] D. M. Tobin and P. S. Maybeck, "Substantial Enhancements to a Multiple Model Adaptive Estimator for Target Image Tracking", In *IEEE Conference on Decision and Control*, Vol.3, pp.2002-2011, 1987

[190] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features", Carnegie Mellon University, *Tech. Report CMU-CS-91-132*, April 1991

[191] T. Tommasini, A. Fusiello, E. Trucco and V. Roberto, "Making good features track better", *CVPR-98*, pp.178-183

[192] P. Torr, "Geometric Motion Segmentation and Model Selection", *Philosophical Transactions of the Royal Society A*, pp.1321-1340, 1998

[193] P. Torr, "An assessment of information criteria for motion model selection", In *Proc. Conf. Computer Vision and Pattern Recognition*, pp.47-52, 1997

[194] S. Tsuji, M. Osada and M. Yachida, "Tracking and segmentation of moving objects in dynamic line images", *IEEE Trans. Patten Anal. Machine Intell.*, Vol. PAMI-2, No. 6, pp. 516-522, Nov. 1980

[195] M. Turk and A. Pentland, "Eigenfaces for recognition", *Journal of Cognitive Neuroscience*, Vol.3(1), pp.71-96, 1991

[196] J. Verestoy and D. Chetverikov "Comparative Performance Evaluation of Four Feature Point Tracking Techniques", *Proc. 22nd Workshop of the Austrian Pattern Recognition Group*, Illmitz, Austria, pp.255-263, 1998

[197] S. Wachter and H. H. Nagel, "Tracking persons in monocular image sequences", *Computer Vision and Image Understanding*, Vol.74(3), pp.174-192, 1999

[198] J. Weber and J. Malik, "Rigid body segmentation and shape description from dense optical flow under weak perspective", In *Proc. 5th Int. Conf. on Computer Vision*, pp.251-256, 1995

[199] G. Welch and G. Bishop, "An Introduction to the Kalman Filter", Course notes, *TR 95-041*, Dept. of Computer Science, University of North Carolina, Chapel Hill, NC, USA.

[200] A. Worrall, R. Marslin, G. Sullivan and K. Baker, "Model-based tracking", *BMVC*, pp.310-318, 1991

[201] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body", *IEEE Trans. on PAMI*, Vol.19(7), pp.780-785, 1997

[202] Y. Yacoob and L. Davis, "Learned temporal models of image motion", *ICCV-98*, pp.446-453

[203] Y. S. Yao and R. Chellappa, "Tracking a dynamic set of feature points", *IEEE Trans. on Image Processing*, Vol.4(10), pp.1382-1395, 1995

[204] A. Yuille and P. Hallinan. "Deformable templates", In *A. Blake and A. Yuille, editors, Active Vision*, pp.20-38, The MIT Press, 1992

[205] Z. Zhang, "Token tracking in a cluttered scene", *Image and Vision Computing*, Vol.12, No.2, pp.110-120, 1994

[206] L. Zhao and C. Thorpe, "Qualitative and quantitative car tracking from a range image sequence", *CVPR-98*, pp.496-501

[207] Q. Zheng and R. Chellappa, "Automatic feature point extraction and tracking in image sequences for arbitrary camera motion", *IJCV*, Vol.15, pp.31-76, 1995