Single and Multiple Target Tracking via Hybrid Mean Shift/Particle Filter Algorithms

Asad Naeem

Supervised by Dr Tony P. Pridmore

Thesis submitted to the University of Nottingham

For the degree of Doctor of Philosophy

March 2010

Abstract

This thesis is concerned with single and multiple target visual tracking algorithms and their application in the real world. While they are both powerful and general, one of the main challenges of tracking using particle filter-based algorithms is to manage the particle spread. Too wide a spread leads to dispersal of particles onto clutter, but limited spread may lead to difficulty when fast-moving objects and/or high-speed camera motion throw trackers away from their target(s). This thesis addresses the particle spread management problem. Three novel tracking algorithms are presented, each of which combines particle filtering and Kernel Mean Shift methods to produce more robust and accurate tracking.

The first single target tracking algorithm, the Structured Octal Kernel Filter (SOK), combines Mean Shift (Comaniciu et al 2003) and Condensation (Isard and Blake 1998a). The spread of the particle set is handled by structurally placing the particles around the object, using eight particles arranged to cover the maximum area. Mean Shift is then applied to each particle to seek the global maxima. In effect, SOK uses intelligent switching between Mean Shift and particle filtering based on a confidence level. Though effective, it requires a threshold to be set and performs a somewhat inflexible search.

The second single target tracking algorithm, the Kernel Annealed Mean Shift tracker (KAMS), uses an annealed particle filter (Deutscher et al 2000), but introduces a Mean Shift step to control particle spread. As a result, higher accuracy and robustness are achieved using fewer particles and annealing levels. Finally, KAMS is extended to create a multi-object tracking algorithm (MKAMS) by introducing an interaction filter to handle object collisions and occlusions.

All three algorithms are compared experimentally with existing single/multiple object tracking algorithms. The evaluation procedure compares competing algorithms' robustness, accuracy and computational cost using both numerical measures and a novel application of McNemar's statistic. Results are presented on a wide variety of artificial and real image sequences.

Published Work

The following research papers were published during the course of the work reported here.

Conference Papers

1. Naeem, A., Mills, S. and Pridmore, T,. Structured Combination of Particle Filter and Kernel Mean-Shift Tracking. 21st International Conference Image and Vision Computing, IVC New Zealand. (2006)

2. Pound, M., Naeem, A., French A., and Pridmore, T. "Quantitative and Qualitative Evaluation of Visual Tracking Algorithms using Statistical Tests", IEEE ICCV Performance Evaluation of Tracking and Surveillance (PETS) , Rio de Janeiro, Brazil October (2007)

3. Evans, D. and Naeem, A. "Using Visual Tracking to Link Text and Gesture in Studies of Natural Discourse", In: 3rd International Conference on e-Social Science, October 2007, Univ. of Michigan Ann Arbor, USA: ESRC/NSF, (2007). Also available in: online proceedings of Exploring Avenues to Cross-Disciplinary Research, University of Nottingham, UK: CDRG. http://www.nottingham.ac.uk/cdrg/, November (2007)

4. Naeem, A., Pridmore, T and Mills, S. ,. 'Managing Particle Spread via Hybrid Particle Filter/Kernel Mean Shift Tracking'. British Machine Vision Conference BMVC, page 970-979 (2007)

5. French, A., Naeem, A., Dryden, I. and Pridmore, T., "Using social effects to guide tracking in complex scenes". IEEE International Conference on Advanced Video and Signal based Surveillance, (AVSS 2007) page 212-217.

6. Brundell, P., Knight, D., Tennent, P., Naeem, A., Adolphs, S., Ainsworth, S., Carter, R., Clarke, D., Crabtree, A., Greenhalgh, C., O'Malley, C., Pridmore, T. and Rodden, T. "The Experience of using Digital Replay System for social science research" , NCeSS 2008, Manchester, 19th June (2008).

7. Tomlinson, M.J., Pooley, K., Simpson, T., Newton, T., Hopkisson, J.F., Jayaprakasan, K., Jayaprakasan, R., Naeem, A. and Pridmore, T., "Validation of a novel computer assisted sperm analysis (CASA) system employing multi-target tracking algorithms". Poster presented at Human Fertility, Edinburgh 7-9 January (2009)

Journal Papers

1. Naeem, A., and Pridmore, T , "Kernel Annealed Mean Shift Tracking" , Journal of Pattern Recognition (Submitted and under review)

2. Tomlinson, M.J. Naeem, A., Pooley K.,, Simpson, T., Newton, T. Hopkisson, J., Kannamanadias Jayaprakasan, K., Jayaprakasan R., and Pridmore T.,, "Validation of a novel computer assisted sperm analysis (CASA) system employing multi-target tracking algorithms.", Journal of Fertility and Sterility 2009, Accepted and awaiting publication.

Acknowledgements

I would like to dedicate this thesis to my parents Naeem Sarwar and Shagufta Naeem and my wife Umber, it was their conviction, persuasion and sheer belief in me that kept me motivated for this hazy journey.

I would specially like to thank my supervisor Dr. Tony Pridmore who's encouraging, enthusiastic and able guidance made this time a lot of fun and one of the best revealing experience of my academic life, working with him was a privilege indeed. Thanks to everyone who helped in the way.

Special thanks to my two beautiful daughters Rameen and Aleena, and my wife whose patience, serenity and affection are second to none. This one is for you all.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Visual Tracking

The goal of visual tracking is to recover from a time-ordered sequence of images a description of the dynamic behaviour of some target object or objects. Visual tracking is a major research area within computer vision. Though analysis of individual, static images is valuable in many application areas, the world we live in is naturally dynamic: time-varying image sequences are the norm.

Visual tracking problems take a variety of forms. Image sequences might be captured by a static camera, but show a dynamic world. Though the background is fixed, objects will move across the image plane, possibly changing their 3D orientations to present different views to the camera. Some (e.g. humans) might change shape as they move, so that object velocity and configuration are combined in the image data. Alternatively, the camera might move through a static world, acquiring data from different viewpoints and, unless illumination is constant across the viewed scene, different lighting conditions. Highly likely in the real world, but less commonly considered in computer vision research, the camera might move through a dynamic environment, gathering images which combine changes in both viewpoint and object location. Each of these scenarios presents different challenges, but each requires some form of object tracking if the resulting image data is to be interpreted.

Current and potential practical applications of visual tracking algorithms are countless and growing. Object tracking is required and has been attempted in many domains, including surveillance (Suyu et al 2008, Chen et al 2007), sports analysis (Perš and Kovačič 2000), gesture recognition (Tennant R. 1998), medical applications such as microscopic sample analysis (Ratnalingam et al 2006), to study traffic and pedestrian flow dynamics for efficient designs of roads and pathways (Suyu et al 2007, Chen et al 2007), growth patterns in plants and animal cells (French et al 2008), tracking and targeting applications and studying group behaviour in moving animals and humans (French et al 2007).

Algorithms that aim to track single objects through image sequences have existed for more than two decades, and have been successful in some circumstances. The general tracking problem is, however, far from solved. No current tracker is perfect; all may safely be assumed to fail at some point due to the complexity of natural image sequences and the wide range of problems (movement noise, background clutter, target occlusion, illumination changes, etc) that can affect their performance. A tracker is generally considered to have failed if it becomes disassociated with its targets so that its motion no longer reflects theirs. Further difficulties arise when multiple targets must be tracked simultaneously. Targets can collide with and/or occlude each other, adding movement noise and/or reducing the information available for use during tracking. If they appear similar, targets may attract each others' trackers, distracting them from their true targets - it is common for multiple independent trackers tracking objects with resembling appearance models, to "coalesce" on a single target (Khan et al 2004).

Visual tracking algorithms combine models of the appearance and motion of the object(s) they are required to track, using them to predict the future position (and other properties like speed and direction of motion) of the target(s) through a sequence of images (video). One way to improve tracking performance is to tune the models used to the task at hand. Many of the motion and appearance models employed in the literature are created for very specific tasks, and only work well under specific sets of circumstances, e.g. tracking a human being based on a stick figure model (Yilmaz et al 2006).

The overarching goal of the research reported here is to understand and develop single- and multi-target visual tracking algorithms which can be applied to a wide variety of tracking problems and situations, making them a good choice to employ in practical applications and products. As a result, generic motion and appearance models are used throughout. Instead of seeking to improve tracking performance through improved motion or appearance models, the work reported here considers the tracking engines used to apply those models. In particular, attention is focused on hybrid tracking algorithms obtained by combining the well-established particle filter and Kernel Mean Shift methods.

1.2 Particle Filtering, Kernel Mean Shift and the Particle Management Problem

Particle filtering is one of the most widely-adopted approaches to visual tracking. The defining characteristic of the particle filter is its use of a set of discrete particles to represent multi-modal probability distributions that capture and maintain multiple hypotheses about target properties. Particle filtering is iterative. Particles are repeatedly selected, projected forwards using a motion model, dispersed by an additive random component, and evaluated against the image data. Many particle filter trackers have appeared since Blake and Isard (Isard and Blake 1998a) first introduced Condensation.

The ability of a set of particles to represent a wide variety of distributions is both the main strength and primary weakness of the particle filter. For effective tracking in real-world environments the particle set must sample widely enough to represent all reasonable alternatives in areas of ambiguity. It must not, however, become diffused, spreading across the image plane rather than clustering around the object of interest. When this happens particles tend to migrate towards local maxima in their evaluation function, becoming caught on clutter and losing track of the target. Similarly, particles should not become too focused. Though it is encouraging to see a particle set coalesce when a single, clearly distinguishable target moves across the image, the tracker should not however become irreversibly locked onto a single mode.

A key issue in the design of particle filter-based trackers is how to manage the spread of the particle set to balance these conflicting requirements. Some researchers seek to maintain a wider distribution, focusing on the problems caused as particles cluster, sometimes very quickly, around one target hypothesis. Other workers consider standard algorithms to spread the particle set too thinly across the image and concentrate effort on forcing particles to coalesce, reducing the number needed. The variance of the posterior is simply and elegantly maintained by the Kalman filter (Kalman 1960), but particle filters cannot assume a Gaussian, or indeed any specific, distribution. Moreover, balance must be achieved with as few particles as reasonably possible. Increasing the particle set improves representational accuracy, but adds significantly to computational overhead.

Recently, Maggio and Cavallaro (Maggio and Cavallaro 2005) introduced the idea of embedding a Kernel Mean Shift tracker (Chang and Ansari 2005) within a particle filter algorithm. Kernel Mean Shift hill climbs towards the target, minimizing the distance between target and appearance model descriptions. A spatial kernel provides some robustness to noise and partial occlusion, and the algorithm provides fast and effective tracking as long as the target object does not move further than its own diameter between frames. The role of the Kernel Mean Shift in Maggio and Cavallaro's hybrid tracker is to move particles towards local maxima of the evaluation function on each iteration of Condensation. Though the authors focus on the computational savings made, Maggio's (Maggio and Cavallaro 2005) hybrid tracker can be viewed as attempting to manage particle spread by alternately diffusing the particle set using Condensation and clustering them with Kernel Mean Shift.

Blake and Deutscher (Deutscher et al 2000) attempt to control the particle set spread using a multi stage annealed particle filter. Each stage consists of the probability density function with a smoothing filter applied to it. The first stage is the smoothest, ironing out the small local maxima, then the second stage is a more irregular one as smoothing effect is reduced and so on. Particles are added with random noise at each smoothing stage, and then the particles with highest weight are picked more often, this tends to guide the particles towards the global maxima, achieving an effect somewhat similar to the Kernel Mean Shift algorithm (Chang and Ansari 2005) as particles tend to move towards the global maxima after each smoothing stage.

1.3 Aims, Objectives and Achievements

The research reported here builds upon the idea of hybrid particle filter/Mean Shift tracking, examining ways in which particle filtering and Kernel Mean Shift might be combined to produce single and multi-target tracking algorithms in which the particle set is well-managed and tracking consequently more robust. Annealing is a key ingredient of two of the novel algorithms developed.

This thesis describes the development of and presents three novel and general hybrid object tracking algorithms. Two are single target trackers, while the last is a multi-target tracker.

Recognizing the strength of the Kernel Mean Shift algorithm, early work considered making Kernel Mean Shift the dominant technology. This lead to the development of the Structured Octal Kernel (SOK) filter (Chapter 3). In the SOK algorithm, a small number of particles are generated, in a structured fashion, to explore further when confidence in Kernel Mean Shift becomes low. Though effective, SOK's search is somewhat crude, and it requires the user to specify both the conditions under which extra particles are spawned and the size of the region to be searched.

To avoid these drawbacks an alternative approach was adopted in the second algorithm, which is termed the Kernel Annealed Mean Shift (KAMS) tracker. Here, rather than shift control away from the particle filter component and towards the Kernel Mean Shift tracker, Condensation is replaced with a more powerful particle filter. KAMS (Chapter 4) is created by combination of the Kernel Mean Shift algorithm with the Annealed Particle filter of Deutscher (Deutscher et al 2000). The hypothesis underlying this decision is that by flattening local maxima in the evaluation function the annealed particle filter will allow a greater spread in the particle set, and reduce the need for a possibly erroneous predictive motion model, while the Mean-Shift component will continue to successfully pull particles back towards the true target. The resulting tracker should, therefore, be more robust than both its component algorithms and previous particle filter/mean-shift hybrids.

Finally, KAMS is extended to multi-target tracking by addition of a multi-stage interaction filter (Chapter 5), building on that proposed by Khan et al (2004). While Khan's original algorithm (Khan et al 2004) simply reduces confidence in hypotheses which fall in close proximity to each other, the use of annealed particle filtering and Mean Shift in the multi-target KAMS (MKAMS) algorithm allows the tracker to place hypotheses at maxima of a smoothed evaluation function. Though target interactions may prevent the algorithm from locating the unsmoothed maxima, it does place hypotheses at well defined locations.

All three algorithms are tested and evaluated on both real world and artificially created, simulated data, against existing algorithms. Evaluation techniques are in an early stage of development, but are increasing in importance The evaluation procedure adopted here compares competing algorithms' robustness, accuracy and computational cost using both numerical measures and a novel application of McNemar's statistic.

During the course of this research, eight conference papers have been published in reputed conferences including BMVC 2007, IVCNZ 2006, IEEE AVSS 2007 and IEEE ICCV PETS 2007. One journal paper was published in the Journal of Fertility and Sterility 2009, and another is under review by Pattern Recognition.

1.4 Thesis Structure

The remainder of the thesis is structured as follows

Chapter 2: Background

This presents an introduction to tracking and an overview of existing tracking algorithms, focusing attention on the methods exploited and combined to form the novel tracking methods described here. Evaluation methods are also considered, and an evaluation technique defined which will be employed throughout this thesis to gauge tracking algorithms against each other.

Chapter 3: Structured Octal Kernel Mean Shift Tracker (SOK)

The first novel single target algorithm is presented and explained. It is evaluated against existing techniques including Maggio's previous hybrid tracker, and results are shown which indicate that SOK has advantages over the other algorithms considered.

Chapter 4: Single Target Kernel Annealed Mean Shift Tracker (KAMS)

The KAMS algorithm, developed for single target tracking, is presented in this chapter. Results of the evaluation process are also presented and discussed. KAMS is shown to be a significant improvement on SOK.

Chapter 5: Multi Target Kernel Annealed Mean Shift Tracker (MKAMS)

The extension of KAMS to multi-target tracking is discussed. Two possible multi-target KAMS algorithms are presented in this chapter along with evaluation results.

Chapter 6: Conclusions and Future Work

This final chapter reviews the advances made during this project, and proposes directions for future research.

All implementations of all the algorithms used for evaluations throughout the thesis are my own.

Chapter 2: Background

2.1 Introduction

Computer Vision is the science that aims to let machines see. It is concerned with the theory, tools and techniques using which we can build systems that can extract useful, quantitative information from images of some aspect of the real world. The image data input to computer vision systems varies widely. Single grey level or colour images are often considered. These might show natural or man-made 3D physical environments (landscapes, offices, streets, etc), they might be portraits of individual or groups of people, medical images acquired using a variety of specialist imaging devices (e.g. MRI, CT), scanned documents, microscope images, etc. Sequences of images attract increasing amounts of attention. These might be live feeds obtained directly from cameras, carefully produced movies, CCTV footage showing everyday activities in inhabited environments, amateur or professionally produced records of sports events, etc. Whatever the details of the available data and task at hand, the goal of Computer Vision is to extract information implicit in the image(s) provided.

Image data provides many cues which can be exploited by computer vision methods. Sharp changes in intensity, colour, or other image properties often mark the boundaries of objects and surfaces (Canny 1986, Pathegama and Gol 2004), allowing the image to be segmented into meaningful regions. Colour, shape, and other features of those regions can allow specific objects or materials to be identified (Ohlander et al 1978, Shi et al 1997). Patterns of shading in a grey-level image can provide information about the 3D orientation of the surface being viewed and/or local illumination conditions (Salih et al 2004). Multiple views of a given object can allow

its 3D position and shape to be recovered (Park 2005), while time-based image sequences provide descriptions of the motion and deformation of objects (Evans et al 2007).

Computer vision has many actual and potential applications. For instance it is used in industrial processes to control production assemblies; many industrial concerns use vision sensors as they perform different operations on production lines. Fault detection and quality control by visual inspection one of the most common uses of computer vision in industrial environments (Wallace A. M. 1988). A good survey of industrial applications of cognitive vision can be found in (Courtney and Böttcher 2003). Other application areas include, but are not limited to:

- Handwriting recognition and more general document image analysis; methods are being developed to extract text, diagrams and drawings from images of paper documents for input to a range of software systems.
- Medical image analysis; medical applications use a wide variety of image sources and types including X-ray and MRI/CT scans, ultrasound analyses etc (Ratnalingam et al 2006) the emphasis being on segmentation and the analysis of shape.
- Interaction between computers and humans; this still relies heavily on special devices such as mice and keyboards, but vision-based interaction devices are now being introduced (e.g. Dornaika and Ahlberg 2004, Green et al 2005).Vision for interaction is a rapidly growing area of research.
- Biometrics: fingerprint and face recognition have received the most attention in this area, though other modalities such as gait (Nixon and Carter 2004) also generate interest.
- Human gesture and event recognition for communication: e.g. American Sign language recognition.
- Automatic surveillance; given recent societal events and the spread of CCTV hardware, automatic surveillance is currently a major growth area for vision research (Suyu et al 2008, Chen et al 2007) Key issues include tracking and event recognition.

- Object modeling for multimedia and entertainment: here, 3D scanners use stereo vision to create 3D models of the object(s) of interest (e.g. Hahn and Duncan 2006).
- Navigation of robots and vehicles through known and unknown terrain: vision sensors are being developed and used in both cars and robots. The MARS Rover, for example, uses a binocular stereo system to navigate on the alien terrain of the planet (Biesiadecki et al 2001).

Object recognition is a key challenge for computer vision and a very active research area. Here, after acquiring the image, vision algorithms try to find and identify different objects using 2D or 3D appearance models (Berg et al 2005, Lowe 2001). Object recognition may be used to identify and locate objects of interest in an individual image or image sequence. Object recognition, along with other computer vision methods, has been used to extract the parameters needed to organize information, e.g. when indexing databases of images and image sequences., Recognition may also be the first step in a larger process, such as object classification, quality assurance and inspection in industrial scenarios, etc. or visual tracking.

Visual tracking requires the objects of interest to be identified and some description of their current state (e.g. location, pose, velocity) to be extracted from each frame of a time-ordered image sequence. Though recent advances in object recognition (Viola and Jones 2003) have suggested that this might be achieved by considering each image independently of the others, successful visual tracking requires access to several (at least two) images at a time.

Section 2.2 discusses the motion analysis and tracking of objects of interest in image sequences, along with the main problems faced during tracking. Several ways to represent objects' appearance are also discussed. Section 2.3 describes the major single target algorithms and section 2.4 describes existing multi-target tracking algorithms. Some of these are created by combining the single target trackers mentioned in section 2.3, while others are based on explicit representations of the joint states of several targets. Section 2.5 describes some applications of tracking in

various areas and domains. Finally, section 2.6 discusses evaluation techniques and presents the methods that have been employed throughout this thesis to gauge algorithms against each other.

2.2 Visual Motion and Object Tracking

Computers have been used for decades to record image sequence data. Videos were stored on disks, tapes and other storage media, to be analyzed manually for intelligent inference. Applications include CCTV surveillance, medical sample analysis (e.g. of blood and semen under a microscope), sports team game analysis, human gesture analysis, traffic monitoring etc. For many years computers could record and store data from these domains in the form of videos, but could not extract information from this data.

The need for automated video analysis has motivated much work in computer vision. In this field artificial intelligence techniques have been employed to analyse videos and infer or deduce facts. Motion analysis is one of the most important parts of computer vision as it deals with the extraction and analysis of object movement and behaviour, a major source of information about the viewed world. This information is valuable in a wide variety of situations, whether the image sequences concerned are available as pre-recorded video files or provided in real time via a camera system.

Approaches to the extraction of motion information from image sequences can be broadly divided into two classes. The first of these assumes that the camera has a high frame rate relative to the motion in the scene. Successive images are then captured from very similar positions and/or depict very similar situations. The key assumption is that successive images are similar enough that gradients of image data can be estimated with respect to time, as well as the spatial dimensions of the image. These gradients can then be used to estimate the motion of pixels and other features across the image plane. This approach to the computation of a dense optic flow field was first introduced by Horn and Schunk (Horn and Schunk 1981), but many variations on the theme have been described. A good review of the field can be found in (Barron et

al 1994). Optic flow methods typically produce a motion estimate at each pixel, but can be computationally expensive and often rely on restrictive assumptions about the images and the motion they depict. The second approach to motion analysis assumes that the differences between successive images are too great to allow reliable estimation of gradient values. Instead, features of interest are extracted from each image independently and matched across frames. The coordinates of corresponding features then describe their motion across the image.

Work in visual tracking assumes the input images vary enough to require some form of matching process. Trackers, however, do not attempt to match a comprehensive set of image features between frames. Instead, they focus on a small set of target objects; many track only a single target.

An image sequence or video consists of consecutive sequences of images, often called frames. Objects moving through a scene appear at different positions in a scene and produce a trajectory. Simply defined, object tracking is the procedure which estimates the state of an object in the image plane as it moves around a scene. An object's state may include descriptions of any of its properties, though in most tracking work, including that reported here, the primary focus is on the position of the target(s) projection onto the image plane.

Automated video/image sequence analysis involves four major steps. The first is the representation of the object(s) of interest, the second is the detection of those objects, the third step is to track detected objects by matching between frames, and the final stage involves analyzing the trajectories and other information obtained to deduce facts and recognise occurrences of events. For example, while performing gesture recognition on sequences showing a human moving his/her head and hands, the developer has to first define how to represent a person. That representation must describe both the appearance of the target – image features and properties that are associated with its presence - and its likely motion. The system must then use the appearance model to detect the head and hands of the person in the scene, use both the appearance and motion models to perform tracking and exploit the tracked output trajectories to support identification of events such as head nods, hand gestures etc. (Evans et al 2007, Knight et al 2008)

2.2.1 Problems in Object Tracking

Object tracking is a challenging task. The performance of a given tracking method in a particular situation can be influenced by a large number of factors. Some of these are properties of the target object(s), some are properties of the surrounding environment, some are properties of the image data, and others are properties of the tracker.

Fast and/or erratic target motion

High speed targets increase the area of the image in which a target might appear. If its direction of movement is unknown, or poorly approximated, a target located at point P in frame N of a sequence can reasonably be expected to lie near the arc of a circle centred on P in frame N+1. The faster the target is moving, the greater the radius, and so the greater the circumference of that circle (or length of that arc). Erratic movement, incorporating large and unpredictable changes in velocity, increases the search area still further. Normal cameras with low shutter speeds may not be a good choice for fast or erratically moving objects as they may appear blurry, so to track them properly cameras with high shutter speeds are required.

Clutter

The presence of clutter – other, unrelated objects similar to the target - may play a major role in tracker failure. Suppose a tracking algorithm uses edge detection to find objects in a scene, in the presence of clutter additional, spurious edge data, will be generated. This will increase the computational complexity of the matching/tracking task and may make it hard to distinguish the true target from the distracting background.

Occlusion

Partial and complete occlusions are one of the biggest obstacles to object tracking. During occlusion obstacles come between the camera and the tracked object, hiding the object partially or completely behind them. The obstacles may be other tracked objects or surrounding structures of no interest. Many tracking algorithms have been designed with occlusion in mind (e.g. French et al 2007), but the problem remains

unsolved. As objects become occluded their appearance changes in ways that are unlikely to be captured by the tracker's appearance model; tracking then fails.

Illumination changes

Tracking applications may be affected by variations in either local or global illumination conditions. While tracking, if the lighting conditions change due to the target passing through shadows or because there is an overall change in illumination source(s), the object's appearance model may no longer reflect its actual appearance. If the appearance model is not sufficiently general, or the tracking algorithm cannot update the model used, the tracker may fail.

Image noise

Successful tracking relies on achieving a good match between models (of appearance and motion) and image data. Excessive image noise can make that matching difficult, unreliable, or even impossible.

Appearance Models

To track objects we have to represent them. If the assumptions made by the appearance model are incorrect or inaccurate, the tracker may fail. For example if we represent a car as a simple polyhedron, and the tracking algorithm uses edge detection to find an appropriate polygon and declares it a car, if the car turns and the camera now looks at a deformed shape, the appearance model that the algorithm is looking for may not be present any more.

Motion Models

Motion models describe the likely movement of the target between successive frames and are used to define the area in which a given target is expected to appear in the next image. Algorithms like the Kalman Filter (Kalman 1960) model motion using linear equations, and are effective when that assumption is true. However, if the motion model does not accurately capture the target's movement the tracking system may fail, losing the target due to incorrect assumptions.

Hardware limitations

Even if accurate motion and appearance models are available, hardware limitations may prevent successful tracking. For instance if objects move too fast, and camera

shutter speeds are not adequate, targets may be blurred and maybe vanish for a frame or two. When this happens tracking will most probably be lost. Similar effects can occur if the camera used is not of sufficient spatial or radiometric resolution, or introduces too much image noise.

2.2.2 Appearance Models

In order to track an object, it must be well-defined. An appearance model must be built or acquired that represents the object. The tracking algorithm will then match this model to future frames. Objects can be represented using their shapes, colour distributions or sometimes a combination of both. Object representation techniques in common use are discussed below.

In Figure 2.1a, the object is represented by a point (Veenman et al 2001), and by a group of prominent points (Serby et al 2004) as in figure 2.1b. Usually the objects represented using points are very small in size, or appear very small in the image plane, like food particles in a fish tank, or very distant flying birds and aircraft. Objects detected in consecutive frames are represented by points and association of points is based on the objects' states, which may include motion and position. Tracking using point representations becomes difficult in the presence of noise, occlusion by other objects and when objects enter and exit the scene. Usually, if a point representation model is employed, then rigid rules are defined for point association and motion.

Objects can also be represented using primitive shapes like squares, ellipses and rectangles as shown in figure 2.1c/2.1d (Comaniciu et al 2003). This representation is more suitable for rigid objects which are not expected to change shape drastically, as when tracking a ball, cars viewed from above etc. Tracking is typically performed by matching some description of the expected shape to features (e.g. edges) extracted from each image in the sequence.

Some objects (e.g. hands, the human body) have more complex structures which keep changing in shape as they move. Of course this cannot be captured by a simple primitive geometric shape like a rectangle or ellipse. Contour and silhouette representations (figure 2.1g, 2.1h) are employed to track such objects. The contour is the outer boundary of the tracked object and the region inside the contour is called the silhouette (figure 2.1i). These representations are suitable for tracking complex non-rigid shapes (Yilmaz et al. 2004). Algorithms employing these representations maintain an appearance model of the object from previous frames and look for a corresponding object in the next frame, this model may be a colour histogram of pixels of the silhouette area, an edge map or the object contour.

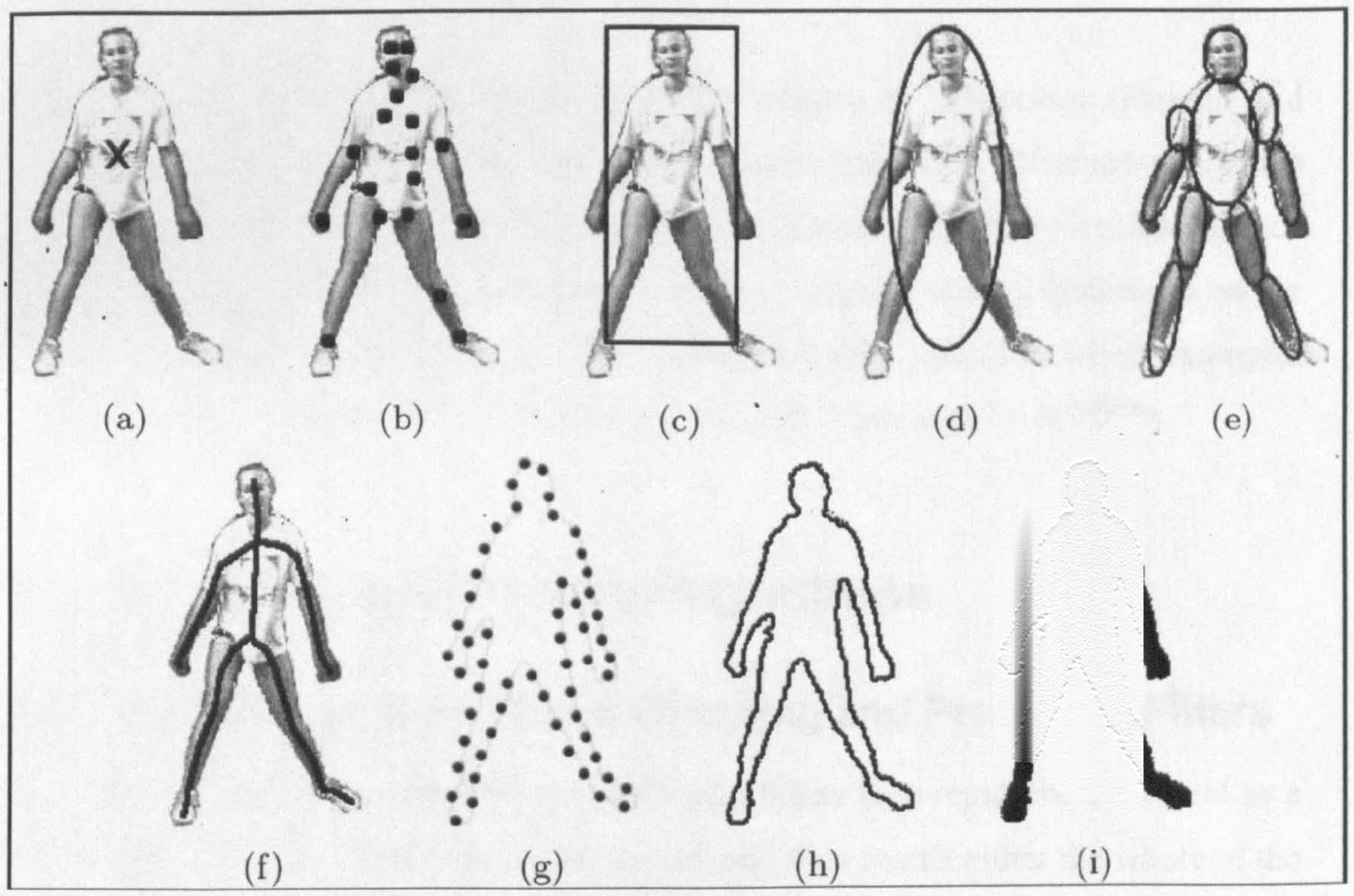


Figure 2.1: Object representations (© Yilmaz 2006)

Articulated shape models are composed of sub-parts held together by joints as shown in figure 2.1e. Relationships between the various parts are governed by kinematic motion models which predict and constrain joint angles. Cylinders and ellipses are often used to represent the major components of such models.

In Figure 2.1f a skeletal model represents the person. This alternative approach to representing articulated shapes is commonly used for recognising objects (Ali and Aggarwal 2001). This model is extracted by applying an axis to the object silhouette (Ballard and Brown 1982).

Similarly, appearance models based on colour histograms have been used to represent the objects in video sequences. One of the advantages of this type of representation is that it is more robust to shape changes, but it may be prone to apparent colour changes caused by varying illumination. This may be handled by using colour spaces that are less prone to illumination changes, for example RGB space is more sensitive to illumination changes than HSI space.

Templates are formed using simple geometric shapes or silhouettes (Fieguth and Terzopoulos 1997). They carry both spatial and appearance information. Active appearance models are generated by simultaneously modelling shape and appearance (Edwards et al. 1998). In general the object shape is represented by landmarks on the contour or inside it in the silhouette. They require a training phase in which examples of similar objects and dissimilar objects are provided (Viola and Jones 2003).

2.3 Single Target Tracking Algorithms

2.3.1 Data Association, Block Matching and Predictive Filters

A crude, but potentially effective, way to track objects is to represent the object as a tight block of pixels which contains the object, and then search either the whole of the next image, or a predefined area around the previous position, for that pixel block. The search consists of scanning through the image trying to find a similar size block with the same characteristics as the tracked object. This technique is called block matching. Block matching is a form of data association (Bar-Shalom and Fortmann 1988, Bar-Shalom and Li 1993, Grzegorz et al 2007). Image features expected to arise from the tracked object are extracted from each image independently and rules defined to associate the objects between frames. For instance one rule may be to accept the candidate in the next frame that lies closest to the target's old position in the previous frame. The data association approach however requires a distinction to

be made between the background and moving objects. This can be achieved by finding out which areas belong to moving targets using techniques like background subtraction.

Perhaps the most widely adopted approach to visual tracking, however, is to view it as a predictive filter; a cyclic process in which the target's state in image N+1 is predicted from its estimated state in image N, and that prediction is then used to initialize a localised search for the target. A local search window is usually employed, in which we search for the target's next position. If one target is found in the search area then the problem is trivial, but if multiple targets are found then we need some heuristic rules to identify the targets of interest. They may be minimum distance measures, appearance models like contour shapes or colour representations etc. This crude method will only succeed if the objects are small in number and almost never interact. The search window must also contain the targets of interest. If the targets move outside the search window, the approach fails.

The success of a predictive tracker relies on the effective combination of a motion model, which determines where any search should commence, and a search area of appropriate size and shape. An accurate motion model greatly eases the tracking problem by reducing the size of the region that must be searched. However, when the motion model is not a good fit to the actual motion of the target, or noise introduces errors into estimates of target state, use of a small search area may lead to the target being missed. This can be compensated for by increasing the size of the search area to allow for prediction errors, but any increase in search area is accompanied by an increased risk that the tracker will become attracted to background clutter that forms a local maximum in its evaluation function The Kalman filter (Kalman 1960) resolves this dilemma well.

2.3.2 Kalman Filter

One of the earliest, but still most widely used predictive tracking algorithms is the Kalman Filter. The Kalman Filter (Kalman 1960) consists of an efficient computational (recursive) solution of the least squares method (Welch and Bishop

2001), and addresses the general problem of trying to estimate the state of a discrete time controlled process that is governed by a linear stochastic difference equation. The Kalman filter is quite efficient in several aspects; it supports estimation of past, present and future states, and it can do so even when the precise nature of the modelled system is unknown. The Kalman filter assumes that the posterior density at every time step is Gaussian, and hence parameterized by a mean and covariance.

The Kalman filter estimates a process by using a form of feedback control; the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback, i.e. for incorporating a new measurement into the a priori estimate to obtain an improved posteriori estimate. The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations. Indeed the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems.

One of the drawbacks of the Kalman filter is that it may fail or perform badly when the estimated problem's state cannot be modelled as a linear stochastic difference equation. This issue was addressed with the Extended Kalman Filter (Julier and Uhlmann 1997). A Kalman filter that linearizes about the current mean and covariance is referred to as an extended Kalman filter or EKF. In the extended Kalman filter the state distribution is approximated using a Gaussian random variable (GRV), which is then propagated analytically through the first order linearization of the non linear system. This can introduce large errors in the true posterior mean and covariance of the transformed GRV, this may lead to failure of the tracking in progress.

The Unscented Kalman Filter (Wan and Merwe 2001) addresses this problem, by using a deterministic sampling approach, where the state is approximated by a Gaussian random variable, but it is now represented using a minimal set of carefully

chosen sample points which in theory completely capture the mean and covariance accurately.

The Kalman filter was the de facto standard filtering algorithm for some time, despite its limitations. Advancements in particle filter-based tracking algorithms capable of supporting non-linear models and multiple hypotheses have, however, diminished the use of Kalman filters in recent years.

2.3.3 Particle Filters

While Kalman filtering has proved very effective as a visual tracking engine, it has two limitations. First, the original Kalman filter requires object motion to be linear. Though the EKF can handle non-linear motion models it is not guaranteed to converge. Second, and perhaps most importantly, the Kalman filter represents its hypothesis of target state as a single Gaussian. As a result, it can only maintain a single hypothesis. In many situations the data input to a tracker is ambiguous, and a successful algorithm must be able to maintain multiple hypotheses – i.e. a multi- not uni-modal probability density. Particle filters were created to support multiple hypotheses and work well in most situations.

Particle filters are sequential Monte Carlo methods based on point mass of particles representations of probability densities, and they can be applied to any state space model. Sequential importance sampling (SIS) algorithms are Monte Carlo methods that form the basis for most sequential Monte Carlo filters (Doucet et al 2001).

The sequential Monte Carlo approach is known by many names, including bootstrap filtering (Gordon et al 1993), interacting particle approximations (Crisanand et al 1999), the Condensation algorithm (Isard and Blake 1998a), particle filtering (Carpenter et al 1999) and survival of the fittest (Kanazawa et al 1995). The key idea here is to represent the posterior density function (pdf) by a set of random samples with associated weight, and to obtain estimates based on these samples and their weights. Particle filtering is an iterative process and uses a set of discrete particles to represent multi-modal probability distributions that capture and maintain multiple hypotheses about target properties. Particles are repeatedly selected, projected

forwards using a motion model, dispersed by noise component, and evaluated against the image data.

The basic particle filter works by spreading *n* number of particles around the tracked object, whose appearance model is known beforehand. This is effectively an object detection/recognition step. The weights of the particles are computed by matching individual particles' state with the appearance model and trying to assess the likelihood of that particle representing the true state (usually location) of the tracked object. In the next frame the object moves, and a new set of *n* particles is generated from the old one, in such a way that the particles with the higher weights in the previous frame are picked more often than the ones with a lower weight. To cater for the unpredictable nature of real world objects in motion, random noise is deliberately introduced to the particles' states in addition to motion characteristic data. This helps the filter to handle situations in which the object changes its path suddenly, not following any pre-defined motion model. After introducing noise into the particles we re-weight all n particles and find out where the objects has moved to by either selecting the highest weighted particle or, more commonly, taking the weighted average of the particles' states, which gives us the position of the object in the current frame. This phenomenon is shown in Figure 2.2.

Since particles are placed randomly, many particles may be needed to cover a given state space. In the presence of background clutter that looks like the object, the particles may scatter widely across the image because particles falling on cluttered areas may have higher weights than the true target and so will be selected more often. This may result in the failure of the tracker.

A standard problem in statistical pattern recognition is finding an object parameterized as x with prior $p(x)$, using data z from a single image. The posterior density given by $p(x|z)$ represents all the information about x that can be obtained from the data z. Posterior density can be obtained by applying Bayes' rule (Papoulis, A. 1990)

$$p(x|z) = kp(z|x)p(x) \tag{2.1}$$

Where k is a normalization constant that is independent of x. When $p(z|x)$ becomes complex $p(x|z)$ cannot be evaluated in simple closed form, so iterative sampling technique like factored sampling is used (Grenander et al 1991). This technique generates a random variate of x from a distribution $\overline{P}(x)$ that approximates the posterior $p(x|z)$. A sample set $\{s^1, s^2, ..., s^N\}$ is first generated from the prior density $p(x)$ and then an index of $i \in \{1,..., N\}$ is chosen with probability π_i where

$$\pi_i = \frac{p_z(s^{(i)})}{\sum_{j=1}^{N} p_z(s^{(j)})} \tag{2.2}$$

and

$$p_z(x) = p(z|x) \tag{2.3}$$

the conditional observation density. The value $x' = x_i$ has a distribution which approximates the posterior $p(x|z)$ increasing accurately as N increases (Figure 2.2).

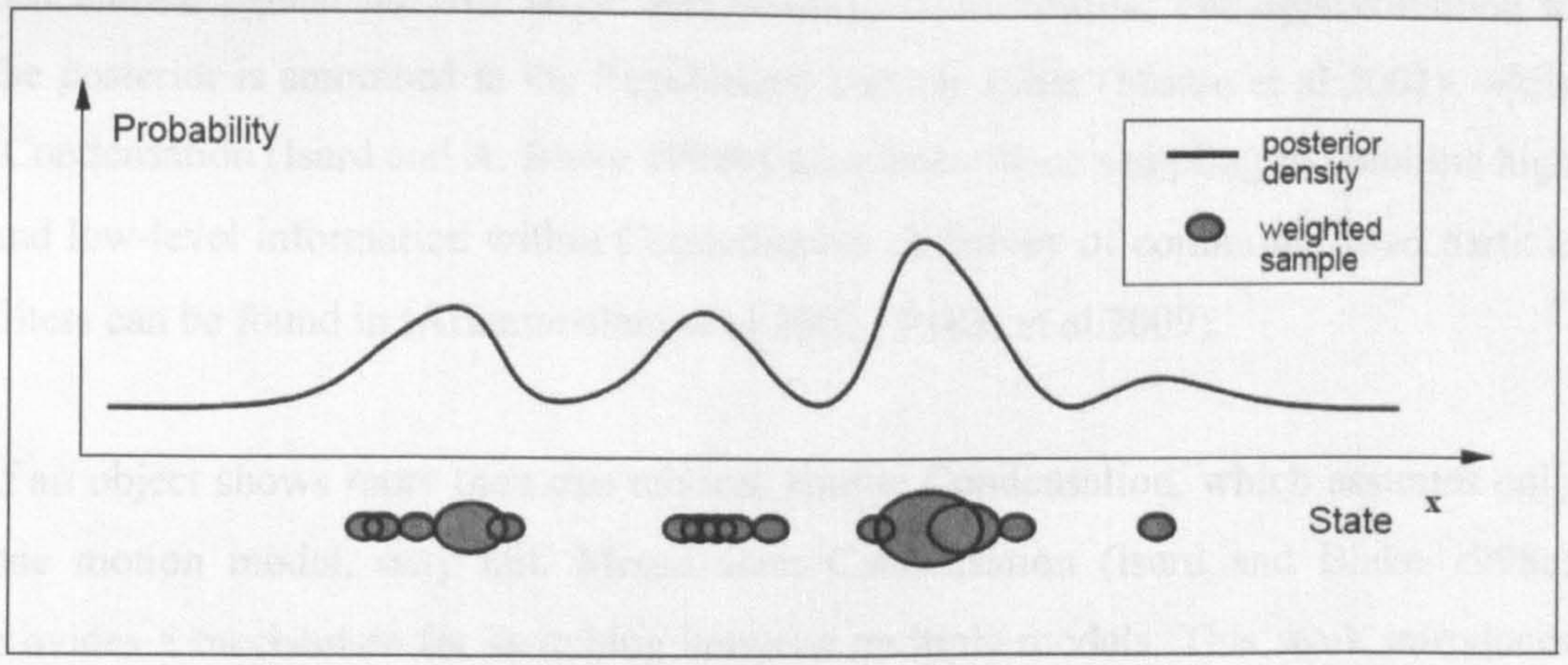


Figure 2.2: Factored Sampling. A set of points s^i, the centres of the blobs in the figure, is sampled randomly from the prior density $p(x)$. Each sample is assigned a weight π_i proportional to the value of the observation density $p(z|x) = s^i$, this is also shown by the blob areas, as higher weight is shown as a larger blob. These weighted points serve as a representation of the posterior density $p(x|z)$.

The process at each time step is a self-contained iteration of factored sampling, the output of the iteration will be a weighted time stamped sample set $\{s_t^n, n = 1,....N\}$

with weights π_t^n, this weight represents the conditional state-density $p(x_t \mid z_t)$ at time t. Clearly the process will begin with a prior density, the prior for time (t) will be (t-1) and prior density will be given by $p(x_{t-1} \mid z_{t-1})$. This would have been derived from a prior sample set represented by $\{s_{t-1}^n, \pi_{t-1}^n, n = 1,.....N\}$.

Now when a new sample set is chosen from the one at time t-1, the probability of selection is proportional to π_{t-1}^n which represents the weights of particle set at time t-1. So particles with high weights in the previous time step are picked more often. This particle set is then disrupted by additive random noise and reweighed using the data z at time t, giving the new distribution $p(z_t \mid x_t)$.

Many particle filter-based trackers have been developed since Blake and Isard first introduced the Condensation algorithm. The Auxiliary Particle Filter (Pitt and Shephard 1999) selects particles in a more intelligent manner, making them concentrate around the true target and yielding better results. The approximation to the posterior is smoothed in the Regularized Particle Filter (Musso et al 2001), while ICondensation (Isard and A. Blake 1998b) uses importance sampling to combine high and low-level information within Condensation. A survey of commonly used particle filters can be found in (Arulampalam et al 2002 , Ristic et al 2009).

If an object shows more than one motion, simple Condensation, which assumes only one motion model, may fail. Mixed state Condensation (Isard and Blake 1998c) provides a mechanism for switching between multiple models. This work introduces an additional state variable that specifies which motion model should be used for tracking at a given instant. A matrix of model-state transition probabilities is provided, and used to process the discrete state label y forward in time. Using this model transition between states occurs automatically, as each state transition with non-zero probability contributes samples to the distribution. So the particle distribution is represented by all available motion models. As one model predicts the target position more accurately, it begins to dominate future predictions. This was successfully shown to track a bouncing ball with multiple motion models (Isard and Blake 1998c); the tracker switched motion models as the ball bounced from a racquet,

and at the top of its flight when it started to come down towards the racquet under the influence of gravity.

For effective tracking in real-world environments the particle set must sample widely enough to represent all reasonable alternatives in areas of ambiguity. It must not, however, become diffused, spreading across the image plane rather than clustering around the object of interest. When this happens particles tend to migrate towards local maxima in their evaluation function, becoming caught on clutter and losing track of the target. Similarly, particles should not become too focused: the tracker should not become irreversibly locked onto a single mode.

A key issue in the design of particle filter-based trackers is how to manage the spread of the particle set to balance these conflicting requirements. The variance of the posterior is simply and elegantly maintained by the Kalman filter, but particle filters cannot assume a Gaussian, or indeed any specific, distribution. Moreover, balance must be achieved with as few particles as reasonably possible. Increasing the particle set improves representational accuracy, but adds significantly to computational overhead.

Several works have addressed aspects of this problem. Some point out that, in practice, the advantages of the particle filter approach are often lost as particles cluster, sometimes very quickly, around one target hypothesis. They focus on maintaining a wider distribution. The Annealed Particle Filter (Deutscher et al 2000) uses annealing to smooth out the evaluation function, making the global maximum clearer and allowing particles to be spread further, by increasing the process noise, without becoming caught on local clutter. Vermaak et al (Vermaak et al 2003) explicitly model the particle distribution as a Gaussian mixture model, forcing the resulting filter to sample an appropriate number of particles from each model component. This prevents a single, slightly more highly weighted, mode from dominating the particle distribution. A similar approach is taken in (Milstein et al 2002), particles are clustered and each cluster tracked individually.

Other workers consider standard algorithms to spread the particle set too thinly across the image and concentrate effort on forcing particles to coalesce, reducing the number

needed and so computational expense. The Kernel Particle Filter (Chang and Ansari 2005) applies a Mean Shift operation to the particle set to pull the centre of the particle distribution towards the target centre. This is effective, but clusters weighted particles without further reference to the image data, taking no account of the actual shape of the evaluation function between the locations sampled by the particle set.

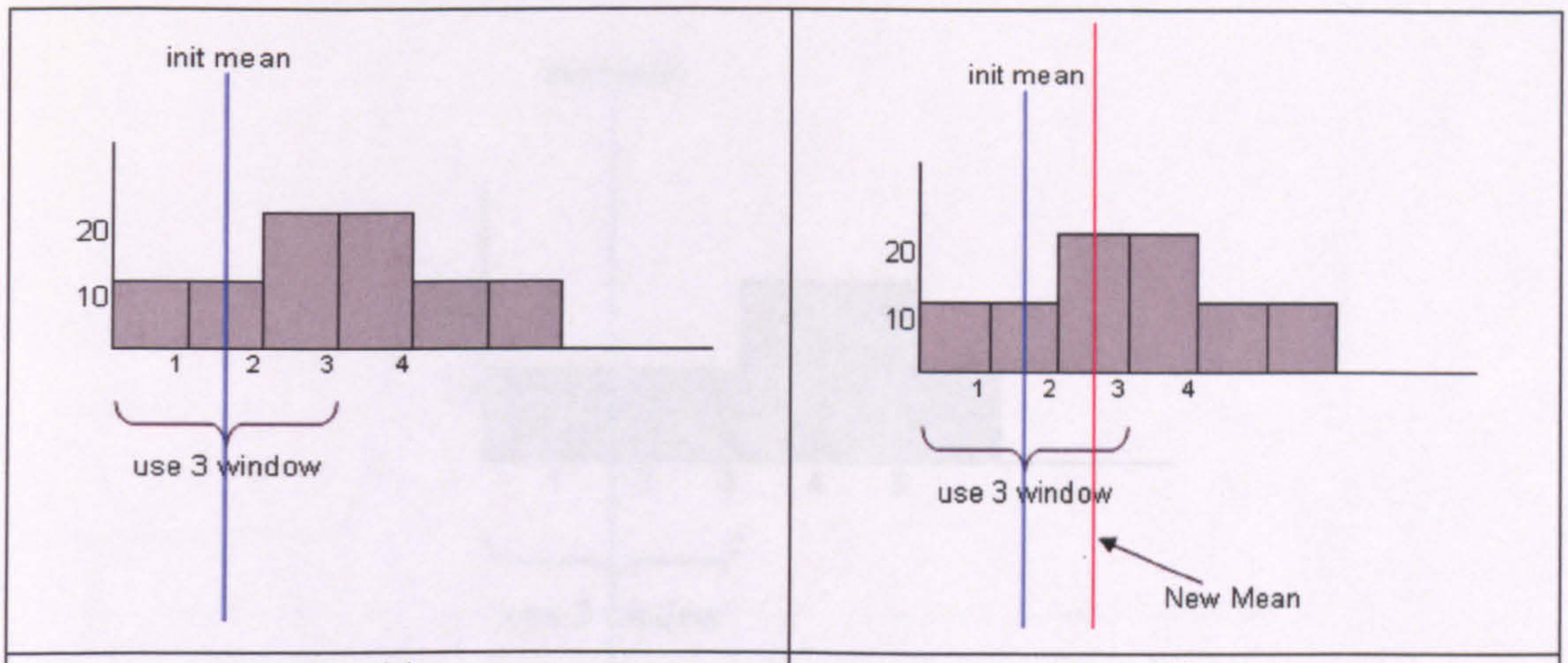
The work reported in this thesis focuses on the problem of managing particle spread by creating hybrid trackers which combine particle filter methods with variational techniques, specifically the Kernel Mean Shift algorithm. The following section reviews Kernel Mean Shift tracking, before previous hybrids are considered in Section 2.3.5.

2.3.4 Kernel Mean Shift

The Kernel Mean Shift algorithm (Comaniciu et al 2003) is an effective and fast algorithm with which to track objects. Here, the target is represented by a feature distribution regularized by a spatial mask with an isotropic kernel. This masking induces spatially smooth similarity functions suitable for gradient based optimization. The Bhattacharya coefficient is used as the similarity measure and the Mean Shift algorithm is used to locate the optimal position. The Kernel Mean Shift algorithm hill climbs towards the target, minimizing the distance between the target and the model descriptions.

The algorithm constructs a probability density function (pdf) using a histogram of the tracked area, and stores it as the model of the object being tracked. It then computes the likelihood of each pixel being the part of the object on the search grid which is a bigger area around the previous position of the object. After that the algorithm moves the target location to the local maxima.

Consider the following $1-D$ example where x is the position on x-axis, $p(x)$ is the probability or likelihood of that pixel on y-axis.



(a) (b)

Figure 2.3 One-dimensional Mean Shift: an example.

Figures 2.3(a) and 2.3(b) show how Mean Shift moves towards the higher weighted areas. Consider the high valued area as the current position of the object that moved recently from its old position shown by the blue line. In this figure we have used a 1D case for simplicity, and use a sliding window of 3 units. In figure 2.3(a) we have 6 blocks showing the weights, the blue line is the current centre, as the high weight area is between block 3 and block 4, we hope to shift the centre towards them. So assuming the 3 block window, we simply plug in the weights of block 1, 2 and 3 (10, 10 and 20 respectively) in equation 2.4.

$$\sum x_i p(x_i) / \sum p(x_i) = (1*10 + 2*10 + 3*20)/40 = 2.5 \qquad (2.4)$$

where x_i is the block number and $P(x_i)$ is the weight of each block. The 2.5 is the new centre shown by the red line in figure 2.3(b), and it is closer to the true centre which is in this case between block 3 and 4. This process is repeated until the tracker reaches the highest weighted position.

If the window does not contain the high weighting region, as shown in figure 2.4, Mean Shift becomes ineffective. If the 3 blocks making up the sliding window lie over positions 1, 2 and 3

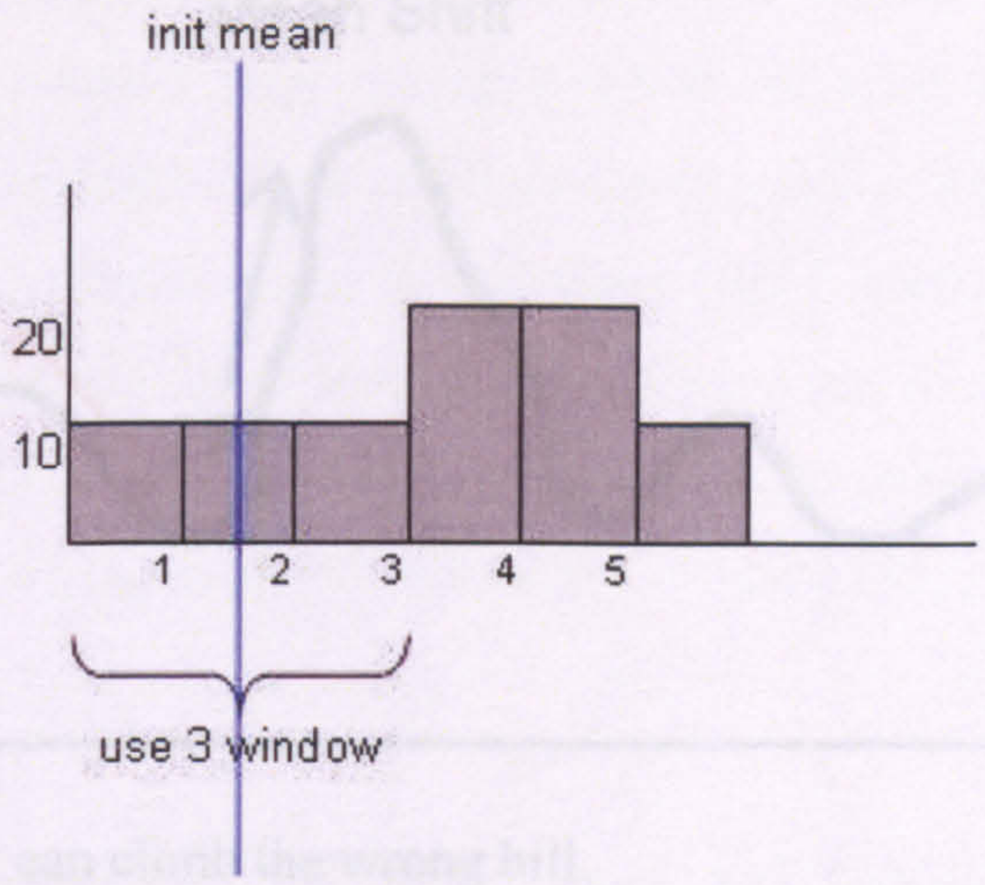


Figure 2.4: Example in which Mean Shift fails due to incorrect window size or fast movement of the object.

$$\sum x_i p(x_i) / \sum p(x_i) = (1*10 + 2*10 + 3*10)/40 = 1.5 \qquad (2.5)$$

The centre has not shifted due to the fact that no high weighted region was within the window. Hence, if the window is not sufficiently big, or the object jumps beyond the scope of the window, tracking fails.

Kernel Mean Shift is effective, but simple. The spatial kernel provides some robustness to noise and partial occlusion, and the algorithm provides efficient and effective tracking of larger, slower moving targets. If, however, the target moves by more than its own diameter between frames, there is little chance that the hill climbing procedure will seek out the correct peak. Even if the tracker is in the vicinity of the target (i.e. the global maximum), the algorithm may climb the wrong hill and latch on to clutter if it lies on the slope of a local maximum (Figure 2.5).

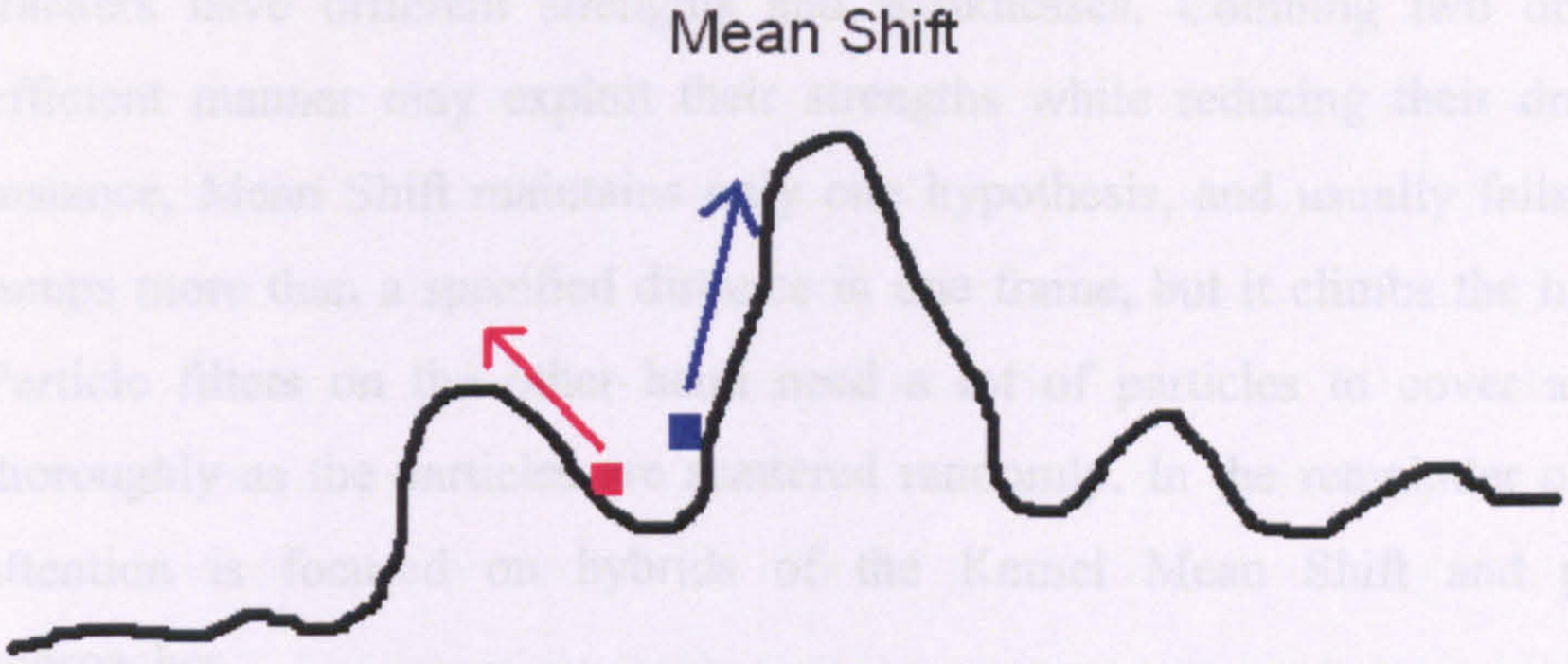


Figure 2.5: Mean Shift can climb the wrong hill.

Figure 2.5 shows two Mean Shift trackers trying to achieve the maximum peak by hill climbing. The top most peak represents the tracked object, and the lower peaks represent clutter that looks somewhat like the object. Now if the Mean Shift falls on a position as shown by the blue square then after hill climbing toward maximum will result in successful tracking, but if it was slightly towards the left it now falls on the slope of the wrong hill belonging to clutter, and now it will climb the wrong hill. Since Kernel Mean Shift only maintains one hypothesis of object's position, it may fail.

A number of variations on this theme have been described; a variety of colour models and similarity measures (Yang et al 2005) have been used and arbitrary spatial weighting (Leung and Gong 2006) has been incorporated to represent objects with arbitrary or changing shapes. Collins (Collins 2003) has extended the approach to track blobs through scale space, alternating Mean Shift tracking in the spatial and scale axes. Zhao and Nevatia (Zhao and Nevatia 2004) employ a Mean Shift with an additional term which requires the target to be different from the local background, while Porikli and Tuzel (Porikli and Tuzel 2005) use a set of kernels of varying sizes to capture a wider range of target motion.

2.3.5 Hybrid Tracking Algorithms

A hybrid filter is formed when two or more existing tracking algorithms are combined to achieve a hopefully superior tracking algorithm. As discussed earlier, different

trackers have different strengths and weaknesses. Combing two or more in an efficient manner may exploit their strengths while reducing their drawbacks. For instance, Mean Shift maintains only one hypothesis, and usually fails when object jumps more than a specified distance in one frame, but it climbs the hill efficiently. Particle filters on the other hand need a lot of particles to cover a given space thoroughly as the particles are scattered randomly. In the remainder of this chapter attention is focused on hybrids of the Kernel Mean Shift and particle filter approaches.

Maggio and Cavallaro's hybrid tracker (Maggio and Cavallaro 2005) combines Condensation with Mean Shift tracking to provide a system in which particles are alternately diffused by Condensation and clustered towards the local maxima by performing Mean Shift on each particle. Multiple hypotheses are maintained by projecting a number of particles randomly around the prior position, and then these particles hill climb towards the best target centre. This results in a smaller number of particles being required to carry out tracking successfully.

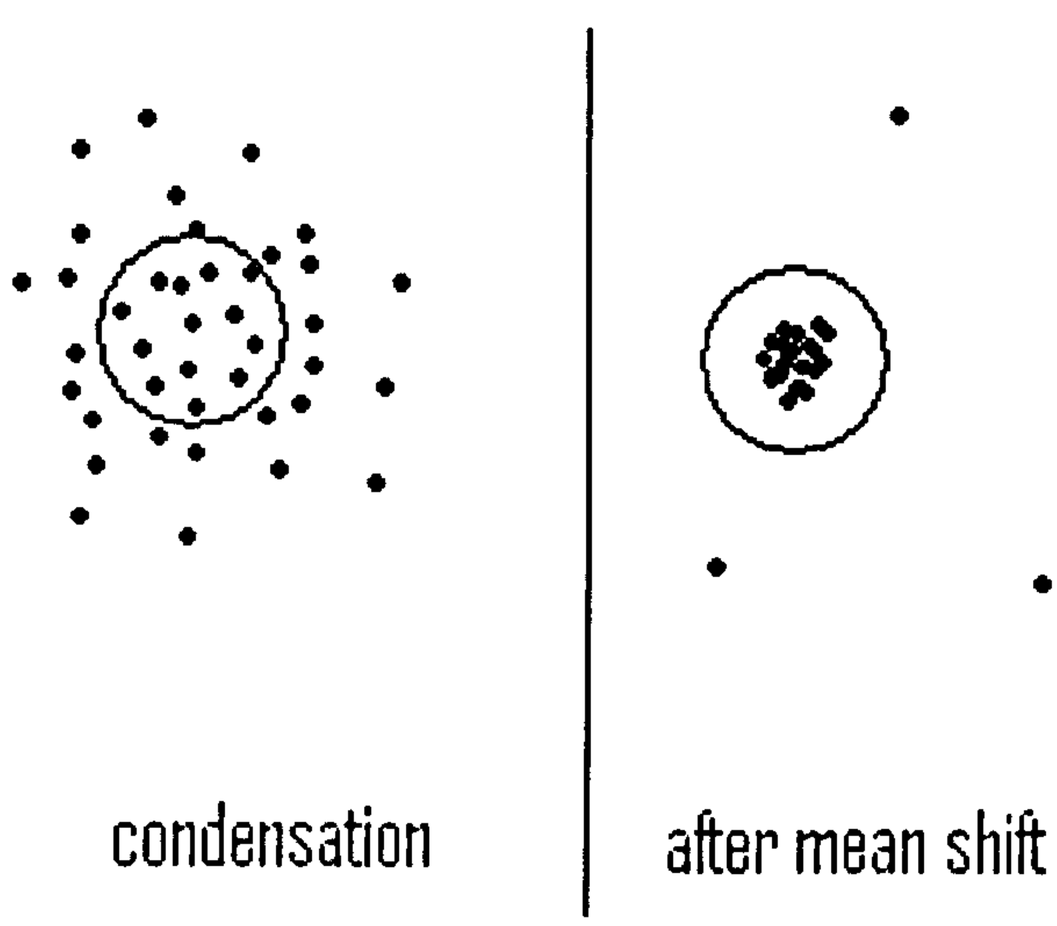


Figure 2.6: Hybrid Filter stages for each frame.

Figure 2.6 illustrates the outlines the operation of the hybrid filter, in the first stage the particles are distributed using a predictive motion model and a small random component, and then each particle is drawn towards the global maximum by application of Mean Shift. This as we can see gives us an accurate measure of the objects centre while maintaining multiple hypotheses.

The hybrid tracker shows performance advantages over both Condensation and Mean Shift tracking, but also has some drawbacks. As the particles are randomly projected we still need a good number to cover a given search space. Running N Mean Shift trackers, where N is the number of particles in the system, also makes the system computationally expensive. Furthermore, many of the particles coalesce during the Mean Shift phase, moving to the same hypothesis and making the representation redundant. If Condensation tends towards an incorrect local maximum, mean-shift will accelerate the process.

Shan (Shan et al 2007) combine the two approaches in essentially the same way as Maggio and Cavallaro (Maggio and Cavallaro 2005), but argue that following Mean Shift the particle set samples a different distribution, in which the influence of the motion model is reduced. Shan et al view the Mean Shifted particles as arising from an importance function, rather than the temporal prior, and combine Mean Shifted and unprocessed particles together as in (Isard and Blake 1998b). Wang (Wang et al 2007) use CamShift (Bradski 1998) to move particles towards modes in the evaluation function, taking advantage of CamShift's ability to adaptively change the scale of the target sought.

Chapter 3 of this thesis proposes a hybrid tracking algorithm (SOK) in which Kernel Mean Shift is the dominant technology. A small number of particles are generated, in a structured fashion, to explore further when confidence in Kernel Mean Shift becomes low. Following analysis of this initial algorithm another single target tracker, Kernel Annealed Mean Shift (KAMS) is developed. KAMS combines Annealed Particle Filtering (Deutscher et al 2000) with Kernel Mean Shift, achieving performance advantages over Condensation, Kernel Mean Shift, annealed particle filtering and Maggio and Cavallaro's hybrid algorithm. KAMS is described and discussed in chapter 4 of this thesis.

2.4 Multiple Target Tracking Algorithms

The previous section considered single target tracking algorithms. Many situations, however, require multiple objects to be tracked. Examples include applications in surveillance, medical sample analysis, and traffic applications (Suyu et al 2007, Chen et al 2007).

In single target tracking the key problems are cluttering and occlusion of the target by surrounding objects, and the effects of noise in the motion and appearance estimates made, When multiple targets must be tracked concurrently, the situation becomes more complex. Simply applying a tracker, independently, to each target is problematic. Targets may pass close to, or even occlude, each other. The problem is greater if the targets appear similar. As each tracker employs a model of its target, those tracking similar targets will use similar, possibly identical models. In the presence of noise, it is almost inevitable that one target will be a closer fit to those models than the rest. Over time, as targets pass close to and/or collide with each other, trackers will tend to migrate onto that object. A set of N trackers associated with each of N targets will quickly coalesce into a set of N trackers all tracking one target, with N-1 targets being ignored.

Multiple target tracking algorithms can be placed in two major categories. In the first, a single target tracker is attached to each object, and some intelligent interaction handling mechanism is introduced to mediate between them. This is there to handle possible occlusions and collisions. Other multi-target trackers may maintain a joint state in which every hypothesis contains information about all the targets. Multi-target tracking algorithms are the topic of the reminder of this section.

2.4.1 Multiple Hypothesis Tracker

If Kalman filtering is used to track many objects that may appear alike, then to the tracker responsible for any one object, others will appear and act like clutter. Multiple 'correct' measurements (i.e. many targets) will exist and hence form a potentially non-Gaussian probability distribution function. A Kalman filter cannot represent this situation effectively. The Kalman filter has however been used as a component in

multi-object tracking algorithms like the Multiple Hypothesis Tracker (Reid 1979). The MHT algorithm maintains several correspondence hypotheses for each object at each time frame. The algorithm has the ability to create new tracks for objects entering the observed area and terminate tracks for objects exiting the field of view. It can also handle occlusions, that is, continuation of a track even if some of the measurements from an object are missing. In the MHT probabilities are employed to assign measurements to tracked objects and then a Kalman filter is used to derive the state estimation. This method has a number of drawbacks. It expects an inflow of new targets into the surveillance region, and can in fact initiate new target tracking from one measurement. This is problematic for any scenes with background clutter, or for any scenarios where the number of targets is fixed. The MHT algorithm is also computationally exponential both in memory and time.

2.4.2 The Joint Probabilistic Data Association Filter

The Joint Probabilistic Data Association Filter or JPDAF (Bar-Shalom and Fortmann 1988) attempts to eliminate some of the problems of the Multiple Hypothesis Tracker. The number of tracked targets is fixed as JPDAF does not expect new objects to enter or existing objects to leave the scene. This algorithm handles the association of an arbitrary number of measurements at a given time to an arbitrary number of established targets, i.e. no new targets are accounted for. However, the algorithm itself has drawbacks, including its inability to handle occlusions well. As image likelihoods are evaluated independently, tracking can break down when targets become close to one another or overlap, and no mechanism is given to overcome this problem.

2.4.3 Mixed state tracker

Particle filters and other Monte Carlo methods are generally poor when the posterior is multi modal as a result of the ambiguity caused by the presence of multiple objects. Though particle filters can in principle represent multi-modal distributions, in practice the particles tend to cluster very quickly around a single (the most likely) hypothesis. The Mixed State Tracker (Vermaak et al 2003) addresses this problem by modeling the target distribution as a non-parametric mixture model. Vermaak and Doucet show that Monte Carlo implementation of a general recursive tracker leads to a mixture of

particle filters that interact only in the computation of the mixture weights, leading to an effective tracking algorithm. This algorithm maintains posterior multi-modality for each object. Usually the number of modes is limited to a threshold and a number of particles are assigned to each mode. For example, if 5 modes are computed and maintained for the object appearance, and 20 particles are assigned to each mode, then for each object we have 100 particles in total. The crucial design issues in mixture particle filters are the choice of the proposal distribution and the treatment of objects leaving and entering the scene.

2.4.4 Boosted particle filtering

A Boosted particle filter (Okuma et al 2004) which extends the approach of Mixed state tracker (Vermaak et al 2003) uses a cascaded Adaboost algorithm (Viola and Jones 2001) to train and learn models of the tracked objects, in this case hockey players. These detection models are used to guide the particle filter. The proposal distribution consists of a probabilistic mixture model that incorporates information from Adaboost and the dynamic models of the individual players. This enables them to quickly detect and track players in a dynamically changing background.

2.4.5 Markov Chain Monte Carlo (MCMC) Algorithm

Some of the most promising work on multi-target tracking was carried out by Khan et al (2003, 2004). The basic hypothesis here is that objects in close proximity influence each other's behaviour. In their first paper (Khan et al 2003), a Markov random field (MRF) motion model is used to model the interactions between ants which were being tracked in a confined area. This similar target tracking problem is one of the most difficult to handle since all the targets have exactly the same appearance, no matter how they are represented. The tracking was very effective in their experiments, but, since the joint state space of all targets is required, the particle filter suffers from exponential complexity in the number of targets. Their second paper (Khan et al 2004) replaces the traditional sampling step of particle filters with a Markov chain Monte Carlo (MCMC) step. This allows a more efficient representation of the joint space and along with the MRF interaction function produces good quality tracking of multiple targets.

One of the problems with single target trackers while tracking similar objects is that individual tracker has no information about other objects, and if the appearance model of one of them appears to be the best in a frame then all trackers coalesce on the best target representation. Khan's MCMC algorithm represents a joint state and is one of the most effective multi object trackers. Once objects start to interact or trackers try to coalesce on an object tracked by another tracker, the MRF-based interaction mechanism tries to prevent it. As well as being evaluated by a measurement from the image, the particles are also affected by the value of the interaction term as in the following equation

$$P(Z_{it} \mid X_{it}) \prod_{j \in E_i} \psi(X_{it}, X_{jt}) \qquad 2.6$$

where $P(Z_{it} \mid X_{it})$ is the image measurement and $\prod_{j \in E_i} \psi(X_{it}, X_{jt})$ is the interaction metric. E_i is the set of Markov random field graph edges connected to the target i (i.e. target with which an interaction can occur), ψ is the interaction function which takes the form of a Markov random field-based motion model, which produces a low probability score if targets are within a certain distance of each other. This prevents trackers from making very similar hypotheses, resisting the tendency of multiple trackers to latch on to the target with the best appearance match.

2.4.6 Motion Parameter Sharing (MPS)

Khan's MCMC algorithm is a very promising tracker though complete and partial occlusions may still cause some targets to be lost. Motion parameter sharing was developed by Andrew French (French et al 2007) to address this problem. MPS (French et al 2007) is similar to Khan's algorithm, but also uses the movement of objects in a group exhibiting similar motion characteristics to predict their future positions.

The assumption underlying motion parameter sharing is that if it could be identified that some targets are exhibiting the same motion characteristics, e.g. speed and direction, then they would probably continue to move in the same way in the future. Even if an object moving in a group is occluded for some time, its state can be

estimated by considering the motion characteristics of visible members of the group it is believed to be moving in.

In French's MPS, motion estimates are maintained for each object over a number of frames, this includes both speed and direction. Objects moving in groups are identified and their motion predicted using motion estimates associated with a randomly selected member of the group. Of course, after each frame the object's positions and group dynamics also changed as its motion history is also updated. So objects may leave and enter a group based on the motion they exhibit. Spearman correlation was used to determine if the motion characteristics of speed and angle of objects correlated enough to consider them members of the same group. French's MPS showed increased accuracy and robustness to situations when occlusions occur and results were published in the AVSS 2007 conference (French et al 2007).

2.5 Major Applications of Tracking

There are many applications of object tracking, it is actively used in automated surveillance applications for monitoring and security (Kerhet et al 2007), (French et al 2007). Many applications analyzing gestures (Starner and Pentland 1995) and events including human behaviour (Nickel and Stiefelhagen 2007) use tracking. It has been used for the past decade in medical applications like microscopic sample analysis and joint movement studies, sports analysis (Perš and Kovačič 2000) may also use object tracking to study team strategies and moves. Many human computer interfaces may use visual tracking and many companies are now developing HCI (human computer interaction) devices based on tracking. These range from web cameras for video conferencing to human input devices used in the gaming industry. Automated traffic flow and monitoring systems also employ these techniques. Many sensors and devices for vehicles are being produced and developed whose purpose ranges from pedestrian safety to vehicle navigation.

2.6 Performance Evaluation of Tracking Algorithms

As the number and variety of tracking algorithms grows, it becomes increasingly important that suitable performance evaluation and comparison techniques be available. Evaluation of visual tracking algorithms is a complex task requiring

consideration of the robustness, accuracy and computational cost of any proposed method. Estimation of accuracy requires some measure to be made of how well tracker output reflects the true path of the target, while robustness measures how well the tracker performs in a large number of different scenarios; the more scenarios and problems it can handle the more robust it is. A tracker is generally considered to have failed when it becomes dissociated from its target, and failure is usually detected by eye. Artificially generated image sequences may be used, providing a high level of control over the test data available. Artificial data, however, can only approximate real world tracking problems and so cannot support a complete assessment. Real image sequences must be included in any comprehensive evaluation protocol.

2.6.1 Previous Work

An increasing amount of work is being carried out in the field of performance evaluation of object tracking algorithms. Ellis (Ellis 2002) investigated major requirements for efficient and effective performance analysis for surveillance systems and proposed some methods for characterizing video datasets. Needham (Needham and Boyle 2003) proposed a set of metrics and statistics for comparing trajectories and evaluating tracking motion systems. Brown (Brown et al 2005) suggest a motion tracking evaluation framework that estimates the number of true positive, false positive and false negative, merged and split trajectories.

Yin (Yin et al 2007) proposes a set of metrics that compare the output of motion tracking systems to a ground truth in order to evaluate performance. They present a set of statistical metrics to assess different aspects of performance of motion tracking. The proposed statistical metrics, such as track matching error, closeness of tracks and track completeness, indicate the accuracy of position estimates, the spatial and temporal extent of the objects respectively and are closely related to the motion segmentation module of the tracker. Metrics such as correct detection track, false alarm track and track detection failure provide a general overview of algorithm performance. Track fragmentation shows the temporal coherence of tracks. ID Change is useful to test the data association module of multi target trackers.

The data used during evaluation must be carefully chosen. Several projects, typified by the i-LIDS programme in the UK (i-LIDS 2007), have sought to provide standard

test sets. I-LIDS have developed an image library to help academics and system manufacturers evaluate video analytics systems to meet Government requirements. i-LIDS currently consists of a video library of CCTV footage based on five different scenarios: abandoned baggage, parked vehicle, doorway surveillance, sterile zone and multiple camera tracking. Though it is a valuable resource for those developing tracking algorithms, i-LIDS is designed with higher level operations in mind.

The remainder of section 2.6 describes the evaluation methods used to assess the visual tracking algorithms developed and reported here. To provide a thorough comparative evaluation of the different hybrid techniques proposed, and their component algorithms, estimates are made of computational cost (section 2.6.2), the accuracy of the target state descriptions produced (section 2.6.3) and robustness (sections 2.6.4 and 2.6.5). Robustness is measured in two ways. Numerical estimates of the sensitivity (proportion of targets correctly labelled) of the various algorithms provide fine-grained measurements (section 2.6.4), but the absolute difference in sensitivity between two algorithms can be hard to interpret. In contrast, McNemar's statistic (section 2.6.5) provides a principled, quantitative test of the relative robustness of tracking algorithms, reporting, with an associated confidence value, which of two algorithms has produced the best performance over a given test set.

2.6.2 Evaluating Computational Cost

Execution times provide an estimate of the computational costs of tracking algorithms. Execution times per frame were computed and are used for comparison throughout the thesis. All algorithms were run on a standard personal computer fitted with an Intel Pentium quad core 2.4 GHz Q6600 processor, 2 GB RAM, 7200rpm SATA HDD and windows XP. Care was taken to ensure that no other applications or extra services were running while execution time measurements were made. The average time taken per frame for each algorithm was computed and is reported.

2.6.3 Evaluating Positional Accuracy

When evaluating the positional accuracy of the tracking algorithms developed here, the positional frame by frame error between tracked path and ground truth, and root mean square (RMS) techniques are adopted. As discussed earlier, RMS gives quite accurate results when the image sequences are artificial, but real image sequences

must also be tested. Both real and artificial sequences are used here The ground truth of artificial videos is provided by the generating software, and the ground truth of the real world videos is obtained manually by clicking on the objects frame by frame, pointing out their locations very carefully. The trackers' outputs were matched to the appropriate ground truth and frame by frame squared error plots were created.

2.6.4 Evaluation of Classification Accuracy

The positional error measures outlined in section 2.6.3 assess the accuracy of the target state descriptions produced by tracking. Visual tracking can also be thought of as a process of classification, in which the tracker labels some image location(s) as containing target(s), and others not. Robust tracking therefore results in a higher proportion of correct classifications. Classification accuracy is traditionally evaluated by considering measures of sensitivity and specificity.

In a recognition task, sensitivity is the proportion of occurrences of the target that are identified correctly, and specificity the proportion of non-occurrences of the target that are correctly identified. In a tracking scenario we define:

- **True Negative,** TN: The number of frames in which both ground truth and system results agree on the absence of any object, so no trackers are active and no objects are in view.
- **True Positive,** TP: The number of frames in which both ground truth and system results agree on the presence of object(s), i.e. the trackers are on those objects respectively.
- **False Negative,** FN: The number of frames in which ground truth contains some object(s), while the system sees no object(s) and hence no tracker is active.
- **False Positive,** FP: The number of frames in which there is a tracker initialized for some object(s), while ground truth either does not contain any object(s) or none of the ground truth's objects fall under any tracker.

Sensitivity and specificity are then defined as:

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.7}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2.8}$$

It is common when evaluating recognition systems to report both measures. The trackers reported here, however, do not include a target detection (i.e. recognition) step. All the trackers considered here are manually initialised at the beginning of each test run, so tracking begins with each tracker associated with a valid target. Target objects also never leave or enter the scene during the image sequences used. As a result, negatives are not reported by any of the algorithms being evaluated and TN and FN cannot be estimated. Only true positives (tracker still on its initial target) and false positives (tracker dissociated from its original target) can be counted. Specificity cannot, therefore, be computed directly. Sensitivity is calculated as the ratio between true positives and total number of frames, and so is available. The sensitivity of each algorithm implemented and compared is computed and reported throughout the thesis.

2.6.5 Using McNemar's Statistic to Evaluate Robustness

The robustness of a visual tracker is a measure of the extent to which it remains associated with the (correct) target throughout the test sequences. At present all trackers can reasonably be expected to lose their target at some point, the purpose of any performance evaluation scheme is to investigate the frequency with which and the conditions under which this occurs. A typical robustness test involves applying the algorithm to a set of image sequences, noting the points at which it fails and discussing why each failure, or a representative set of such failures occurred. Comparative analysis is achieved by pointing out situations in which tracker A lost the target while tracker B maintained its lock, and vice versa. Robustness analysis is therefore often qualitative, and largely subjective.

French (French et al 2007) compares the particle filter with his MPS object tracker and reports the number of objects successfully tracked till the end by both algorithms. Khan (Khan et al 2004), counts the number of failures, when the trackers go off the target by more than 50 pixels and reports them for each algorithm for a very long image sequence consisting of 10400 frames with 20 tracked objects.

Estimates of sensitivity, and specificity when appropriate, go some way towards addressing these problems. They provide clear numerical measures of the robustness of target detection/tracking algorithms. They are naturally applied to each frame of the test sequences used and so provide fine-grained descriptions of performance. Care must be taken, however, when choosing test sequences. A tracker that fails early in a long sequence will generate a much lower sensitivity measure than one which fails the same number of frames into a short sequence, despite both having failed once. Perhaps more importantly, though the sensitivity measures obtained from competing algorithms identify the algorithm with the highest sensitivity, the absolute difference between sensitivity values is hard to interpret. Is a given difference in sensitivity significant, or not?

To address these issues a further, complementary, evaluation technique is used. Statistical tests exist which can be applied to the results of tracking targets through a set of sequences to provide principled, quantitative statements of the relative robustness of tracking algorithms. McNemar's test is appropriate to this type of comparison (Clark et al 2008), and is applied here to evaluate robustness of competing pairs of algorithms at the image sequence, as opposed to frame, level

McNemar's statistic is a form of chi-square test for matched paired data. Consider the following 2 × 2 table of results for two algorithms (Table 2.1):

	Algorithm A Failed	Algorithm A Succeeded
Algorithm B Failed	N_{ff}	N_{sf}
Algorithm B Succeeded	N_{fs}	N_{ss}

Table 2.1: Table showing different scenarios for McNemar's test.

Let N_{xy} give the number of times algorithm A produced result x and algorithm B produced result y, and f and s denote failure and success respectively. McNemar's statistic is then:

$$X^2 = \frac{(|N_{sf} - N_{fs}| - 1)^2}{(N_{sf} + N_{fs})} \tag{2.9}$$

where the -1 is a continuity correction. If the number of tests is greater than about 30 then the central limit theorem applies. In such a case, the Z score (standard score) is obtained from (2.10) as:

$$Z = \frac{(|N_{sf} - N_{fs}| - 1)}{\sqrt{(N_{sf} + N_{fs})}} \tag{2.10}$$

If the two algorithms give similar results then Z will tend to zero (though z tends to infinity if the two algorithms perform exactly the same). As their results diverge, Z increases. Confidence limits can be associated with the Z value (Table 2.2).

Z Value	Degree of confidence (one-tailed prediction)
1.645	95%
1.960	97.5%
2.326	99%
2.576	99.5%

Table 2.2: Converting Z scores into confidence limits

To apply McNemar's to compare two algorithms tracking some target(s) through a set of videos, a definition of success and failure is required. We consider tracker A to have succeeded and tracker B to have failed if algorithm A maintains tracking for a greater proportion of a given image sequence, measured from the beginning of the sequence and using the same starting parameters. In effect we define success to be tracking as long as the more successful of the two trackers, and employ test sequences that are long enough and complex enough to force failure. In our applications of McNemar's to date (Chapter 3,4,5) loss of target is identified by eye, but clearly

defined as there being no association between any part of the target and any aspect of the tracker. Future robustness tests in this thesis will all refer to McNemar's test.

McNemar's statistic is computed over a representative, and ideally standard and publically available set of image sequences, requiring more than 30 sequences to produce a statistically reliable result. The image sequences used to evaluate the trackers developed here are described in Appendix A and Appendix B. Appendix A contains a single frame from test sequences showing a single moving target, Appendix B shows a single frame from videos showing multiple targets. Both include short descriptions of the videos. McNemar's test was employed for both single and multiple target tracking algorithms.

Detailed qualitative analysis of the causes of tracking failure will always be required to identify the strengths and weaknesses of tracking algorithms and to identify directions for future research. We believe, however, that statistical tests are valuable in the formal assessment of performance, and that McNemar's statistic, used as described here, is a valuable tool in the comparative analysis of tracking algorithms.

Chapter 3: Structured Combination of Particle Filter and Mean Shift Tracking

3.1 Introduction

Tracking algorithms may maintain a single or multiple hypotheses of possible object positions. Algorithms like Kalman filtering and Kernel Mean Shift maintain a single hypothesis which, although efficient computationally, may lead to tracker failure. If a tracker maintaining only a single hypothesis loses the object due to high-speed motion, occlusion, clutter, or for any other reason, it may fail to recover. If the only hypothesis maintained is incorrect there is no other option to turn to.

As discussed in earlier chapters, particle filters' main strength lies in the fact that they can represent multi-modal probability density functions that can capture and maintain multiple hypotheses. The particles in a particle filter are dispersed based on the current motion model, and random noise added to each particle. Even if the object suddenly changes its motion, some particles can reasonably be expected to fall on the new position of the object, provided the filter has sufficient particles to cover a decent area around the object.

One of the potential weaknesses of the original particle filter, however, is that if clutter is present around the tracked object which matches the appearance model of the target, then dispersed particles falling on local maxima (clutter) may obtain high weights from the evaluation function. While generating a new particle set from the old

one based on those weights, particles on clutter may be selected more often than those on the target. Since particle filters are iterative processes in which the particle set is repeatedly selected from the previous one based on their weights, the particle set may become diffused across the image after several iterations. A number of solutions to the problem have been proposed, as discussed in Chapter 2.

Recently, Maggio and Cavallaro (Maggio and Cavallaro 2005) used the Kernel Mean Shift tracking algorithm (Comaniciu et al 2003) to move particles towards local maxima on each iteration of Condensation (Isard and Blake 1998a). Kernel Mean Shift tracking is a hill climbing approach which first computes the likelihood of each pixel in a circular search space around the prior target centre being the next target centre, then moves the previous centre towards the maximum likelihood solution. The object model and candidate model both comprise probability density functions (pdfs) approximated by 2-D normalised histograms over the RGB colour space. The two dimensions are the ratios red/blue and green/blue. A kernel mask is used to give a higher weighting to pixels nearer the centre of the circular search region, making the algorithm more robust to target localisation errors and partial occlusions. The Bhattacharya distance is a measure of how close the tested appearance model is to the actual target model of the object, it lies between 1 and 0. A Bhattacharya distance close to zero means that the tested model is closer to the target model, while a distance closer to 1 means the similarity between the two is lower. Kernel Mean Shift tracking is an iterative process which continues until the Bhattacharya distance between the target pdf and the candidate pdf is either zero or a minimum value (Comaniciu et al 2003).

Kernel Mean Shift provides efficient and effective tracking as long as the target object does not leave the search area or move further than its own diameter between frames, this is due to the fact that the kernel mask used to cancel the effect of partial occlusions of nearby objects reduces the importance of outer pixels as compared to the centre pixels of the tracker area. If the target object jumps further than its diameter it will move beyond the kernel mask, and pixels beyond the kernel mask are ignored, so the Mean Shift tracker will fail.

Mean Shift is a competent tracker and in many situations can maintain tracking without the multiple hypotheses represented by the particle set. While valuable in areas of high ambiguity, in most cases the Condensation component of the hybrid tracker is an unnecessary overhead. These observations lead us to propose an alternative hybrid approach in which Kernel Mean Shift is the dominant technology, with a small number of particles being generated, in a structured fashion, to explore further when confidence in the Mean Shift algorithm becomes low.

The proposed algorithm, which we term the Structured Octal Kernel (SOK) filter, is described in Section 3.2. Section 3.3 describes the algorithms that SOK was gauged against, and the tests conducted. Experimental results are presented in Section 3.4. Section 3.5 compares SOK's structured search with the random particle placement normally associated with particle filtering, and conclusions are drawn in Section 3.6.

3.2 The Structured Octal Kernel Filter

The Structured Octal Kernel (SOK) filter is a Kernel Mean Shift tracker augmented by a backup strategy triggered when confidence in the current location estimate is low. Confidence at time t is given by

$$C_t = (1.0 - bhata(t)) \tag{3.1}$$

where bhata(t) in equation 3.1 is the Bhattacharya distance between object model and image data at time t.

C_t may be defined as a confidence level, ranging from 0 to 1, that shows if the target is on the object or not. A high value (close to 1) of C_t shows that target is on the object and vice versa. A user-defined threshold, T, is applied to C at each time step, If C_t is below threshold a set of eight independent Kernel Mean Shift trackers are spawned, each with the same object model as the original but at locations designed to cover a search area around the current position estimate (Figure 3.1). When these additional trackers have also each converged, nine estimates of target location are available, each with an associated confidence level. The estimate with the highest confidence is selected, control is shifted to single Mean Shift algorithm and the process continues. This mirrors the hybrid tracker of (Maggio and Cavallaro 2005); the algorithm effectively generates eight evenly spaced particles when confidence in the Mean Shift is low.

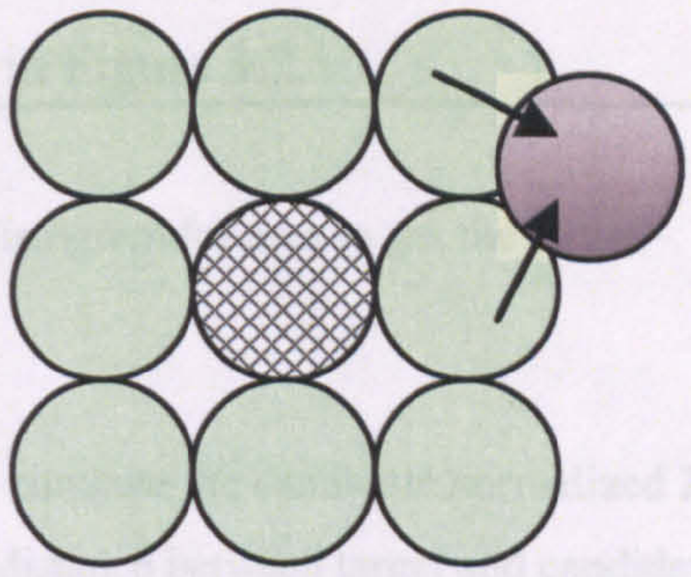


Figure 3.1. The SOK particle distribution. A hatched circle shows the primary KMS tracker, light circles the secondary "particles", a dark circle the target.

The appearance model used to represent the target object in all the implementations of tracking algorithms used in this chapter is a 2D colour histogram of size 50x50. The colour space used is RGB, with the red(r) and green (g) components normalized by the corresponding blue (b) value (r = r/b; g = g/b). This way the 2D histogram represents all the 3 colours with an added advantage. Division by the blue value reduces the effects of illumination changes, hence making the object model more robust to slight to medium lighting changes. Each red, blue and green component of each pixel is represented by 0-255 (256) values. Since we have only 50 bins for each r/b and g/b component, each bin represents a specific range, for instance the first bin represents r/b from 0-255/50 and g/b from 0-255/50.

The SOK algorithm is given in Figure 3.2.

1. Pick a target area (centre)
2. Compute a Normalized 2-D Histogram for area to get the target.
3. Loop {
    1. Get a frame.
    2. From the current centre compute the candidate normalized 2-D Histogram.
    3. Compute Bhattacharya distance between target and candidate
    4. Loop Till Bhattacharya distance becomes constant
        a. Hill climb towards the maxima
        b. Compute candidate histogram again
        c. Compute Bhattacharya distance again
    5. If Confidence < threshold value T
        a. Using current centre and radius place eight search areas in a structured manner as shown in the figure 3.1.
        b. Hill climbs each particle towards the nearest maxima.
        c. Choose the one with the lowest value of Bhattacharya distance
        d. The chosen particle is the new target centre.

}

Figure 3.2: SOK algorithm.

Normalisation ensures that the 2D histogram entries sum to 1, providing some robustness to movement along the line of sight. Note that radius of the object r and T remain fixed throughout and that $bhata(i, j)$, and so C, varies between 0 and 1, easing selection of T, which is chosen empirically.

The SOK algorithm combines the particle filtering with the Kernel Mean Shift algorithm in a simple, but effective manner. Recognising the strength of the Kernel Mean Shift algorithm in many situations it uses a single such tracker when confidence in the target location is sufficiently high. In areas of low confidence a burst of particles is emitted, allowing the tracker to search more widely. In the initial design the intention was to distribute these particles randomly. As there is no motion model in the Mean Shift tracker, however, and no prior distribution available to drive particle location, only a simple random distribution about the current location (e.g. uniform or Gaussian) is possible.

Noting the ability of the Kernel Mean Shift tracker to climb to a local maximum if and only if the tracking window overlaps the target object, we adopt the simple particle distribution of Figure 3.1. This uses a small, fixed number of particles to cover a regular search area around the current hypothesis. To be beyond this search area the object would have to move more than twice its own radius between frames, which is unlikely. If high velocity motion is expected the particle set can be extended to create a larger search area, though in such circumstances Kernel Mean Shift may not be the best approach and an explicit motion model may be required.

The SOK algorithm tracks its target using the simple Kernel Mean Shift algorithm as long as the confidence level stays above a certain threshold. The threshold is chosen to reflect the level of mismatch expected between the target and its appearance model. In the real world there can be no guarantee that lighting changes, shifts in camera position, camera sensor noise and other factors like shadows, partial occlusions etc. will not affect the target object's appearance. We can therefore never assume that the target 2D colour histograms or any appearance model of the tracked object will always remain 100% similar to the object model extracted from the first frame, where tracking commenced. Note also that Kernel Mean Shift tracking algorithms are accurate to one pixel only, so there is a good chance that the Mean Shift's best position on the target is still one or two pixels away from the true centre. This positional inaccuracy also reduces the confidence scores that can be expected during successful tracking.

If the confidence threshold is set too high, then there will be many outbursts of the 8 subsidiary trackers. This greatly increases the size of the search area and so the danger of the tracker being caught on background clutter. If the threshold is too low, however, then there is a danger that the tracker will latch onto objects with a different appearance model to the tracked object. Experience suggests that the threshold should be within 0.6-0.8. The default value of 0.7 has been found to work best in most scenarios.

3.3 Experimental Evaluation

3.3.1 Algorithms

The proposed tracker has been experimentally compared with three existing algorithms; Kernel Mean Shift, Condensation, and Maggio and Cavallaro's hybrid. Here we briefly review the methods involved and describe their implementations. The image sequences used are presented and discussed in section 3.3.5.

3.3.2 Kernel Mean Shift Tracker

The Kernel Mean Shift tracker (Comaniciu et al 2003) hill climbs from the previous location estimate toward a local minimum in the Bhattacharya distance between normalised, kernel weighted colour histograms representing the object model and local image data. We use a linear kernel having maximum weight at the centre and zero weight at boundaries and beyond. The object model and candidate model are 256 x 256 bin histograms recording red/blue against green/blue. This provides some robustness to changes in illumination. The histogram is normalised so the bin values sum to 1.

The Bhattacharya distance between model and candidate target is:

$$bhata = \sqrt{1 - \sum_{i}^{M}\sum_{j}^{M}\sqrt{p(i,j) \times d(i,j)}} \qquad (3.2)$$

where M is the size of each dimension of the histogram (256), and p and d are the object and the candidate models respectively. Note that the object model is computed only once. The candidate model is calculated in each frame from the position of the object in the previous frame.

The iterative Mean Shift operation is as follows:

$$x = \frac{\sum_{i}^{M}\sum_{j}^{M}\left(\sqrt{\frac{p(i,j)}{d(i,j)}} \times i\right)}{wt} \qquad (3.3)$$

$$y = \frac{\sum_{i}^{M}\sum_{j}^{M}\left(\sqrt{\frac{p(i,j)}{d(i,j)}} \times j\right)}{wt}$$

where

$$wt = \sum_{i}^{M} \sum_{j}^{M} \sqrt{\frac{p(i,j)}{d(i,j)}} \tag{3.4}$$

And x and y are the coordinates of the next estimate of the position of the centre of the object, and M is again the resolution of the histograms modelling object p and candidate.

3.3.3 Condensation

The particle filter used in the experiments conducted here is a straightforward implementation of Isard and Blake's (Isard and Blake 1998a) Condensation. The object and candidate models are exactly the same as those employed in the Kernel Mean Shift filter, with Bhattacharya distance between them computed in the measurement phase. A simple motion model – constant velocity – is used throughout and, unless otherwise stated, all experiments use 100 particles.

3.3.4 Hybrid Condensation/Kernel Mean Shift Tracker

This again is a straightforward implementation of an existing technique – the hybrid tracker of Maggio and Cavallaro (Maggio and Cavallaro 2005). The Condensation algorithm outlined in section 2.3.3 provides a harness into which the Kernel Mean Shift tracker outlined in section 2.3.4 is slotted. At each time step 100 (unless stated otherwise) particles are evaluated by computing the Bhattacharya distance between the object and their candidate model. A further 100 particles are then selected with probability proportional to their measurement value and projected into the next image by a constant velocity motion model. A Kernel Mean Shift tracker is initialised at each particle location and run until its associated Bhattacharya distance becomes zero or constant. The process is then repeated. This disperses the particle set in the Condensation phase, then draws it together in the Mean Shift phase.

3.3.5 Image Sequences and Evaluation Criteria

The four trackers described above have been evaluated and compared using a variety of real and artificial image sequences:

- Artificial sequences showing a multicoloured circular target moving across a white background allow the trackers' positional estimates to be compared to ground truth in the presence of controlled amounts of noise.

- A mixed set of 36 image sequences (some shown in Appendix A) allows evaluation of robustness. Sensitivity is measured across this data and reported, while McNemar's statistic provides principled, quantitative statements of the relative robustness of the competing algorithms.
- The computational cost of each algorithm is estimated by measuring processing times over the same test set.
- Numerical measures are supported with qualitative examination of selected image sequences. To examine robustness to background clutter a hand-held ball is moved in front of a complex environment and viewed by a fixed camera. The sequence comprises 220 384 x 288 pixel frames. To examine robustness to unpredictable motion, a hand-held camera is used to capture a 420 frame sequence of a child at play. Each frame is 720 x 576 pixels.

3.4 Results

3.4.1 Accuracy

Figure 3.3 shows the result of applying the four trackers to an artificial sequence in which a multicoloured target followed the path shown in Figure 3.4. This path comprises a number of straight sections corrupted by high levels (σ = 10 pixels) of Gaussian noise. Absolute error (in pixels) is plotted against frame number.

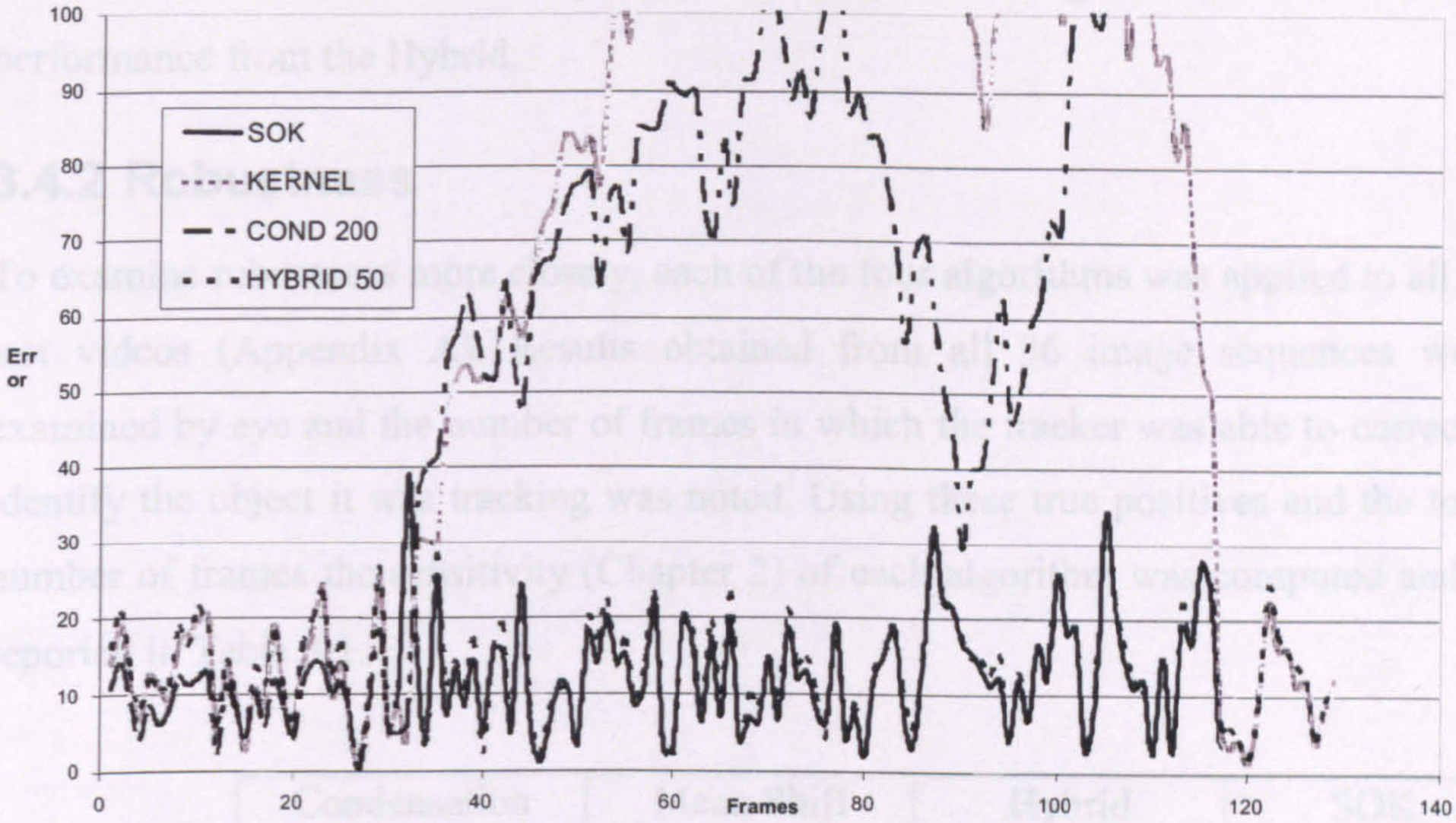


Figure 3.3. Absolute error (pixels) in four algorithms' tracking of a noisy (Gaussian, σ = 10) synthetic sequence showing a multicoloured target.

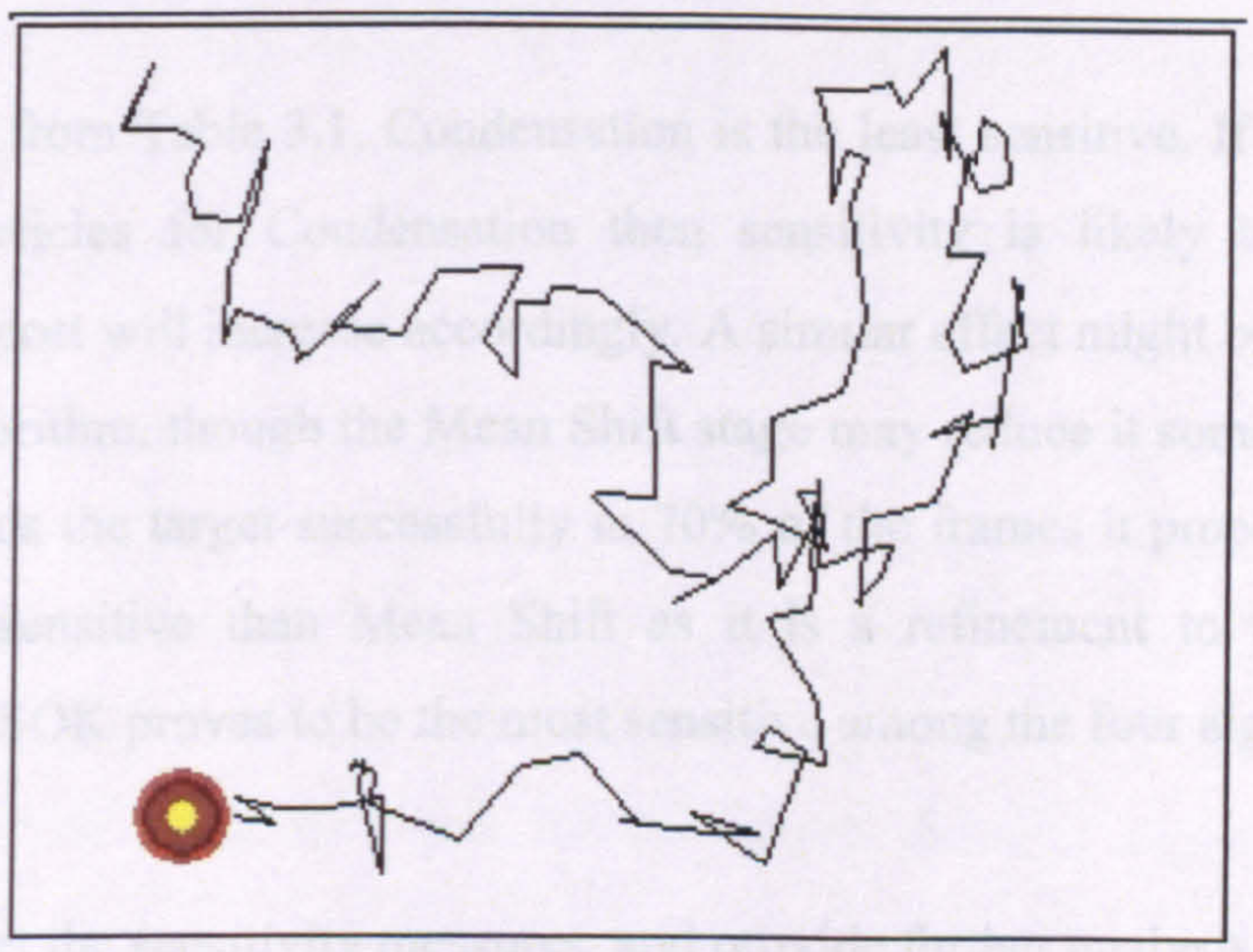


Figure 3.4. The multicoloured target moved over a white background to construct artificial test data.

The four algorithms produce similar levels of positional accuracy during the periods when they all track the target successfully. These periods are, however, fragmented. In the typical example shown in figure 3.3, Kernel Mean Shift and Condensation both fail after the first sudden change in trajectory, and only Maggio and Cavallaro's (Maggio and Cavallaro 2005) Hybrid and the SOK filter track successfully. Note

however, that the SOK filter used only one or eight particles, depending on tracking confidence, while at least 50 particles were needed to gain the same level of performance from the Hybrid.

3.4.2 Robustness

To examine robustness more closely, each of the four algorithms was applied to all 36 test videos (Appendix A). Results obtained from all 36 image sequences were examined by eye and the number of frames in which the tracker was able to correctly identify the object it was tracking was noted. Using these true positives and the total number of frames the sensitivity (Chapter 2) of each algorithm was computed and is reported in Table 3.1.

	Condensation	Mean Shift	Hybrid	SOK
Sensitivity	0.46	0.70	0.72	0.76

Table 3.1: Sensitivity reported for Condensation, Mean Shift, Hybrid and SOK

As we can see from Table 3.1, Condensation is the least sensitive. If we use a larger number of particles for Condensation then sensitivity is likely to increase, but computational cost will increase accordingly. A similar effect might be expected from the Hybrid algorithm, though the Mean Shift stage may reduce it somewhat. Standard Mean Shift finds the target successfully in 70% of the frames it processes, Hybrid is slightly more sensitive than Mean Shift as it is a refinement to the Mean Shift algorithm, and SOK proves to be the most sensitive among the four algorithms tested.

To help interpret the sensitivity measures, and provide further analysis of sensitivity at the image sequence, rather than frame, level, McNemar's test (Clark et al 2008) was applied to the output of the four trackers over the same set of 36 image sequences. To apply McNemar's, a definition of success and failure is required. Focussing on robustness, we define algorithm A to have succeeded and algorithm B to have failed if algorithm A maintains tracking for a greater proportion of a given image sequence, from the same starting parameters. In effect we define success to be tracking as long

as the better of the two trackers. The results of this exercise are presented in Table 3.2, and show:

97.5% confidence that SOK is more robust than Mean Shift.
96% confidence that SOK is more robust than the Hybrid filter.
98% confidence that SOK is more robust than Condensation

	SOK vs. Mean Shift	SOK vs. Hybrid	SOK vs. Condensation
Z	2.04	1.81	2.22
Confidence	97.5%	96%	98%

Table 3.2. McNemar's comparison of SOK with Mean Shift, Condensation and Hybrid trackers over the image sequences

Figure 3.5 shows selected frames from the four algorithms' tracking of a quickly moving, hand-held ball. Condensation fails after frame 35, when the particles diffuse towards different false local extrema. Kernel Mean Shift hovers around a confined area and loses the ball as soon as it moves quickly. Though it recaptures the ball later, when it passes under the Mean Shift window, this is not a robust effect. The Hybrid filter tracks quite well, but slips away a couple of times around frame 40. SOK tracks very well, using its structured backup when the ball slips away, e.g. in frame 40. SOK, however, slips around frame 210 and regains the target by chance again around frame 220, though it does track the object for the longest duration among the compared techniques.

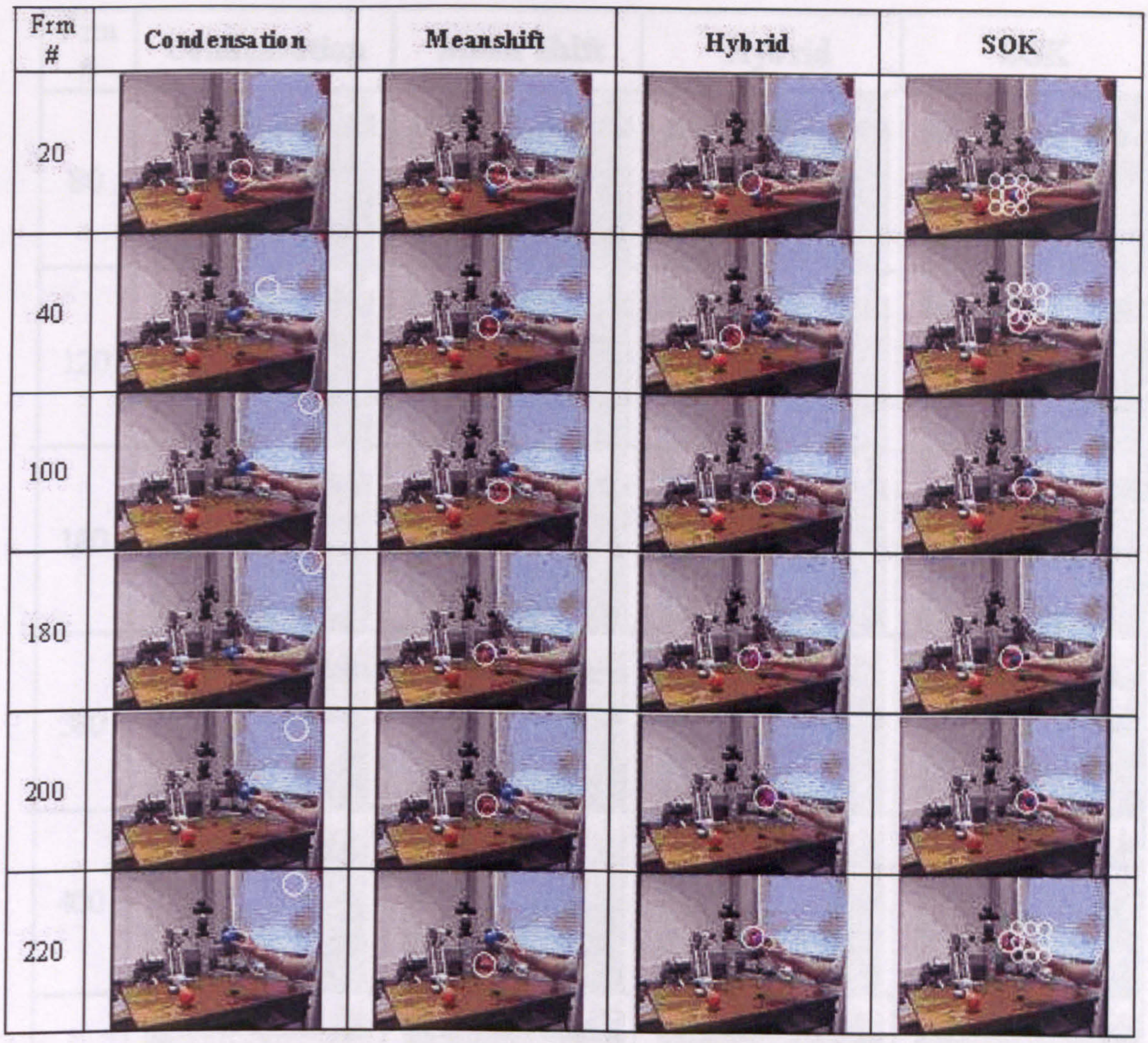


Figure 3.5. Tracking a hand-held ball through clutter.

Figure 3.6 summarises tracking of a young girl running and jumping in front of a hand held, moving camera. Condensation starts to fail before the camera Shifts suddenly around frame 180. Mean Shift and Hybrid track well until frame 180, then fail due to high levels of both camera motion and target acceleration. SOK uses its structured search strategy to lock on to the girl at frame 180 and tracks her for the remainder of the sequence.

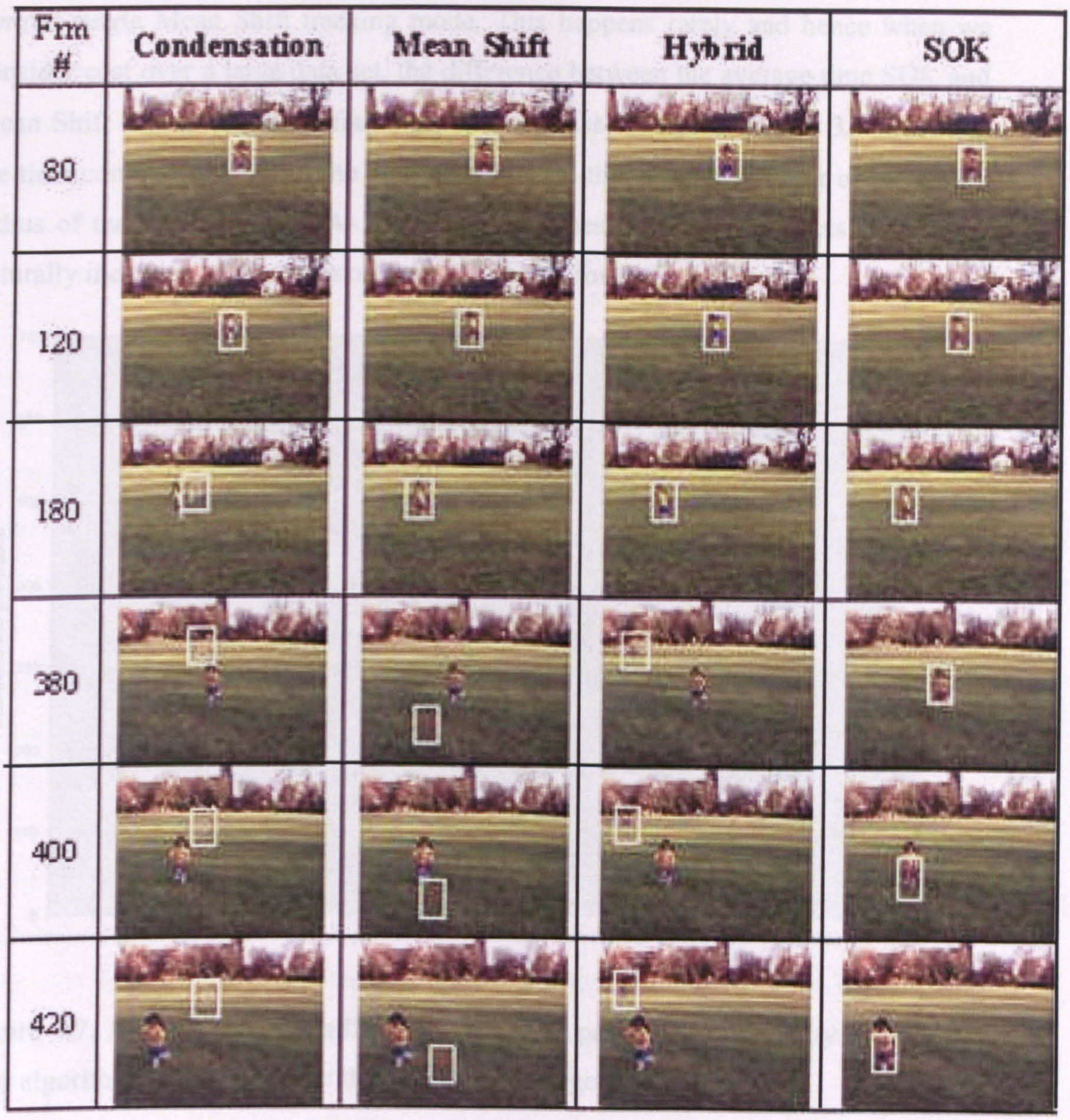


Figure 3.6. Tracking a girl at play, the camera is also moving and follows the girl.

3.4.3 Computational Cost

As SOK mainly uses Mean Shift, its cost in normal mode (using just Mean Shift to track) is the same as the original Mean Shift algorithm. The worst case scenario for SOK is when the algorithm switches to 8 particles, spread in a structured manner and each running a Mean Shift, making the cost 8 times that of a single Mean Shift. SOK only switches to structured particle mode if the confidence falls below a threshold, as soon as the object is reacquired and confidence is restored, SOK switches back to

normal single Mean Shift tracking mode. This happens rarely and hence when we consider cost over a large data set, the difference between the average time SOK and Mean Shift take to process a frame becomes almost negligible. Figure 3.7 illustrates the time comparisons for all the 36 videos used in this chapter in order of increasing radius of the tracked object. As the radius increases the time to process each frame naturally increases as there is more area to process for each algorithm.

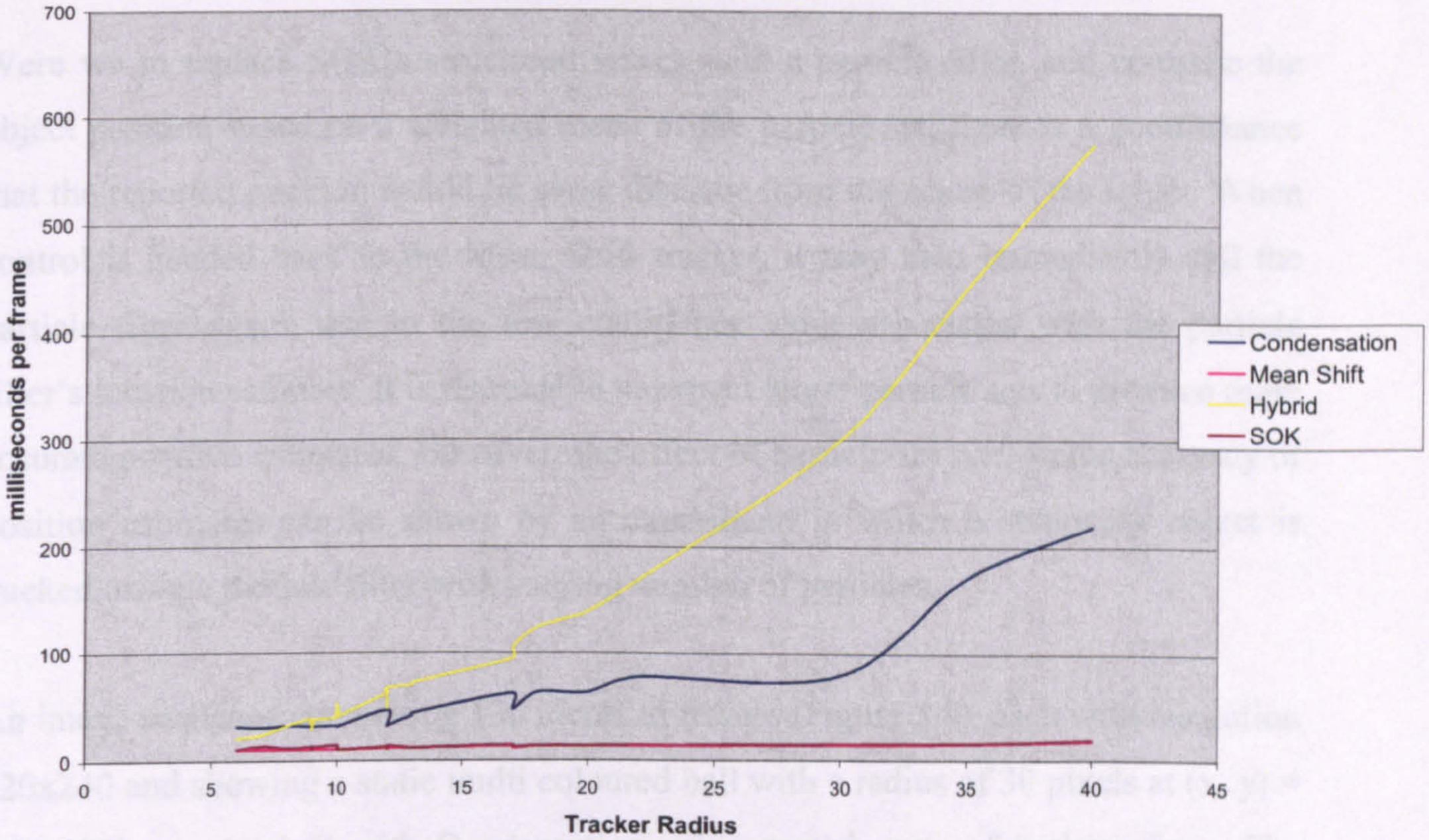


Figure 3.7. Average time in milliseconds taken to process one frame by each of the four algorithms, plotted against the radius of the target.

As shown in Figure 3.7, SOK and Mean Shift have roughly the same, and the lowest, average costs. All four algorithms show increased time consumptions as the radius of the tracked object is increased. Hybrid is the most expensive as it uses both Condensation and Mean Shift during frame processing.

3.5 Structured Search vs. Particle Filter

It could be argued that instead of using a grid of eight trackers around the object we could switch to a simple particle filter, locate the position of the object and once it is found transfer back to the sole Mean Shift tracker. Such an algorithm would be very similar to the hybrid tracker of Deguchi (Deguchi et al 2004), which runs Kernel Mean Shift and Condensation algorithms in parallel and uses the highest confidence

particle to initialise Mean Shift at each time step. To do this effectively, however, would require a large number of particles to be spread around the previous position of the tracker to accurately locate the object, this can be quite expensive. Secondly, when using a particle filter, we are never sure exactly where the target object's centre is. Most implementations rely on the weighted average of the x and y coordinates of the particle set to provide an estimate of the location of the object.

Were we to replace SOK's structured search with a particle filter, and compute the object position based on a weighted mean of the particle set, there is a good chance that the reported position would lie some distance from the centre of the target. When control is handed back to the Mean Shift tracker, it may then immediately call the particle filter again, due to the low confidence value associated with the particle filter's location estimate. It is reasonable to expect larger particle sets to produce more accurate position estimates. However, the effect of particle set size on the accuracy of position estimates can be shown by an experiment in which a stationary object is tracked using a particle filter with varying number of particles.

An image sequence comprising 130 identical frames (Figure 3.8), each with resolution 320x240 and showing a static multi coloured ball with a radius of 30 pixels at (x, y) = (100,100) was tracked with Condensation using particle sets of various sizes. The average error and the maximum error relative to the ground truth are reported in Table 3.3, and shown graphically in Figures 3.9 and 3.10.

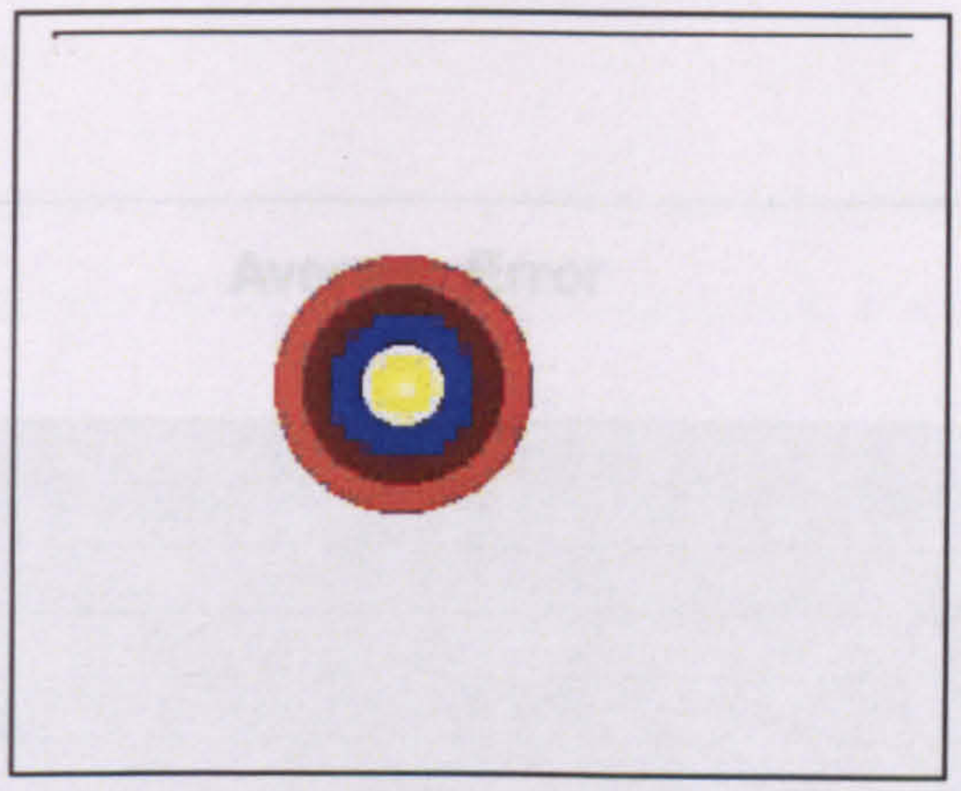


Figure 3.8: image sequence with resolution 320x240, with 130 frames, having a static multi coloured ball with a radius of 30 pixels at (x, y) as (100,100)

	20 Particles	50 Particles	100 particles	200 particles	300 particles	500 particles	1000 particles
Maximum Error	30	22	18	12	12	12	12
Average Error	16	9	8	6	5	5	5

Table 3.3: Showing maximum and average errors from particle filters with different number of particles.

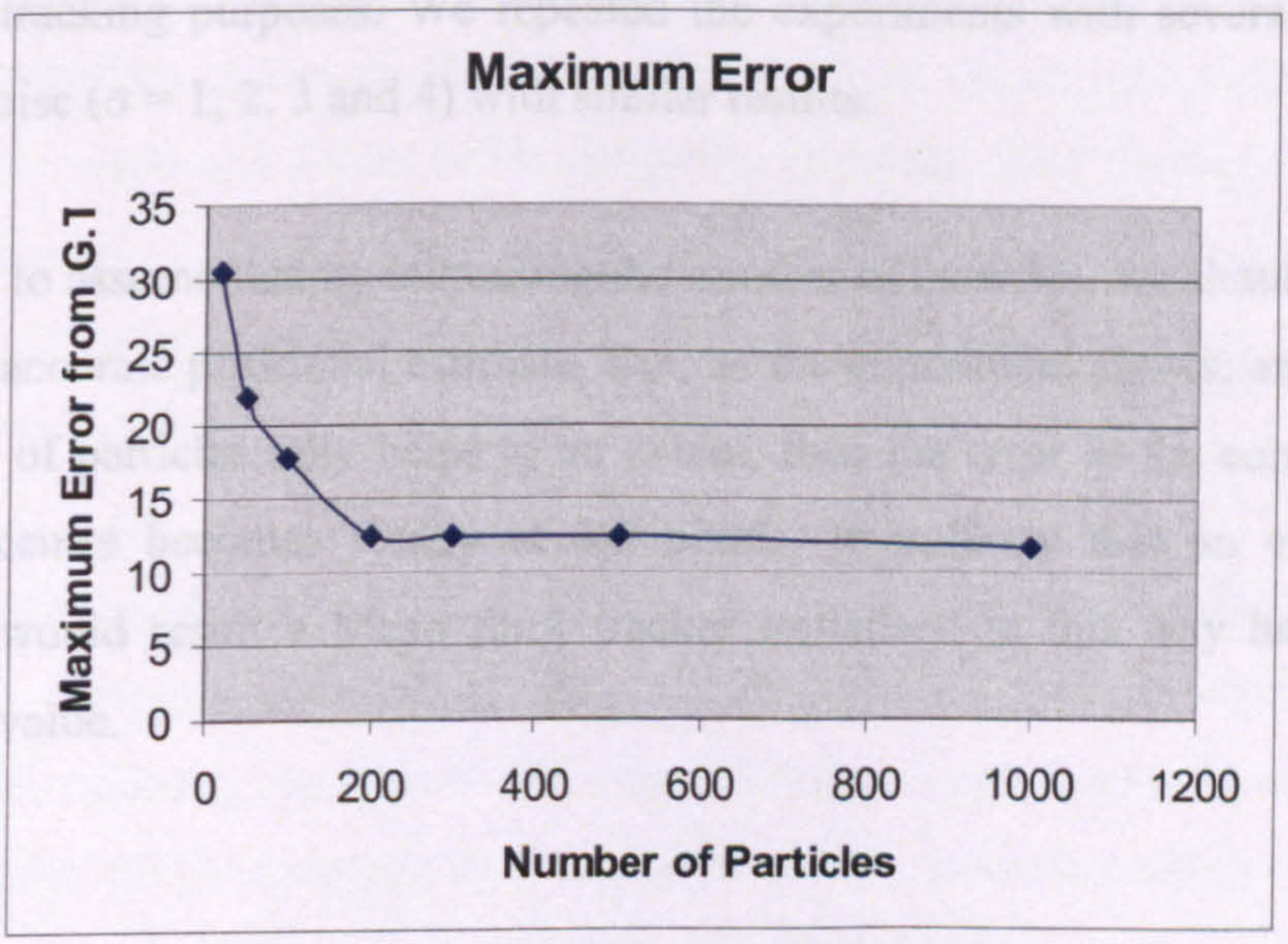


Figure 3.9: Maximum error for the entire track of 130 frames is plotted for varying number of particles. $\sigma = 1$

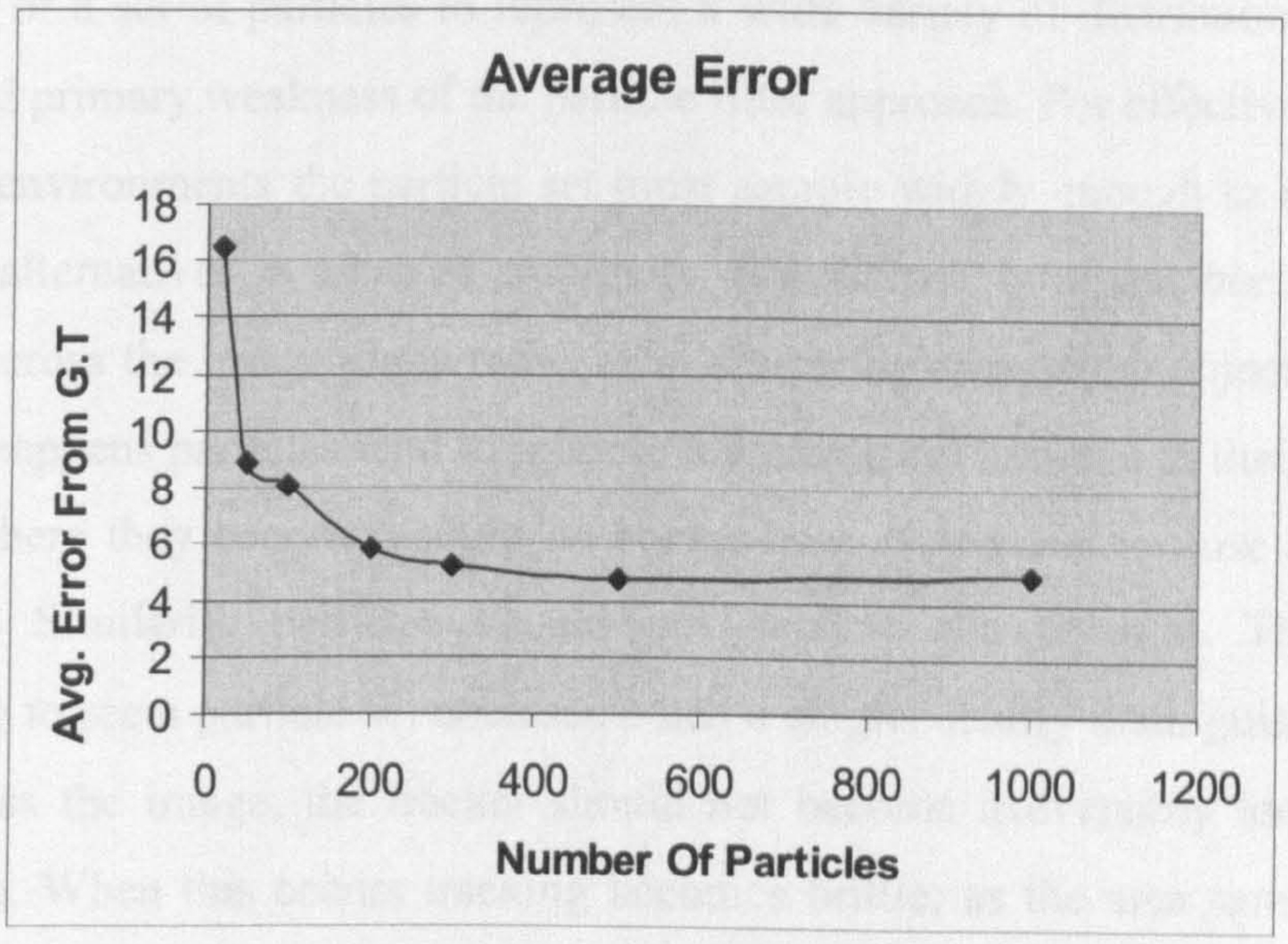


Figure 3.10: Average error for the entire track of 130 frames is plotted for varying number of particles. $\sigma = 1$

The number of particles used varied from 20 to 1000. We see that above 300 particles, the average error from the ground truth becomes almost constant for a given spread of particles. By spread we mean the random noise introduced in the chosen particles to scatter for tracking purposes. We repeated the experiments with several spreads of Gaussian noise ($\sigma = 1$, 2, 3 and 4) with similar results.

It is natural to assume that by increasing the number of particles, we should be able to get a more accurate positional estimate. But, as the experiment shows, an increase in the number of particles only helps to an extent, then the error in the computation of the target centre becomes steady at 4-5 pixels. It is likely that an error of this magnitude would result a Mean Shift tracker initialised in this way having a low confidence value.

3.6 Conclusion

The ability of a set of particles to represent a wide variety of distributions is both a strength and primary weakness of the particle filter approach. For effective tracking in real-world environments the particle set must sample widely enough to represent all reasonable alternatives in areas of ambiguity. It must not, however, become diffuse, spreading across the image plane rather than clustering around the object of interest. When this happens particles tend to migrate towards local maxima in their evaluation function, where they become caught on background clutter and so lose track of the true target. Similarly, particles should not become too focused. Though it is encouraging to see a particle set coalesce when a single, clearly distinguishable target moves across the image, the tracker should not become irreversibly locked onto a single mode. When this occurs tracking becomes brittle; as the area sampled by the particle set becomes smaller it is increasingly likely that noise or an unexpected movement of the target will cause it to become dissociated from the tracker.

Maggio and Cavallaro's (Maggio and Cavallaro 2005) hybrid tracker can be viewed as attempting to manage particle spread by alternately diffusing the particle set using Condensation and clustering it with Kernel Mean Shift. The Kernel Mean Shift tracker (Comaniciu et al 2003) is a robust and effective tracker with a very low computational cost. It performs well as long as the target object does not jump suddenly beyond its radius, or become occluded by an object with a similar model. Condensation (Isard and Blake 1998a) outperforms Mean Shift during sudden object or camera motion, provided that a large enough particle set is employed. Increasing the number of particles used, however, quickly increases computational cost. Maggio and Cavallaro's Hybrid attempts to gain the best of both worlds.

Maggio and Cavallaro's algorithm shows the performance expected of Condensation, but requires noticeably fewer particles, greatly reducing computational cost. The hybrid tracker typically requires 80-90% fewer particles than regular Condensation to achieve similar results (Maggio and Cavallaro 2005). Particle selection and initial posterior location is, however, managed by standard Condensation. If Condensation tends towards an incorrect local maximum, mean-shift will accelerate the process.

This chapter has proposed a hybrid tracker that makes explicit the iterative diffuse-cluster structure implicit in Maggio and Cavallaro's work, only diffusing when necessary and then carpeting a fixed area around the prior with particles. Experimental evaluation shows the proposed SOK filter to be more robust than the Condensation, Kernel Mean Shift, and Hybrid trackers over a statistically significant set of image sequences. SOK also provided more accurate tracking than Kernel Mean Shift and Condensation. Examination of computational cost showed SOK and Mean Shift to take 10-18 milliseconds on average to process a frame, depending on the radius of the tracked target. Both algorithms can easily be used at frame rates above 30fps, i.e. in real time, while the other two quickly become costly as target radius increases.

Chapter 4: Kernel Annealed Mean Shift Tracking

4.1 Introduction

The success of a predictive tracker relies on the effective combination of a motion model, which determines where any search should commence, and a search area of appropriate size and shape. An accurate motion model greatly eases the tracking problem by reducing the size of the region that must be searched. However, when the motion model is not a good fit to the actual motion of the target, or noise introduces errors into estimates of target state, use of a small search area may lead to the target being missed. This can be compensated for by increasing the size of the search area to allow for prediction errors, but any increase in search area is accompanied by an increased risk that the tracker will become attracted to background clutter that forms a local maximum in its evaluation function.

The SOK algorithm presented in Chapter 3 extended the Kernel Mean Shift tracking algorithm to increase the size of its search area whenever the algorithm's confidence in its lock on the target fell below a threshold value. Though the experimental results show SOK to be more effective than preceding hybrid particle filter/Mean Shift algorithms, the search patterns available to SOK are limited and its method of choosing between them is crude.

The SOK algorithm requires the user to specify the conditions under which extra particles are spawned and the size of the region to be searched. Though only a single threshold need be set, this is irksome and open to error. The need for a fixed threshold could be removed by recasting the SOK algorithm as a form of mixed-state particle filter (Isard and Blake 1998c). Instead of generating additional particles when confidence falls below threshold, the confidence value could be used to capture the transition probabilities between two states – one in which a single Mean Shift/particle is used, and one in which the larger search space is used. Regardless of the details of the switching mechanism, the larger search area used in the SOK algorithm is potentially very large. This, along with the simple hill-climbing search used, leads to a significant risk that SOK will climb to a local, rather than the global, maximum.

The work reported in this chapter aims to produce a hybrid, particle filter-based tracker with a similar, but more refined and flexible, ability to shrink and expand its search area. Rather than shift control away from the particle filter component and towards the Kernel Mean Shift tracker, Condensation is replaced with a more powerful particle filter.

A novel hybrid of the annealed particle filter (Deutscher et al 2000) and Kernel Mean Shift (Comaniciu et al 2003) tracking algorithms is presented, which we term Kernel Annealed Mean Shift tracking (KAMS). The use of a full particle-based representation allows KAMS to represent arbitrary, multimodal sets of hypotheses when searching for the target object. Kernel Mean Shift increases the robustness and accuracy of particle filtering by guiding particles towards maxima in the evaluation function; this allows particles to be spread over a larger area. Mean Shift, in turn, is complemented by annealing, which increases its effective range. The need for an accurate, predictive model of local motion is therefore reduced, increasing the generality of the approach.

Like SOK and the annealed particle filter, KAMS does not use motion estimates to predict future state, but begins its search from the last estimated position of the target. We hypothesize that by flattening local maxima in the evaluation function the annealed particle filter will allow a greater spread in the particle set, and reduce the

need for a possibly erroneous predictive motion model, while the Mean Shift component will continue to successfully pull particles back towards the true target

Annealed particle filtering (Deutscher et al 2000) is first reviewed in Section 4.2. The Kernel Annealed Mean Shift (KAMS) algorithm is then described in section 4.3. The robustness, accuracy, and computational costs of the algorithms are assessed in Section 4.4. KAMS is compared first with its constituent algorithms – Kernel Mean Shift (Comaniciu et al 2003) and annealed particle filtering (Deutscher et al 2000) – then with previous combinations of similar technologies. Results are discussed in section 4.5 before conclusions are drawn in section 4.6.

4.2 Annealed Particle Filtering

Annealed particle filtering (Deutscher et al 2000) relies upon a series of particle weighting functions $w_0(z,x)$ to $w_M(z,x)$ in which z is a measurement vector extracted from the image and x is the current model state. A given weighting function w_m is obtained by raising the original weighting function $w(z,x)$ to a power β_m, so that

$$w_m(z,x) = w(z,x)^{\beta_m} \tag{4.1}$$

where $\beta_0 = 1.0$ and $\beta_0 > \beta_1 > \beta_2 > \ldots > \beta_M$. As β_m increases, extrema in the weighting function become more pronounced. So $w_0(z,x)$ is the raw weighting function while $w_M(z,x)$ captures only the broad structure of the search space. In (Deutscher et al 2000) $w(z,x)$ is the sum of squared differences between the model and image data.

In annealed particle filtering each particle is evaluated at each time step using each $w_m(z,x)$, starting with $w_M(z,x)$ and moving to $w_0(z,x)$. At a given time step t_k the process begins with a set of N unweighted particles

$$s_{k,M} = \{s_{k,M}^{(0)}, s_{k,M}^{(1)}, \ldots s_{k,M}^{(N)}\} \tag{4.2}$$

Each particle $s_{k,M}^{(i)}$ is then assigned a weight $\pi_{k,M}^{(i)}$ where

$$\pi_{k,M}^{(i)} \propto w_m(z_k, s_k) \tag{4.3}$$

and, in the first step, $w_m(z_k, s_k^{i}) = w_M(z_k, s_k^{i})$, resulting in a set of weighted particles $s_{k,M}^{\pi}$. N particles are now drawn randomly from $s_{k,M}^{\pi}$ with replacement and used to create a set of un-weighted particles for evaluation using the next weighting function

$$s_{k,M-1}^{(i)} = s_{k,M}^{(i)} + B_m \tag{4.4}$$

where B_m is a multi-variate Gaussian random variable with mean 0 and variance p_m. $s_{k,M-1}^{\pi}$ is then weighted using $w_{m-1}(z_k, s_k^{i})$. This is repeated until $s_{k,0}^{\pi}$ is produced (Figure 4.1).

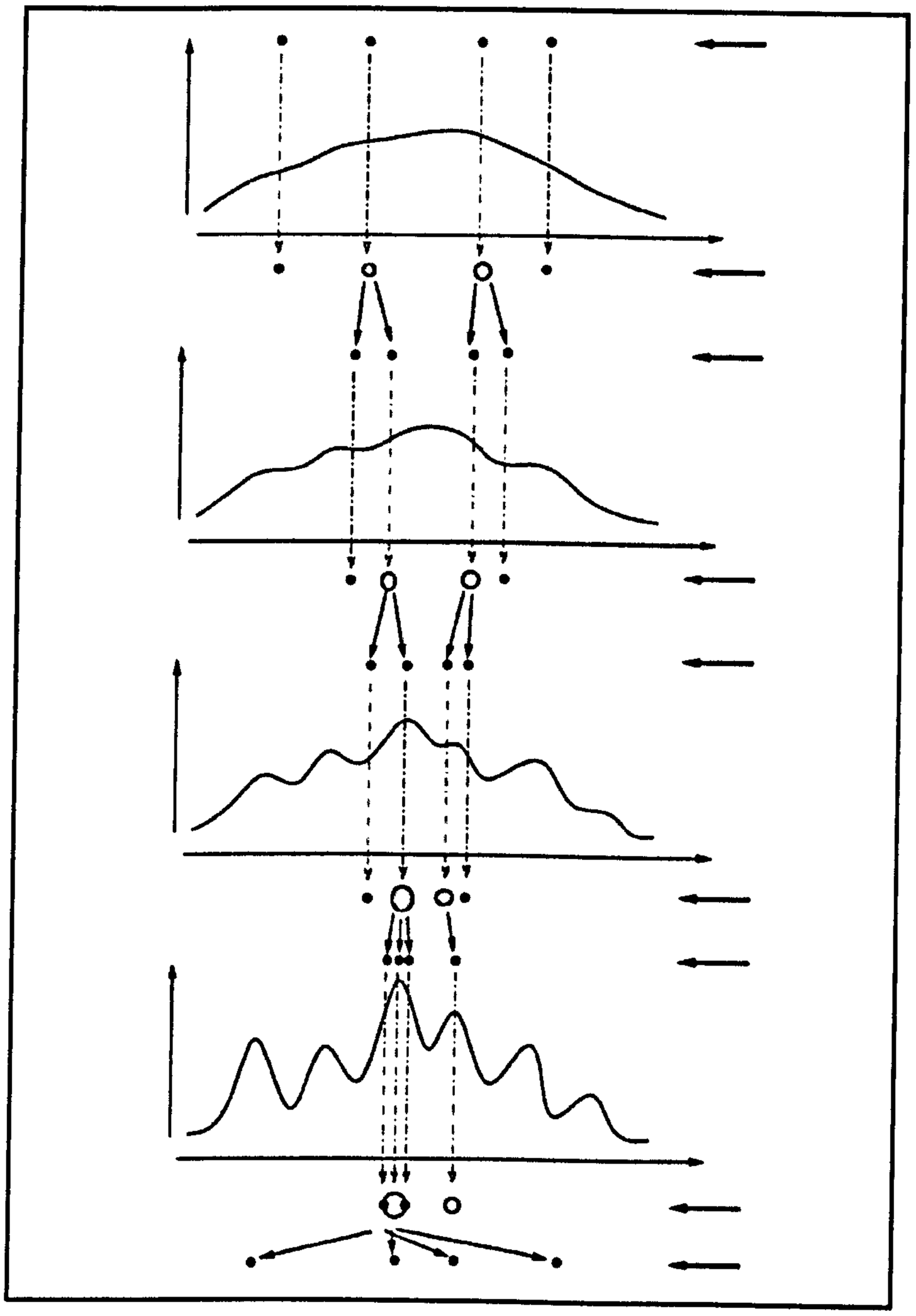


Figure 4.1: Illustration of a 4 stage annealed particle filter. The multi layered search allows particles to migrate towards the global maxima without getting stuck in the local maxima through each annealing stage.

Annealed particle filtering was developed to address the problem of high-dimensional search spaces requiring an impractically large particle set. By introducing narrow peaks in the evaluation function gradually it reduces the likelihood that particles will become locked onto these local artefacts and so the number of particles needed to identify the true target. In lower dimensional spaces, annealing allows us to counteract the natural tendency of particle filters to cluster particles together by increasing the variance P_m, confident that the smoother weighting functions used in the early part of the annealing run will steer particles away from local extrema. Increasing the spread

of the particle set, however, potentially increases the number of particles required to effectively sample the search area.

4.3 The Kernel Annealed Mean Shift Tracking Algorithm

To make explicit and accelerate the process of seeking the global maxima we apply a Kernel Mean Shift tracking step to each particle at each stage in the annealing run. The resulting KAMS algorithm is given in Figure 4.2.

Kernel Annealed Mean Shift Tracking:

Acquire frame at time t_k, having a set $s_{k,M}$ of N un-weighted particles from the previous time step

Set weighting function index m = M

While (m>0)

{

Assign each particle a weight $\pi_{k,m}^{(i)}$

Select N particles with replacement and add Gaussian noise:

$s_{k,m-1}^{(i)} = s_{k,m}^{(i)} + B_m$

Apply Kernel Mean Shift to each particle until the Bhattacharya distance between the model and image measured by the weighting function $w_{m-1}(z_k, s_{k,m-1}^{(i)})$ becomes stable or minimum.

m = m-1

}

Figure 4.2. The Kernel Annealed Mean Shift (KAMS) tracking algorithm

In the current implementation the object and candidate are 10 x 10 x 10 bin histograms (L=10) recording RGB colour values. The histogram is normalized to sum to 1. Experience has shown this to provide an effective compromise between

descriptive power and ability to generalise. Though any suitable kernel could be employed, for simplicity and generality we use a linear kernel having maximum weight at the centre of the circular target area and zero weight at boundaries and beyond.

The original annealed particle filter (Deutscher et al 2000) used sum of squared difference as its base weighting function. The Kernel Mean Shift algorithm (Comaniciu et al 2003) relied upon Bhattacharya distance. To allow comparison we employ Bhattacharya distance throughout. Kernel Mean Shift is run until Bhattacharya distance either falls below a small threshold or becomes stable. Experience has shown that this usually occurs within five iterations, so a limit on the number of iterations applied can reasonably be used, if needed, to reduce computation. Note that we follow Maggio and Cavallaro (Maggio and Cavallaro 2005) rather than Shan (Shan et al 2007) in that we sample directly from the Mean Shifted particle set. As annealed particle filtering is not a Bayesian technique there is no theoretical or practical reason to include the unshifted particles.

Following the original description of the annealed particle filter we use four stages. Figure 4.3 illustrates the flow of the algorithm for one frame.

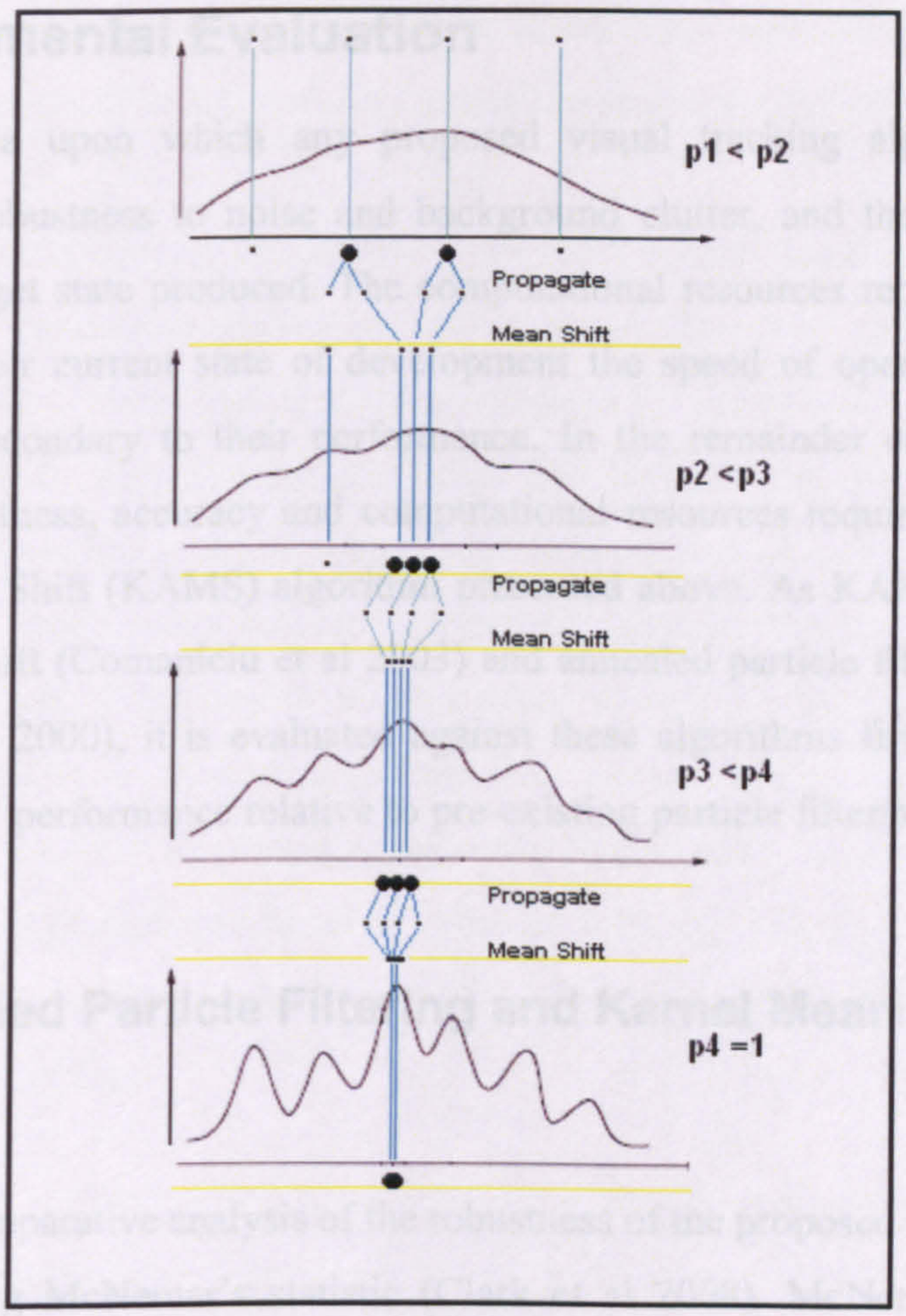


Figure 4.3: Visual representation of the KAMS filter in operation with four annealing stages.

In figure 4.3 the first stage (p1<p2) has the smoothest evaluation function. Particles are spread, weighted and then selected and projected onto the next annealing level (p2 < p3) based on their weights. After projection each particle is Mean Shifted towards a maximum in the second level's evaluation function. Once we have a concentrated set of particles around the target, particles are reweighed based on the p2 < p3 evaluation function and a further particle set selected based on the new weights. This set is projected onto the next level (p3 < p4), with a random component again added and Mean Shifted again.. This process focuses the particles on the global maximum in the base evaluation function and reduces the risk of them from getting stuck on local maxima.

4.4 Experimental Evaluation

The key criteria upon which any proposed visual tracking algorithm must be evaluated are robustness to noise and background clutter, and the accuracy of the estimates of target state produced. The computational resources required are also an issue, but at their current state of development the speed of operation of tracking algorithms is secondary to their performance. In the remainder of this section we assess the robustness, accuracy and computational resources required by the Kernel Annealed Mean Shift (KAMS) algorithm presented above. As KAMS is a hybrid of Kernel Mean Shift (Comaniciu et al 2003) and annealed particle filtering techniques (Deutscher et al 2000), it is evaluated against these algorithms first. Attention then shifts to KAMS' performance relative to pre-existing particle filter/Mean Shift hybrid trackers.

4.4.1 Annealed Particle Filtering and Kernel Mean Shift

Robustness

Quantitative, comparative analysis of the robustness of the proposed KAMS algorithm is achieved using McNemar's statistic (Clark et al 2008). McNemar's test, which requires 30 data items to provide a reliable result, was applied to a set of 36 assorted image sequences (some selected videos with description can be found in appendix A, the full set is available from the web-link (Link 2). The image sequences used were chosen to be representative of the type of data a general purpose tracking algorithm might be required to process. The test set contains both synthetic and real world data showing targets of varying size, shape, appearance and motion characteristics. The artificial sequences include varying levels of background clutter and motion noise. The real world data contains videos of different resolutions showing a range of scenarios. These include animals in the wild, sports (table tennis, soccer, basketball), children playing in a park, pedestrians, microscopic cells such as sperm and plant cells, and some surveillance data from the PETS (PETS Dataset 2007) test set.

Comparing KAMS with Kernel Mean Shift and annealed particle filtering in this way gives Z-scores of 5.12 and 3.8 respectively. Kernel Annealed Mean Shift tracking is significantly more robust than both its two constituent algorithms, with a confidence

of 99.5%. All algorithms were manually initialized to the same point and both KAMS and annealed particle filtering used four annealing stages throughout.

The strength of McNemar's test is that it provides a clear statement of the relative performance of competing algorithms. In isolation, however, this statement lacks detail. To provide a more fine grained assessment of the relative robustness of the Kernel Mean Shift, annealed particle filter and KAMS algorithms the results of applying each algorithm to all 36 image sequences were analyzed and the number of frames in which each tracker was able to correctly identify the object it was tracking was recorded. Using these counts of true positives and the total number of frames involved the sensitivity of each algorithm was estimated and given in Table 4.1. While Kernel Mean Shift displayed a sensitivity of 0.7, and annealed particle filtering 0.6, the KAMS algorithm produced a value of 0.92. KAMS identified the tracked object successfully in 92% of the frames considered, a considerable improvement on its component algorithms.

	Mean Shift	Annealed Particle Filter	KAMS
Sensitivity	0.7	0.6	0.92

Table 4.1: Sensitivity reported for Mean Shift, Annealed particle filter and KAMS

Figures 4.4 and 4.5 show selected frames from the results of applying these three algorithms to samples of the sequences used in the McNemar's test. Figure 4.4 shows a tiger sprinting through dense jungle. The animal's motion is smooth, but quite fast, with frequent changes in head direction. Surrounding trees generate many partial occlusions and the dark stripes on the animal and the shadows caused by the leaves are similar, generating high levels of potentially confusing background clutter.

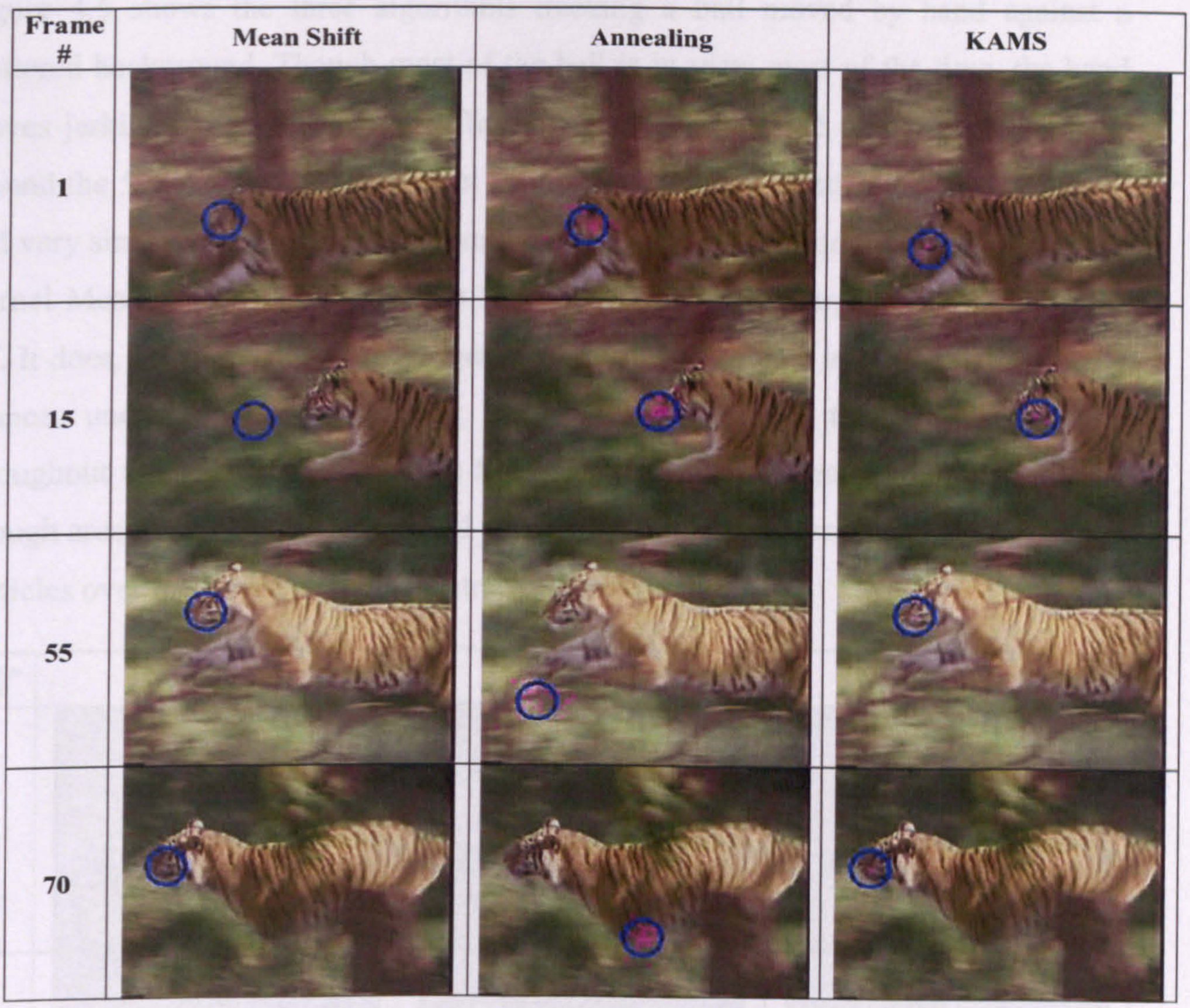


Figure 4.4. KAMS and its component algorithms track a sprinting tiger.

Mean Shift fails around the 15th frame due to the tiger's high speed, but latches back onto the head by chance around frame 41. The annealed particle filter fares better, and keeps hold of the object until around the 50th frame, when changes in lighting conditions temporarily make clutter within its search area appear more like the head model than the true head does. At this point there is a danger that any tracking algorithm will migrate to the background, as the annealed particle filter does here. Its selection mechanism quickly pulls the particle set towards the local spike in the evaluation function and tracking is lost. KAMS successfully tracks to the end of the sequence. Though its particles are also drawn towards the spike, the Mean Shift step keeps some locked onto the true target. When the spike disappears, and the true target again becomes apparent, these particles dominate once more.

Figure 4.5 shows the three algorithms tracking a ball moved by hand against a cluttered background. Though most of the ball is in view most of the time, the hand moves jerkily, and at a range of different velocities Annealed particle filtering fails around the 5th frame as its particle set is too dispersed and so attracted to very heavy, and very similarly coloured, background clutter. A tight focus on the target allows the Kernel Mean Shift to track until the 58th frame, when high target velocity throws it off. It does, however, regain the target around the 118th frame as the hand moves, by chance, underneath the wandering tracker. KAMS tracks the ball successfully throughout the sequence. Its particle filter stage spreads the search area over a large enough area to capture the high speed movement, while its Mean Shift stage keeps the particles over the target, despite a distracting background.

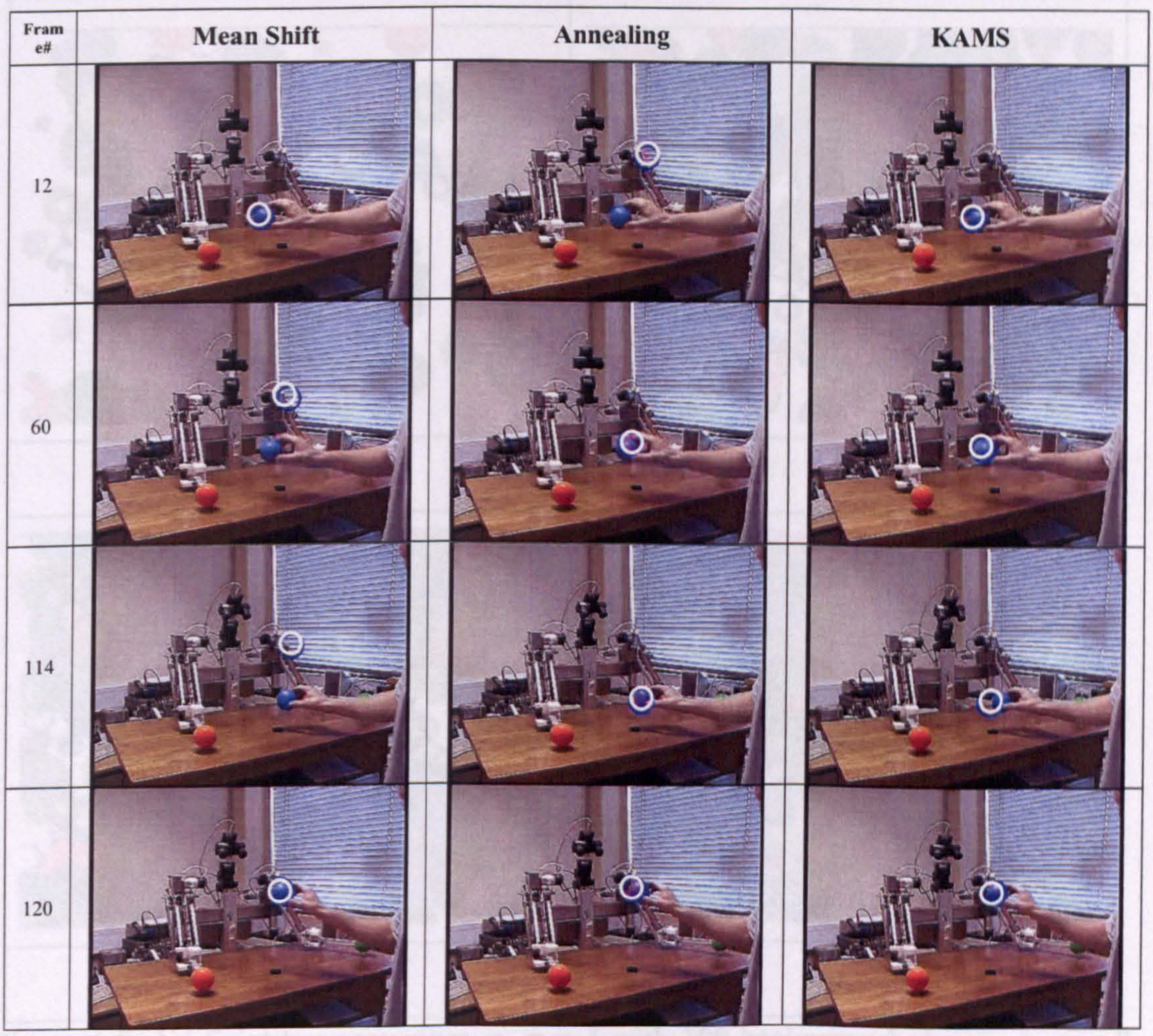


Figure 4.5. KAMS and its component algorithms track a hand-held ball.

Accuracy

Artificial sequences showing a multi-coloured circular target moving across a static background allow the trackers' positional estimates to be compared to ground truth in the presence of controlled amounts of measurement noise and clutter. Noise is simulated by perturbing the target's position in each frame with additive Gaussian noise. Clutter is added by randomly placing a user-defined number of similar circular objects on the otherwise white background. These distracting objects introduce local maxima into the evaluation function, while increased measurement noise raises the likelihood that a given tracker will come into contact with those maxima. Figure 4.6 summarizes the four test sequences used here; each consists of 140 (320x240 pixel) frames.

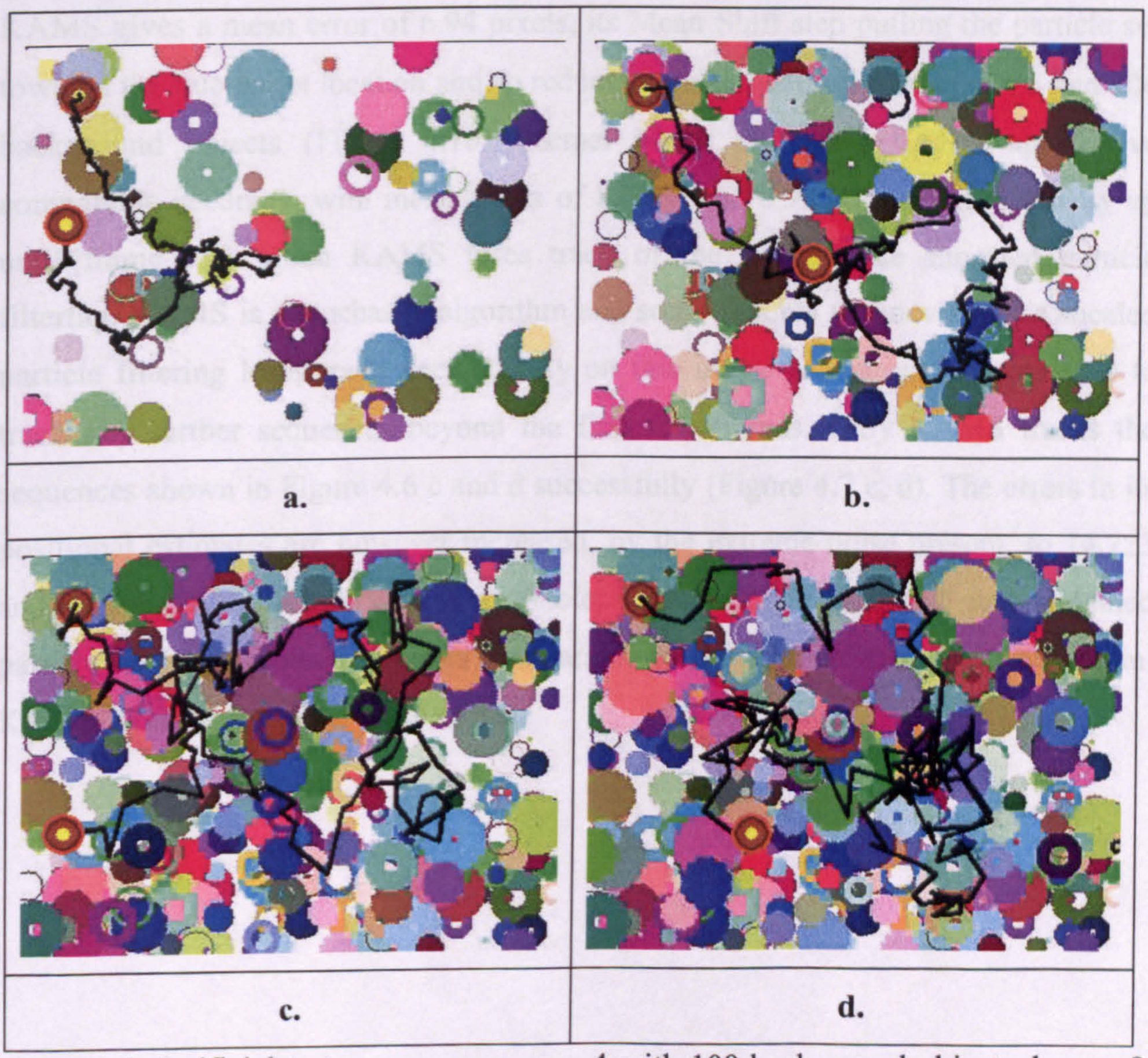


Figure 4.6. Artificial test sequences, a. $\sigma = 4$ with 100 background objects, b. $\sigma = 8$ with 300 objects, c. $\sigma = 12$ with 500 objects, and d. $\sigma = 14$ with 600 objects. Black lines show target path, with the target displayed at either end.

Figure 4.7 a-d give the error between the true target position and the estimates provided by the KAMS, Kernel Mean Shift and annealed particle filter algorithms over the four test sequences shown in Figure 4.6 a-d respectively. The mean error exhibited by each of these algorithms is shown in Table 4.2. The weighted mean of the particle set is calculated to generate a single position estimate from KAMS and annealed particle filtering, which again both employed four annealing stages.

With $\sigma = 4$ and 100 background objects (Figure 4.7a) all three algorithms track the object successfully through the sequence. The annealed particle filter's particle set is, however, spread out somewhat, giving a mean error of 11.02 pixels. Kernel Mean Shift is, as might be expected, the most accurate, with a mean error of 5.44 pixels. KAMS gives a mean error of 6.94 pixels, its Mean Shift step pulling the particle set towards the true target location and so reducing positional errors. With $\sigma = 8$ and 300 background objects (Figure 4.7b), Kernel Mean Shift and KAMS again give comparable accuracy, with mean errors of 8.0968 and 8.9294 pixels respectively up until frame 115, when KAMS loses track of the target. Like annealed particle filtering, KAMS is a stochastic algorithm and some failures are inevitable. Annealed particle filtering loses track very quickly on this image sequence, and also fails to track any further sequences beyond the first few frames. Only KAMS tracks the sequences shown in Figure 4.6 c and d successfully (Figure 4.7 c, d). The errors in its positional estimates are however increased, by the extreme noise present, to 14.723 and 17.303 pixels. Where data is available, KAMS is more accurate than annealed particle filtering, producing errors comparable to the Kernel Mean Shift algorithm. KAMS is, however, much more robust.

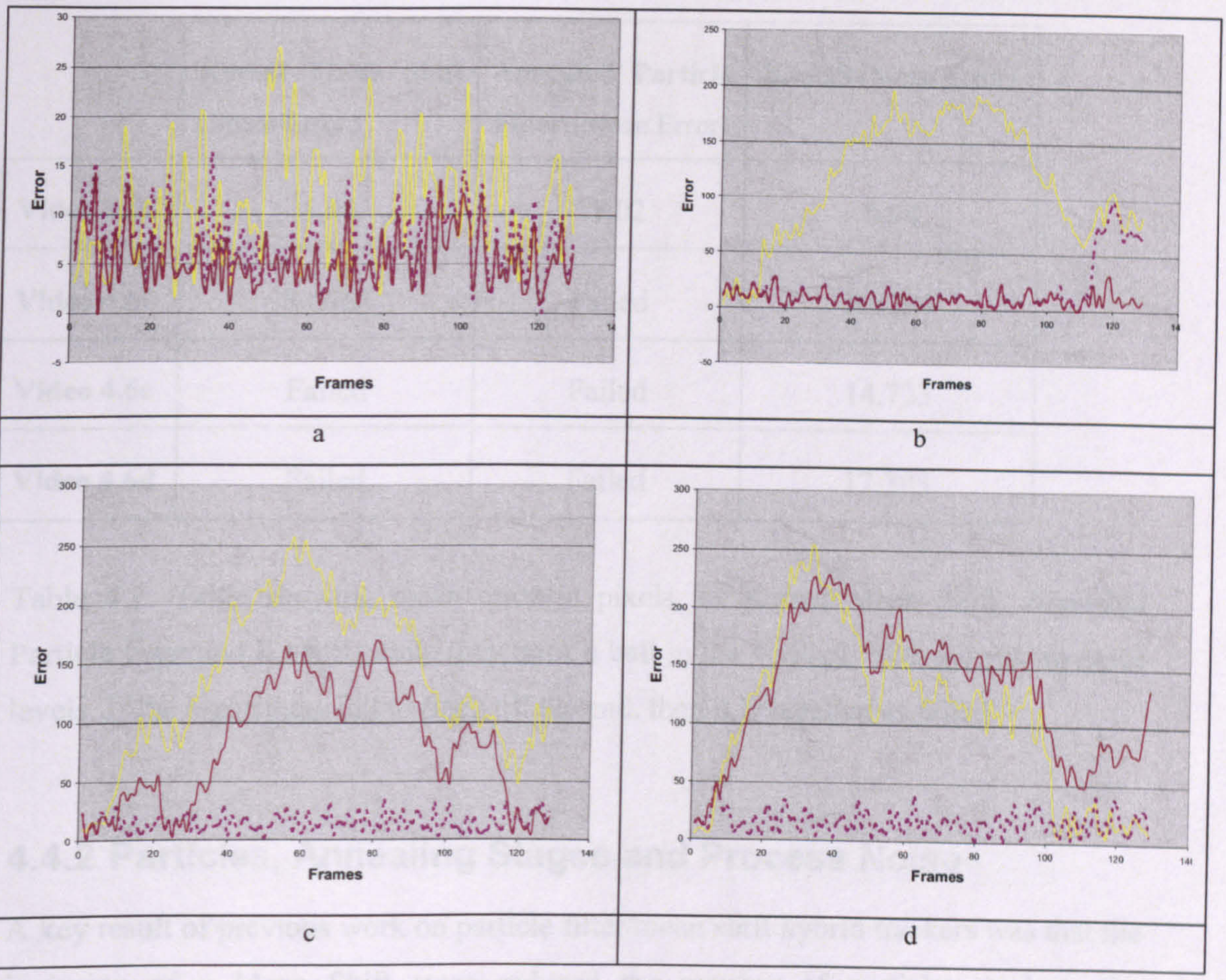


Figure 4.7. Errors in the position estimates provided by KAMS (dashed line) annealed particle filtering (light line) and Kernel Mean Shift (dark line) tracking, when applied to the image sequences summarized in Figure 4.6.

	Kernel Mean Shift (Mean Error)	**Annealed Particle Filter** (Mean Error)	**KAMS** (Mean Error)
Video 4.6a	5.44	11.02	6.94
Video 4.6b	8.0968	Failed	8.9294
Video 4.6c	Failed	Failed	14.723
Video 4.6d	Failed	Failed	17.303

Table 4.2: Table showing mean error in pixels, of Kernel Mean Shift, Annealed Particle filter and KAMS, while they track a ball in the 4 videos with increasing noise levels. If the algorithms fail to track till the end, then it is labelled as failed.

4.4.2 Particles, Annealing Stages and Process Noise

A key result of previous work on particle filter/mean shift hybrid trackers was that the inclusion of a Mean Shift stage reduced the number of particles, and so the computational resources, required for successful tracking. That result has been replicated here; in the experiments described above annealed particle filtering used 200 particles throughout, while KAMS required only 20 to produce superior performance. These numbers are in line with those reported by Maggio (Maggio and Cavallaro 2005) and Shan (Shan et al 2007).

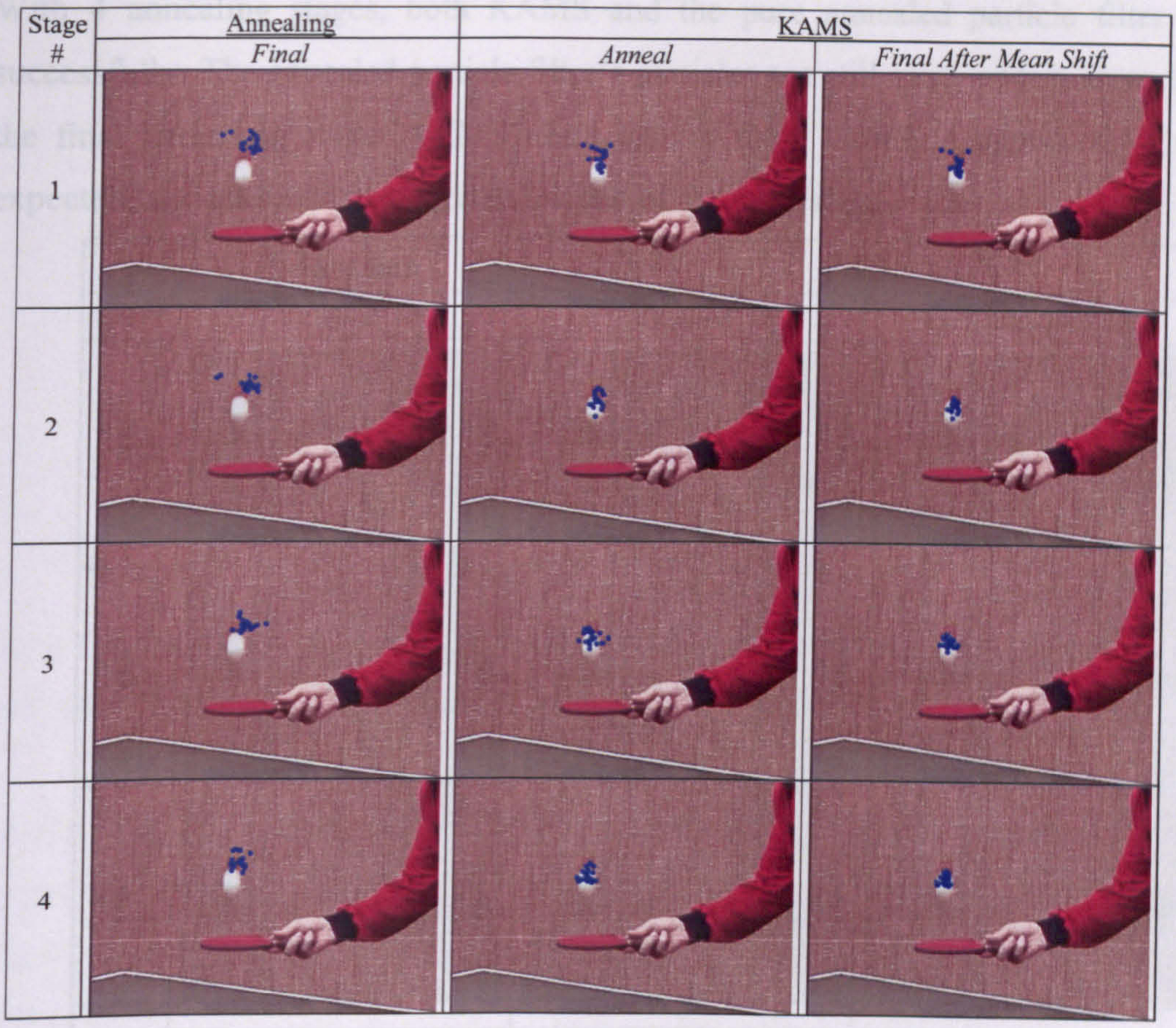


Figure 4.8. KAMS and the annealed particle filter track a bouncing ball using 4 annealing stages. See text for details.

KAMS' ability to track using a smaller particle set suggests that the algorithm is targeting its particles more effectively. This raises the possibility that fewer annealing stages might be needed in KAMS than in annealed particle filtering, reducing computational load further. To examine the effect of reducing the number of annealing stages employed in KAMS, the algorithm has been applied to a number of real and artificial test sequences, drawn from those described above. Figures 4.8-4.10 show typical results when tracking a table tennis ball bounced repeatedly on a bat. In each figure the left column shows the particle set produced, at each annealing stage, by the annealed particle filter. The middle column shows the particle sets generated at the same point in the sequence, before Mean Shift, by KAMS. The right column shows those particles after KAMS' Mean Shift operation.

With 4 annealing stages, both KAMS and the pure annealed particle filter track successfully. The annealed particle filter's particles are still quite widely spread after the final annealing stage. KAMS' particles are more closely grouped, as is to be expected, and all lie on the target at the end of the third stage.

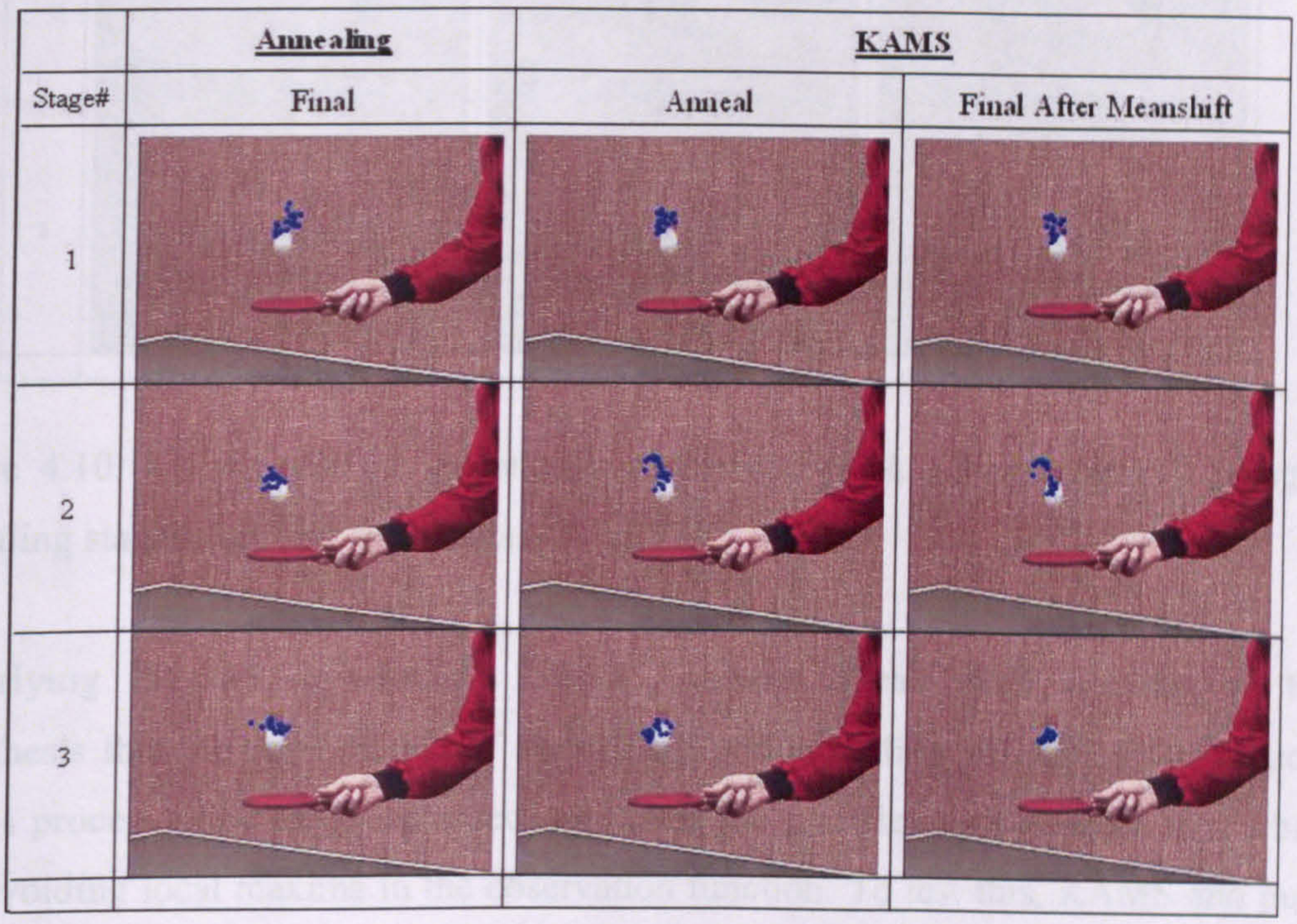


Figure 4.9. KAMS and the annealed particle filter track a bouncing ball using 3 annealing stages. See text for details.

Figure 4.9 shows the particle spread when both algorithms are run with 3 annealing stages. Particle placement in the annealed particle filters reduces in accuracy compared to KAMS. Annealing fails at stage 3 and loses track in the next frame. KAMS, on the other hand, continues to track well. Figure 4.10 shows tracking with two stages, though both the raw annealed particle filter struggle to hold onto the target, KAMS' Mean Shift step pulls several particles onto the ball and tracking continues. KAMS is more robust to reductions in annealing stages than the original annealed particle filter.

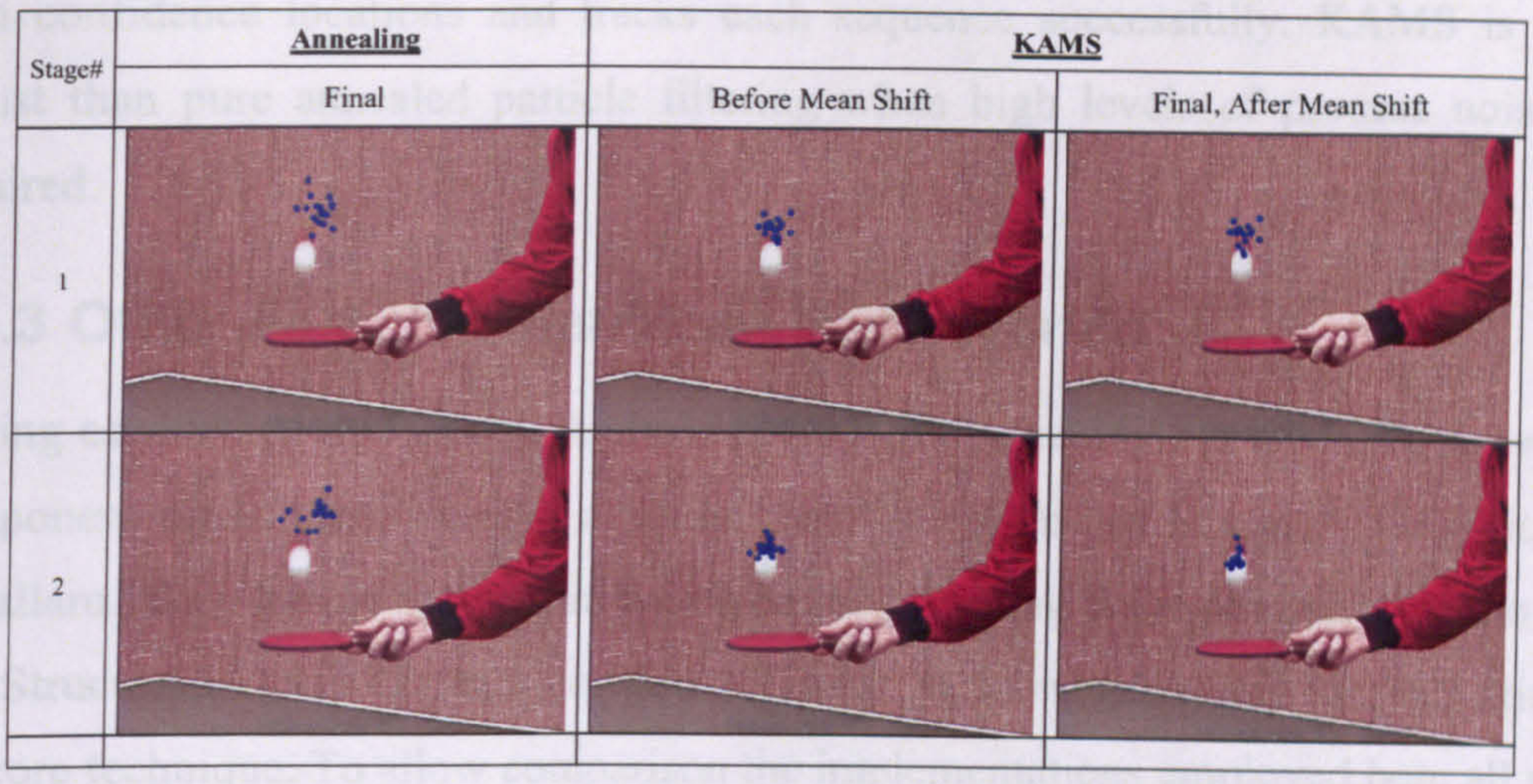


Figure 4.10. KAMS and the annealed particle filter track a bouncing ball using 2 annealing stages. See text for details.

Underlying the development of Kernel Annealed Mean Shift tracking is the hypothesis that the combination of Mean Shift and annealing will allow the particle filter's process noise to be increased, spreading the particles over a wider area while still avoiding local maxima in the observation function. To test this, KAMS and pure annealed particle filtering were both run on the artificial image sequence of Figure 4.6b with process noise settings of $\sigma = 1, 2, 4, 8, 10$ and 12.

With low process noise levels ($\sigma = 1, 2$) annealed particle filtering cannot cope with the high speed motion of the target; the particle set lags behind and tracking is lost. KAMS tracks successfully, because some of the particles are close enough to the target to be pulled to the correct maximum by the Mean Shift step. The selection process then generates more particles from these at the next time step. With $\sigma = 4$ the spread of the particle set neatly matches the speed and size of the target and both algorithms track safely. The particle spread is a little too large when $\sigma = 8$. Though it samples the area in which the target lies and tracks successfully, the distance between particles is too great to allow annealed particle filtering to track with confidence. Increasing the number of particles would, however, make the annealed particle filter robust here. KAMS tracks well, as Mean Shift moves the particles to high confidence locations. With higher levels of process noise annealed particle filtering fails – its particle set is spread over too large a search area. KAMS again pulls particles onto

high confidence locations and tracks each sequence successfully. KAMS is more robust than pure annealed particle filtering when high levels of process noise are required.

4.4.3 Other Particle Filter/Mean Shift Hybrids

Having established that Kernel Annealed Mean Shift tracking has advantages over its component algorithms, we now evaluate it in the context of Maggio's (Maggio and Cavallaro 2005) hybrid method, in which particle filtering is the dominant technology, and Structured Octal Kernel algorithm (Chapter 3), in which Mean Shift tracking is the core technique. To allow comparison the implementations employed here all use a 10 x 10 x 10 RGB histogram to represent the target.

Robustness

A McNemar's comparison of KAMS with Maggio and Cavallaro's hybrid and SOK algorithms, using the 36 image data set described above, gives Z-scores of 4.83 and 3.59 respectively. KAMS is significantly more robust than both algorithms with a confidence of 99.5%.

	Hybrid	SOK	KAMS
Sensitivity	0.72	0.76	0.92

Table 4.3: Sensitivity reported for Hybrid particle filter, SOK filter and KAMS.

In Table 4.3, Maggio and Cavallaro's hybrid displays a sensitivity of 0.72, and SOK 0.76, in comparison to KAMS' 0.92. Though it should be noted that both the hybrid and SOK algorithms fare better than raw Condensation, which achieves a sensitivity of only 0.46 over the same dataset, KAMS performs considerably better. KAMS has the highest sensitivity rate among all the single target tracking algorithms considered here.

Figure 4.11 summarizes the result of applying KAMS, SOK and Maggio and Cavallaro's hybrid algorithm to the tiger sequence of Figure 4.4.

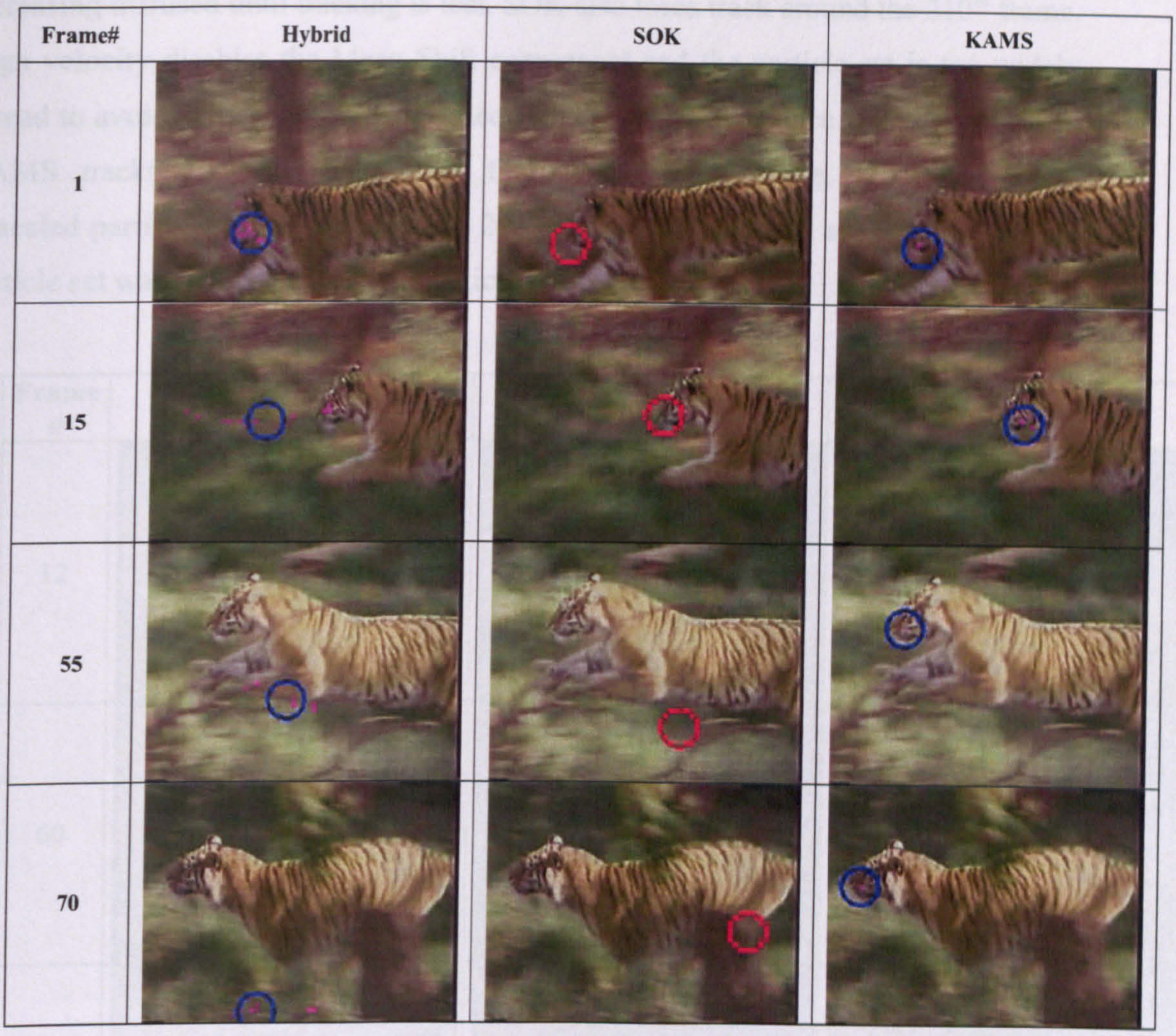


Figure 4.11. Three hybrid mean-shift/particle filtering algorithms track a sprinting tiger.

In figure 4.11 Maggio's hybrid tracking algorithm fails at frame 12 as the frequent changes in head velocity, despite being quite smooth, violate its motion model. The SOK algorithm fares better, but its large search area covers the spike generated around frame 50. As it only maintains one hypothesis, the SOK filter latches irretrievably onto the spike and, when the spike disappears, tracking is lost. KAMS successfully tracks to the end of the sequence.

Figure 4.12 shows the three algorithms' performance on the ball sequence of Figure 4.5. Maggio's Hybrid fails around the 59th frame. No motion model can adequately describe the near-random movement of the target, and the particle set becomes

increasing diffused until tracking is lost. SOK also loses track around the 210th frame. High velocity disables the Mean Shift component and the particle set is too widely spread to avoid clutter. Again SOK reacquires the target by chance around frame 220. KAMS tracks the ball successfully throughout the sequence. Moreover, while annealed particle filtering failed using 200 particles, KAMS still succeeded when its particle set was reduced to only 50 particles.

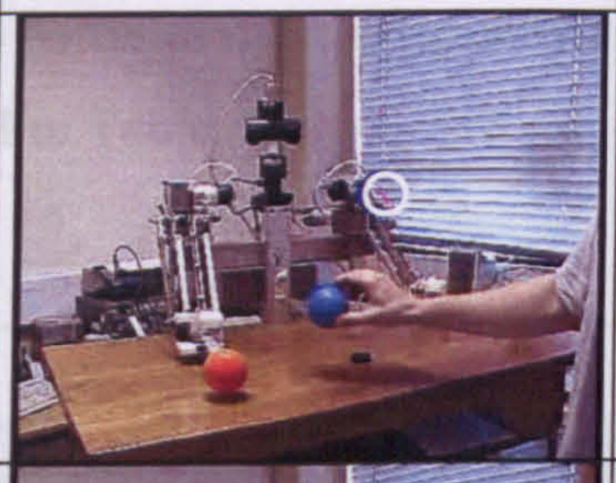
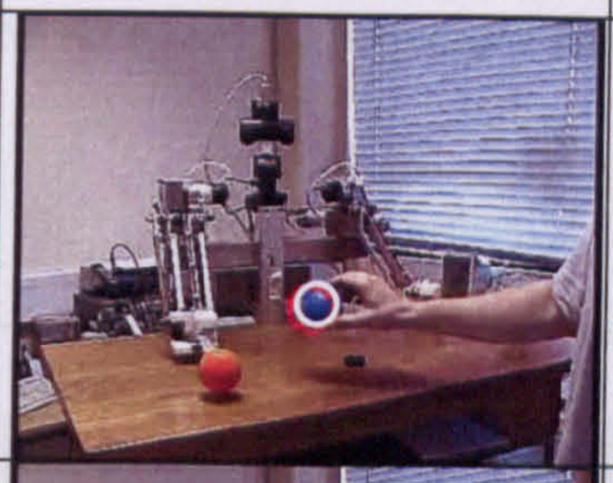
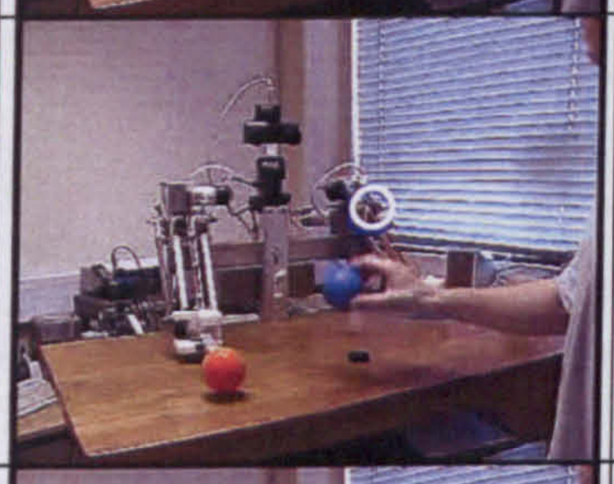
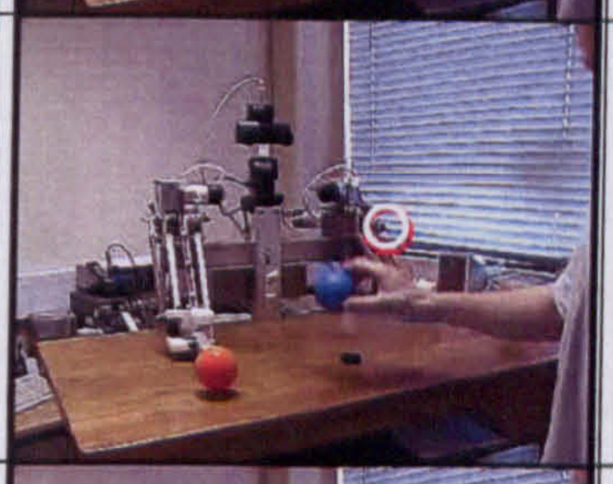
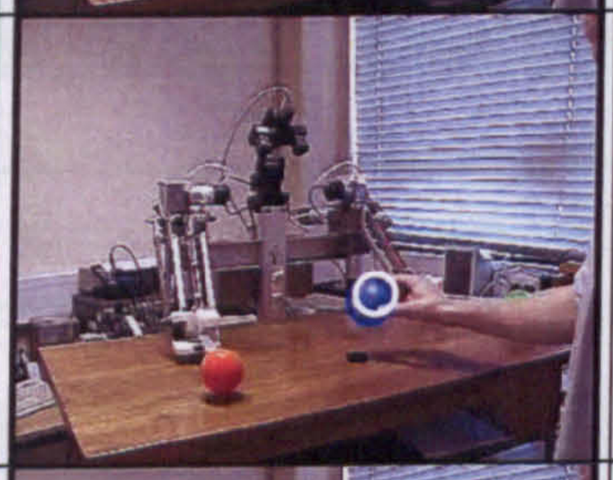
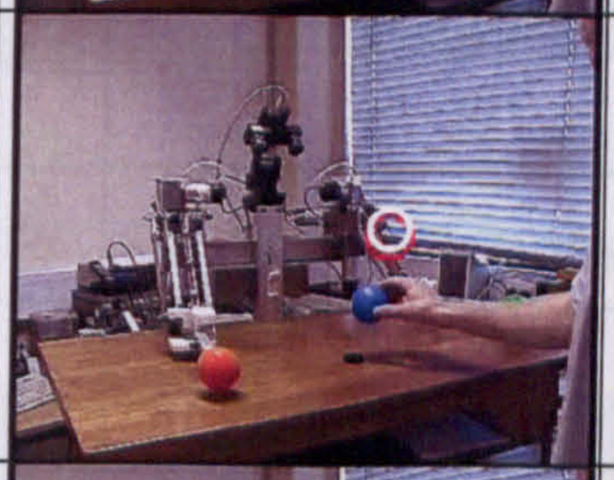
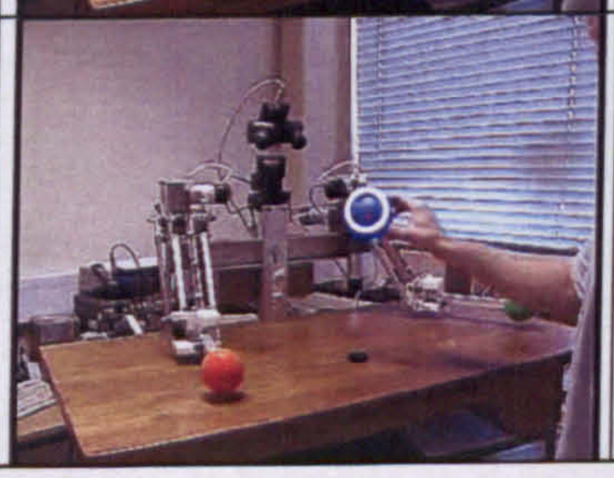
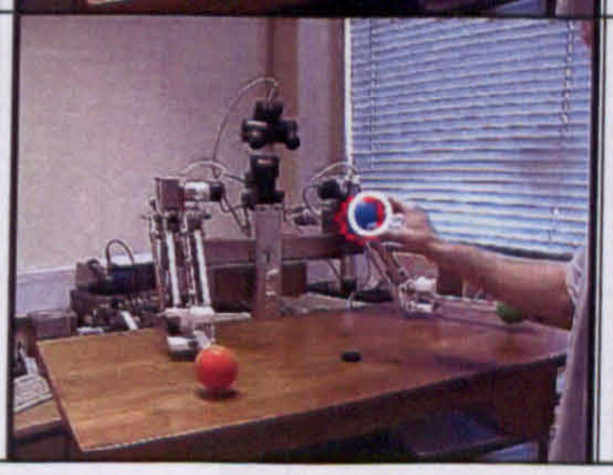
Frame #	Hybrid	SOK	KAMS
12			
60			
114			
120			

Figure 4.12. Three hybrid mean-shift/particle filtering algorithms track a hand-held ball.

Accuracy

Hybrid completed the sequence of Figure 4.13a with a mean error of 6.32 pixels, but failed around frame 20 when noise and clutter increased. SOK completed Figures 4.13a and 4.13b with mean errors of 5.66 and 9.60 pixels, but failed thereafter. Only KAMS managed to track through all the sequences summarised in Figure 5.6, with mean errors of 6.94, 8.9294, 14.723 and 17.303 pixels. While the algorithms produced similar levels of accuracy (where comparable data is available), KAMS is noticeably more robust. Note also that Hybrid required 100 particles and KAMS only 40. KAMS manages its particles more efficiently and so needs significantly fewer. The results are summarized in Table 4.4.

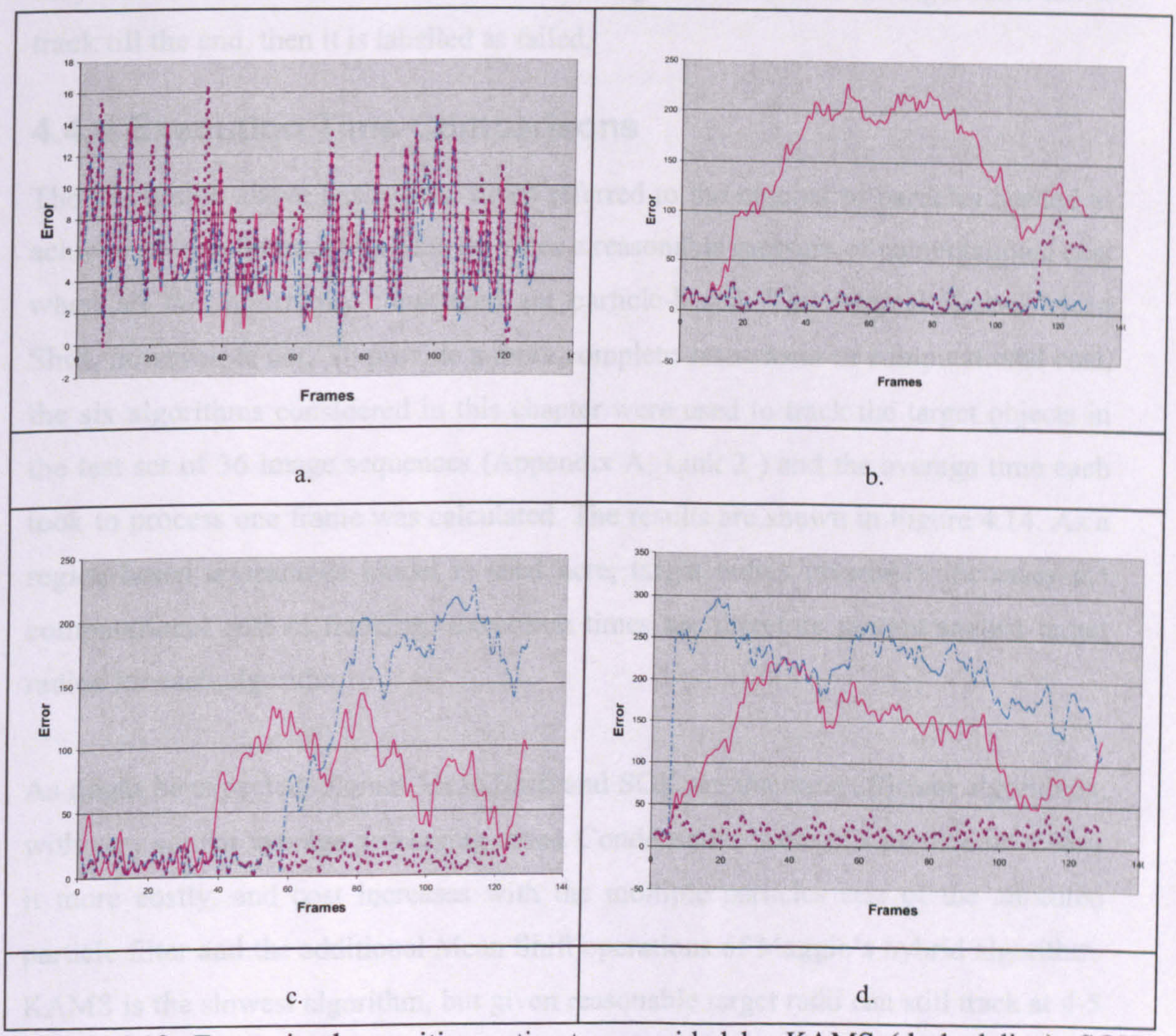


Figure 4.13. Errors in the position estimates provided by KAMS (dashed line), SOK (light line) and Maggio and Cavallaro's hybrid (dark line) algorithms, when applied to the image sequences summarised in Figure 4.6.

	Hybrid (Mean Error)	**SOK** (Mean Error)	**KAMS** (Mean Error)
Video 4.6a	6.32	5.66	6.94
Video 4.6b	Failed	9.6	8.9294
Video 4.6c	Failed	Failed	14.723
Video 4.6d	Failed	Failed	17.303

Table 4.4: Table showing mean error in pixels, of Hybrid, SOK and KAMS, while they track a ball in the 4 videos with increasing noise levels. If the algorithms fail to track till the end, then it is labelled as failed.

4.4.4 Execution Time Comparisons

The discussion above has several times referred to the number of particles needed to achieve successful tracking. This provides a reasonable measure of computational cost when all the algorithms considered are particle-based. The original Kernel Mean Shift, however, is not, To provide a more complete assessment of computational cost, the six algorithms considered in this chapter were used to track the target objects in the test set of 36 image sequences (Appendix A, Link 2) and the average time each took to process one frame was calculated. The results are shown in Figure 4.14. As a region-based appearance model is used here, target radius invariably increases the computational cost of tracking. Execution times are therefore plotted against target radius for each algorithm.

As might be expected, Kernel Mean Shift and SOK are the most efficient algorithms, with very similar average processing times. Condensation's use of a particle set makes it more costly, and cost increases with the multiple particles sets of the anncaled particle filter and the additional Mean Shift operations of Maggio's hybrid algorithm. KAMS is the slowest algorithm, but given reasonable target radii can still track at 4-5 frames/second, even if four annealing stages are used

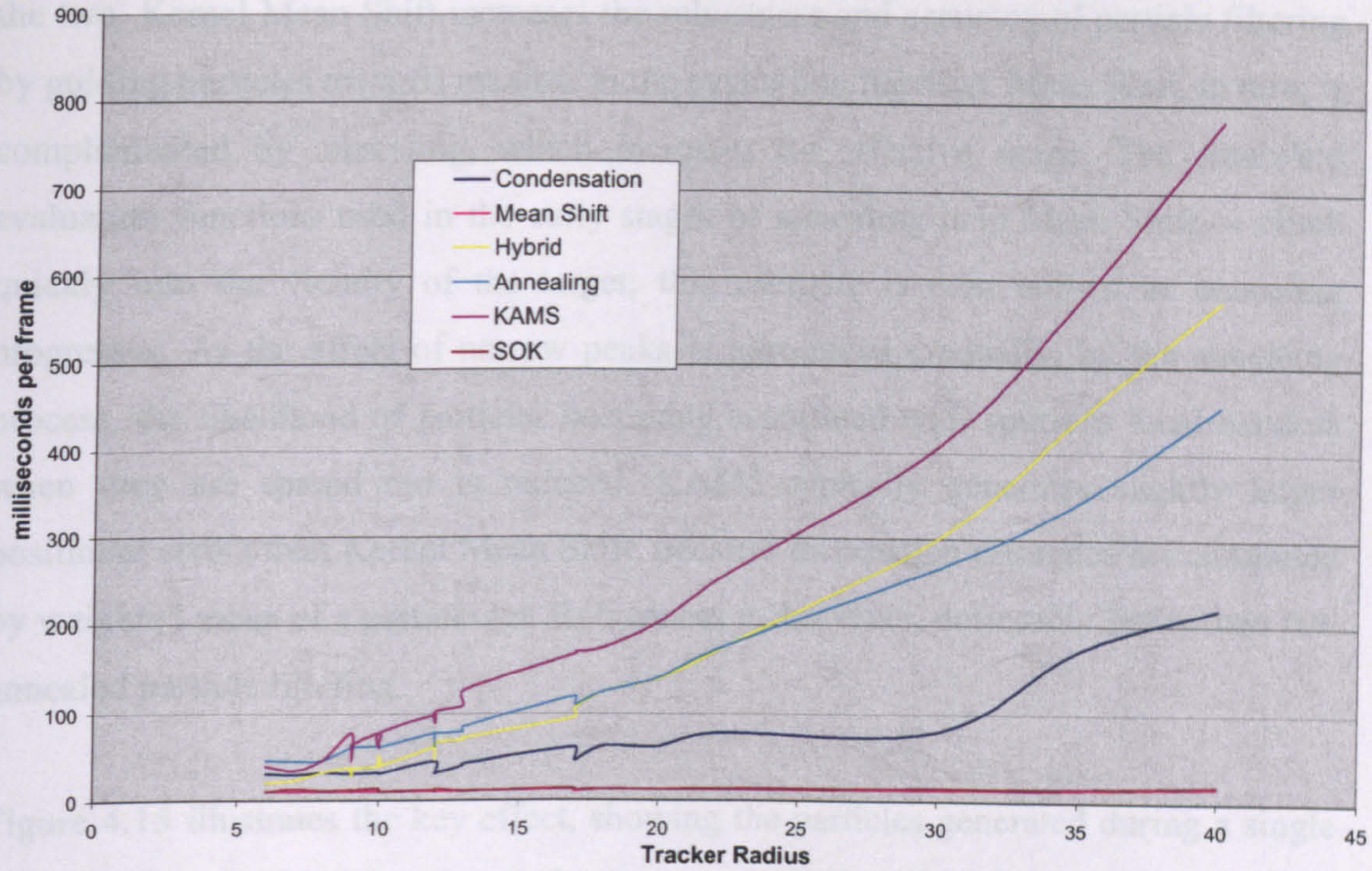


Figure 4.14: Time in milliseconds to process one frame on the average for algorithms in increasing order of tracked object radius.

4.5 Discussion

The Kernel Mean Shift and annealed particle filtering algorithms each have their strengths and weaknesses. Kernel Mean Shift explicitly seeks a maximum in the evaluation function, and so can provide accurate tracking, but lacks robustness when targets move quickly. Annealed particle filtering allows particles to be spread further apart than earlier particle filters, and so can deal with high speed motion. Increased spread can, however, introduce errors into the reported target state. There is no guarantee that any particle will lie on the true maximum, and the standard weighted mean estimate will include measurements made some way from the true target. The stochastic nature of the particle filter also means that, unless an unfeasibly large particle set is used, it is possible for all the particles to miss the target, leading to robustness problems.

The experiments reported here show that combining Kernel Mean Shift and annealed particle filter tracking gives an algorithm that is more robust than both of its components, with precision comparable to Kernel Mean Shift; the more accurate of

the two. Kernel Mean Shift increases the robustness and accuracy of particle filtering by guiding particles towards maxima in the evaluation function. Mean Shift, in turn, is complemented by annealing, which increases its effective range. The smoothed evaluation functions used in the early stages of annealing help Mean Shift to climb quickly into the vicinity of the target; this estimate is then refined as annealing progresses. As the effect of narrow peaks is introduced gradually, by the annealing process, the likelihood of particles becoming associated with spurious local maxima when they are spread out is reduced. KAMS typically generates slightly larger positional errors than Kernel Mean Shift, because its position estimates are computed by weighted mean of a particle set. Robustness is, however, noticeably better than raw annealed particle filtering.

Figure 4.15 illustrates the key effect, showing the particles generated during a single annealing run in KAMS. A larger than necessary (200) particle set is used to clarify the operation of the algorithm. Each row shows the two particle sets created for a single value of M. The left image shows the particles after addition of Gaussian noise, the right after application of Kernel Mean Shift. Note the alternating expansion and contraction of the particle set; KAMS particles explore a sizeable area of the search space, while ensuring they each lay on, or very close to, a maximum value.

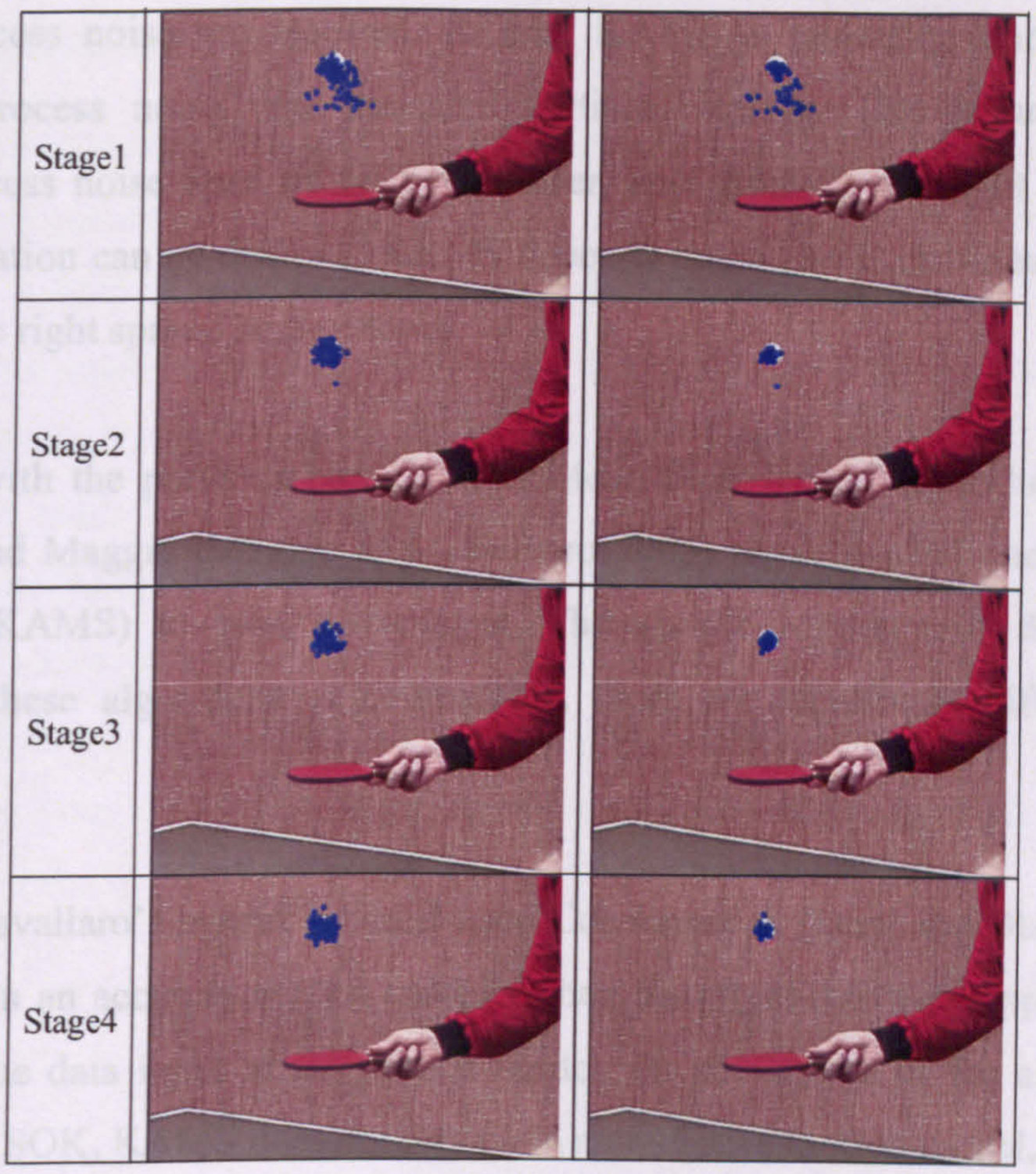


Figure 4.15. Dispersal and clustering of particles during an annealing run in KAMS. See text for details.

Previous work on Mean Shift/particle filter hybrid trackers (Maggio and Cavallaro 2005), (Shan et al 2007) and (Wang et al 2007) has noted that the inclusion of the Mean Shift step reduces the number of particles required for successful tracking. This result has been replicated here; KAMS typically needs only 20% of the particles required by raw annealed particle filtering. Combination with a Kernel Mean Shift also reduces the number of annealing stages required. Experience suggests that the four annealing stages used by Deutscher (Deutscher et al 2000) can be safely reduced to three in KAMS, and that two stages may suffice.

Underlying the notion of Kernel Annealed Mean Shift tracking is the hypothesis that combining Mean Shift and annealing will allow the annealed particle filter's process noise to be increased, increasing the effective search of the algorithm and reducing the need for a potentially inaccurate motion model. The available evidence suggests that this is true - KAMS is more robust than pure annealed particle filtering when high

levels of process noise are required. In fact, KAMS is generally more robust to changes in process noise than annealed particle filtering. This is an important property. Process noise must be set by the user, and choosing the right value for a particular situation can be difficult. KAMS depends much less than annealed particle filtering on the right spread being chosen.

Comparison with the previous particle filter/Mean Shift hybrid algorithms of SOK (Chapter 3) and Maggio (Maggio and Cavallaro 2005) also shows Kernel Annealed Mean Shift (KAMS) to have advantages. Though the accuracy of the tracking provided by these algorithms is comparable, there are significant differences in robustness.

Maggio and Cavallaro's hybrid is based upon Condensation (Isard and Blake 1998a), and so relies on an accurate motion model. When the target moves unexpectedly, or noise affects the data input to the motion model, the robustness of the algorithm is impaired. Like SOK, KAMS does not rely upon a predictive motion model, but begins its search from the last estimated location of the target. For this strategy to be effective, the tracker's search area must be widened. SOK employs a large, fixed search pattern which does not reflect the underlying evaluation function. As a result, SOK is more likely to become trapped on a local extrema. KAMS also uses a large search area, created by the process noise step in the particle filter. However, KAMS' Mean Shift search clusters its particles around maxima in the evaluation function, with the annealing process reducing the likelihood of the tracker becoming caught on spurious, narrow peaks. KAMS' effective management of its particle set also allows it to track successfully using fewer particles than raw annealed particle filtering.

KAMS' improved performance does, however, come at the price of increased processing time. The algorithm's use of multiple annealing stages, each of which employs multiple Mean Shift operations, makes it the slowest to process a frame, with processing time increasing with target size. Tracking at 5 frames/second can still be achieved, however, if target radii are less than 20 pixels. KAMS is efficient enough for real-time deployment in many applications.

4.6 Conclusion

We have proposed a novel tracking algorithm, Kernel Annealed Mean Shift tracking (KAMS), which combines Kernel Mean Shift with the annealed particle filter. The key feature of KAMS is its effective exploitation of the natural tension between the particle dispersal caused by the process noise of the particle filter and the clustering performed by Kernel Mean Shift. The KAMS algorithm has been applied to a variety of artificial and real image sequences and found to have performance and efficiency advantages over both pure Kernel Mean Shift and annealed particle filtering algorithms and existing particle filter/Mean Shift hybrid trackers. Combining Kernel Mean Shift and annealed particle filter tracking gives an algorithm that is more robust than both of its components, with precision comparable to Kernel Mean Shift; the more accurate of the two.

Chapter 5: Multiple Target Tracking Using Kernel Annealed Mean Shift Tracking

5.1 Introduction

Much research has been carried out on single target tracking algorithms, and an introduction to the issues and common approaches has been given in chapter 2. Two novel single target tracking algorithms were presented in chapter 3 and chapter 4, named SOK and KAMS respectively, and it was shown experimentally that KAMS outperforms all the algorithms that were evaluated against it in chapter 4 both in robustness and accuracy.

Although single target trackers are very important, most real world applications employing visual tracking require tracking of multiple objects. Effective surveillance requires multiple people and/or vehicles to be tracked to determine presence and detect events (French et al 2007, Suyu et al 2007, Chen et al 2007). Analysis of team sports like football involves tracking multiple players (Perš and Kovačič 2000). Gesture recognition can involve tracking multiple body parts of one or more individuals (Starner and Pentland 1995, Nickel and Stiefelhagen 2007). Medical applications like microscopic sample analysis may rely on analysis of the motion of multiple, similar organisms like blood cells or sperm.

When tracking single targets occlusion only arises from surrounding objects which are not of interest, during multi-target tracking other tracked objects may also occlude a given target. Care must also be taken to make sure that the right trackers are on the right targets after occlusions and collisions. Coalescence of multiple trackers onto a single target is a particular problem when the tracked targets are similar, for example when analyzing team games, or tracking animals like ants. Moving from single to multiple target tracking may also mean a lot more computing power is required to achieve the same performance levels. New challenges arise when there are multiple objects of interest. In many circumstances, e.g. when tracking people through a mall, the tracker must handle new objects entering and other objects leaving the scene. In some applications, e.g. when tracking cells, one target may divide into two objects of similar type.

Multiple target tracking algorithms can be placed into two major categories. In the first each object is tracked using a separate single target tracker, and inter-object interactions are handled by some form of interaction handling mechanism which is introduced between these sets of trackers. This is there to handle possible occlusions and collisions. Other multi-target trackers may maintain a joint state in which information about all the trackers is made explicit in a single representation.

In what follows the single target KAMS algorithm is extended to support multi-target tracking using a single KAMS tracker for each tracked object and an interaction filter to handle the occlusions and interactions. This puts the multiple target tracking version of KAMS in the first of the two categories of multi-target tracking algorithms discussed above. Attention is focused on the integration of Khan's (Khan et al 2004) interaction filter into the annealing process used by KAMS. Khan's (Khan et. al. 2004) method is one of the most successful and widely adopted multi-target tracking algorithms, but is limited to simply reducing the weight of particles associated with interacting targets. The use of multiple evaluation functions during KAMS' annealing process allows for a wider range of more informed responses. Khan's approach is described in section 5.2., before two variations of multi-target KAMS are presented in section 5.3. Both multi-target KAMS algorithms are then experimentally evaluated against each other and Khan's (Khan et al 2004) algorithm in section 5.4. Finally, conclusions are drawn in section 5.5.

5.2 Multiple Targets and Khan et al's Interaction Filter

Some of the best known and most promising work on multiple target tracking was done by Khan (Khan et al 2004). They use a joint state MCMC tracking algorithm in which every particle stores the state of every target. Their basic hypothesis was that objects in close proximity interact with each other and should influence each other's behaviour.

The multiple target tracking problem can be expressed as a Bayes filter in which posterior distribution $p(x_t \mid z^t)$ is recursively updated over the joint state of all targets $\{x_{it} \mid i \in 1...n\}$ given all observations $z^t = \{z_1...z_t\}$ up to and including observation at time t. The following is a representation of a particle filter with the assumption that targets do not interact with each other.

$$p(x_t \mid z^t) = kp(z_t \mid x_t) \int_{x_{t-1}} p(x_t \mid x_{t-1}) p(x_{t-1} \mid z^{t-1}) \qquad (5.1)$$

$p(z_t \mid x_t)$ is the likelihood expressed as the measurement model. z_t is the measurement given the state x_t at time t. $p(x_t \mid x_{t-1})$ represents the motion model which predicts the state x_t at time t given previous state x_{t-1}. Khan et. al. assume that the likelihood $p(z_t \mid x_t)$ factors across targets as $p(z_t \mid x_t) = \prod_{i=1}^{n} p(z_{it} \mid x_{it})$.

When targets do not interact, the exact Bayes filter can be approximated using multiple single target particle filters, so the motion model $p(x_t \mid x_{t-1})$ is factored as $\prod_{i=1} p(x_{it} \mid x_{i,t-1})$. In the real world targets do interact, collide and overlap each other. Hence using multiple single trackers with the assumption that they do not influence each other may not always be true, and the system will fail when targets interact. Interaction may be as simple as two targets coming into close proximity. The major failure mode observed is that multiple trackers lock onto the same object.

To handle these interactions Khan et al proposed a more capable motion model based on Markov random fields (MRF). They model the interaction between objects using a graph-based MRF constructed on the fly, dynamically, at each time step. A MRF is a graph (V,E) with undirected edges between nodes. The joint probability is factored as a product of local potential functions at each node and interactions are defined on neighbourhood cliques. The most commonly used form is a pair wise MRF, in which cliques pairs of nodes (objects) that are connected by undirected arc. Khan (Khan et al 2004) assumes the following pair wise MRF form where $\psi(x_{it}, x_{jt})$ are pair wise interaction potentials.

$$p(x_t \mid x_{t-1}) \propto \prod_i p(x_{it} \mid x_{i(t-1)}) \prod_{ij \in E} \psi(x_{it}, x_{jt}) \tag{5.2}$$

Khan (Khan et al 2004) express the interaction potential in terms of Gibbs distribution

$$\psi(x_{it}, x_{jt}) \propto \exp(-g(x_{it}, x_{jt})) \tag{5.3}$$

where $g(x_{it}, x_{jt})$ is a penalty function which depends (in Khan's case) entirely on the number of overlapping pixels of the bounding boxes of two tracked objects i and j in the pair. This increases with area of overlap and is minimal when the two objects are distinct. Khan et al compute the weight of the particles concerned based on this penalty function

$$\pi_t^{(s)} = \prod_{i=1}^{n} p(z_{it} \mid x_{it}^{(s)}) \prod_{ij \in E} \psi(x_{it}^{(s)}, x_{jt}^{(s)}) \tag{5.4}$$

where $\pi_t^{(s)}$ is the weight at time t from sample set s. So each particle's weight also depends on the amount of interaction with other objects being tracked. If a given particle's state estimate accurate describes one of its targets and so has high weight, but that state also violates the temporal space of another tracked object, then the penalty function reduces the weight of the particle, making it less likely to be projected to the next particle set.

5.3 Multi-target Kernel Annealed Mean Shift Tracking

The multi-target Kernel Annealed Mean Shift tracker (MKAMS) is an extension to the single target tracking algorithm (KAMS) discussed in chapter 4. Annealed particle filtering and Mean Shift are major components of KAMS and are discussed in section 4.2 and section 2.3.4 respectively.

While tracking multiple targets an interaction handling mechanism has to be introduced to guard against failures caused by many trackers acquiring the same target at the same time. This is commonly known as target 'coalescence'. Khan (Khan et al 2004) used an interaction filter based on a Markov random field to handle scenarios in which objects interact. However, in Khan's algorithm particles are projected randomly, and their weights computed once and then adjusted according to a penalty function as discussed in section 5.2. KAMS on the other hand incorporates multiple annealing stages, in each of which particles are projected, Mean Shifted towards the best match and weighted by some evaluation function. These evaluation functions are related by a smoothing operator (Chapter 4). After each of the 4 stages used, the tracker's estimates of the position of each target may change. A multi-target KAMS algorithm must check after each individual stage if one target has violated another object's boundaries.

It is hoped that each stage in the annealing process used in KAMS (and MKAMS) will improve the tracker's estimate of target position. The estimates obtained at the end of each stage may therefore be thought of as successive approximations to the true position. If at some point in the annealing process an object starts to violate other objects' boundaries, the KAMS approach raises the possibility of retreating to an estimate provided by a previous stage. Keeping the best estimate obtained without occlusion seems to be a logical choice. In Khan et al's algorithm, in contrast, a target's position estimate is reset to its position in the previous frame when the interaction filter fires. In effect all data from the current image is ignored.

Given the notion of a multiple stage interaction filter two variations of MKAMS present themselves. One approach would be to apply an interaction filter at each stage and, as soon as a stage indicates occlusion, restore the tracker's position to the estimate obtained at the previous successful stage and move to the next object. This algorithm makes very conservative use of the annealing stages, only proceeding until interactions begin to appear.

Alternatively, the annealing stage showing target interaction could be ignored. This could be achieved by restoring the target position to the estimate obtained in the previous stage, then projecting that onto the next stage hoping to get to a better position without occlusions. The effect is to seek estimates of target state in the closest approximation to the desired evaluation function that does not show target interactions. The first approach we call "Simple MKAMS", and the latter we name "Complex MKAMS".

The MKAMS algorithm, in both variations, is stated in figure 5.1.

Kernel Annealed Mean Shift Tracking Algorithm:

Acquire frame at time t_k, having a set $S_{k,M}$ of N unweighted particles from the previous time step, for 'x' number of objects.

Set x = number of objects tracked

Define variable to store information for one object 'temp'

While(x >0) This loop will deal with an object at each iteration

{

Set weighting function index m = M

While (m>0) This loop will deal with multiple stages while considering One object

{

a. temp = current state of object x before processing this stage.

b. Assign each particle a weight $\pi_{k,M}^{(i)}$

c. Select N particles replacing the old particle set and add Gaussian noise: $s_{k,m-1}^{(i)} = s_{k,m}^{(i)} + B_m$

d. Apply kernel Mean Shift to each particle until the Bhattacharya distance between the model and image measured by the weighting function $w_{m-1}(Z_k, s_{k,m-1}^{(i)})$ becomes stable or minimum.

e. Work out the new position $p(x, y)$ for object x.

f. Check if new position of object x violates the airspace of other objects

i. If no,

- Continue

ii. If yes,

- restore all information back from temp
  - For MKAMS Simple version, break inner loop.
  - For MKAMS Complex version, Continue.

g. m = m-1

}

x = x - 1

}

Figure 5.1: Multi-Object KAMS algorithm, showing both simple and complex MKAMS.

The algorithm described in figure 5.1,

- saves the initial state of an object (which includes its particle set, initial position and other parameters) before each annealing stage,
- undergoes that annealing stage
- checks if, in its new position, the specific object occludes other objects (i.e. if it triggers Khan's interaction filter)
- if it does and Simple MKAMS is desired then than the parameters saved before that stage are restored and attention moves to the next object
- if it does and Complex MKAMS is desired then the parameters saved at the previous successful stage are restored and the target projected onto the next annealing stage, where the process is repeated.
- if no interaction is detected, annealing proceeds as in KAMS.

The two MKAMS algorithms presented above can be expected to have different strengths and weaknesses. Complex MKAMS proceeds as far through the annealing process as possible, producing estimates from the best evaluation function that it can. To achieve this, however, it skips any annealing stages in which interaction is detected. This reduces the effect of the annealing process, and may make the algorithm prone to becoming caught on local maxima. This is due to the fact that each smoothing stage may shift the detected maxima away from the true global maxima gradually and towards the higher clutter density (figure 5.2). If we study figure 5.2 we note that the actual distribution is in stage (a). As we smooth out the distribution, the peak shifts away from the true peak in each smoothed stage. This effect is dependant on the position of the clutter in the area; if most of the clutter is on the right side of the global maxima as in this case, than smoothing will shift the centre towards the right and vice versa. In the example shown in figure 5.2, skipping stages b and c will cause the tracker to move to the incorrect local maxima to the right of its initial position. If all four annealing stages are used, however, the tracker will move to the left.

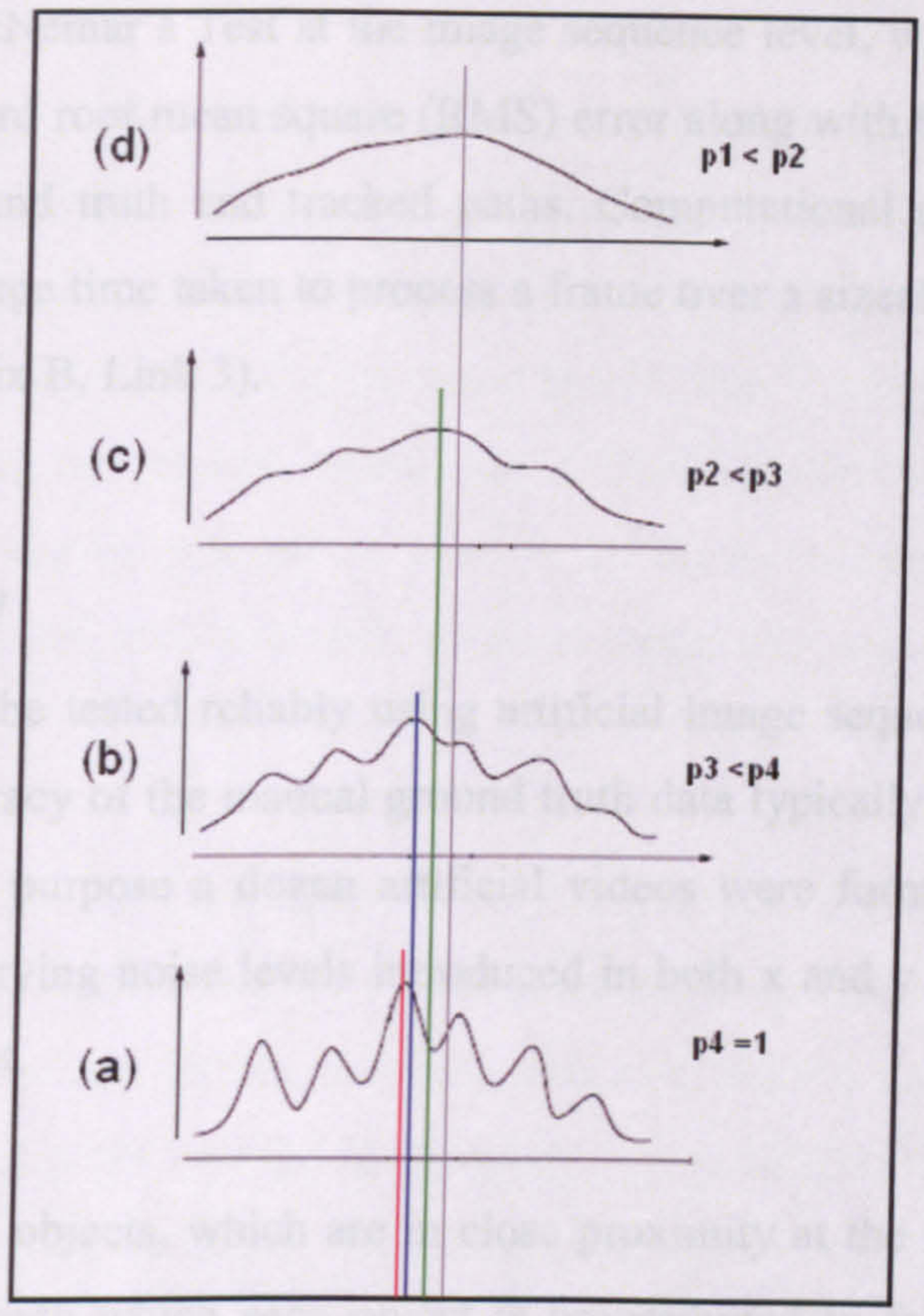


Figure 5.2: Illustrating the shift in maxima as the evaluation function is smoothed.

Simple MKAMS can be expected to produce estimates using smoother evaluation functions that are more distant approximations to the desired function. This might be expected to introduce some degree of positional error into the tracker's output. The annealing process used to identify them is, however, complete, reducing the danger of distraction by background objects.

Both versions of the MKAMS algorithms have been implemented and compared against each other and Khan's MCMC filter. Results are given in section 5.4.

5.4 Experimental Evaluation

Real world applications of tracking algorithms usually require tracking multiple objects. To gauge different algorithms we must consider robustness, accuracy and computational cost.. To assess robustness we will measure sensitivity at the frame

level and apply McNemar's Test at the image sequence level, while for accuracy we compute the standard root mean square (RMS) error along with frame by frame error plots between ground truth and tracked paths. Computational cost is estimated by measuring the average time taken to process a frame over a sizeable set of (33) image sequences (Appendix B, Link 3).

5.4.1 Accuracy

Accuracy can only be tested reliably using artificial image sequences, as we cannot depend on the accuracy of the manual ground truth data typically extracted from real sequences. For this purpose a dozen artificial videos were formulated with known ground truth and varying noise levels introduced in both x and y coordinates at each frame for each object.

Each video shows 5 objects, which are in close proximity at the start of each video. There is a definite path which each object is programmed to follow in each video (Figure 5.3). Since in multiple target tracking added problems of collisions and occlusions occur, we start with a sequence showing a smooth trajectory of 5 objects and then add increasing amounts of random noise to each object, causing them to collide and occlude. This will gradually test not only the algorithms ability to track speedy targets with irregular motion, but also how well an algorithm copes with collisions and occlusions. Figure 5.3 gives the exact path, without any random noise added, for each object in the original video.

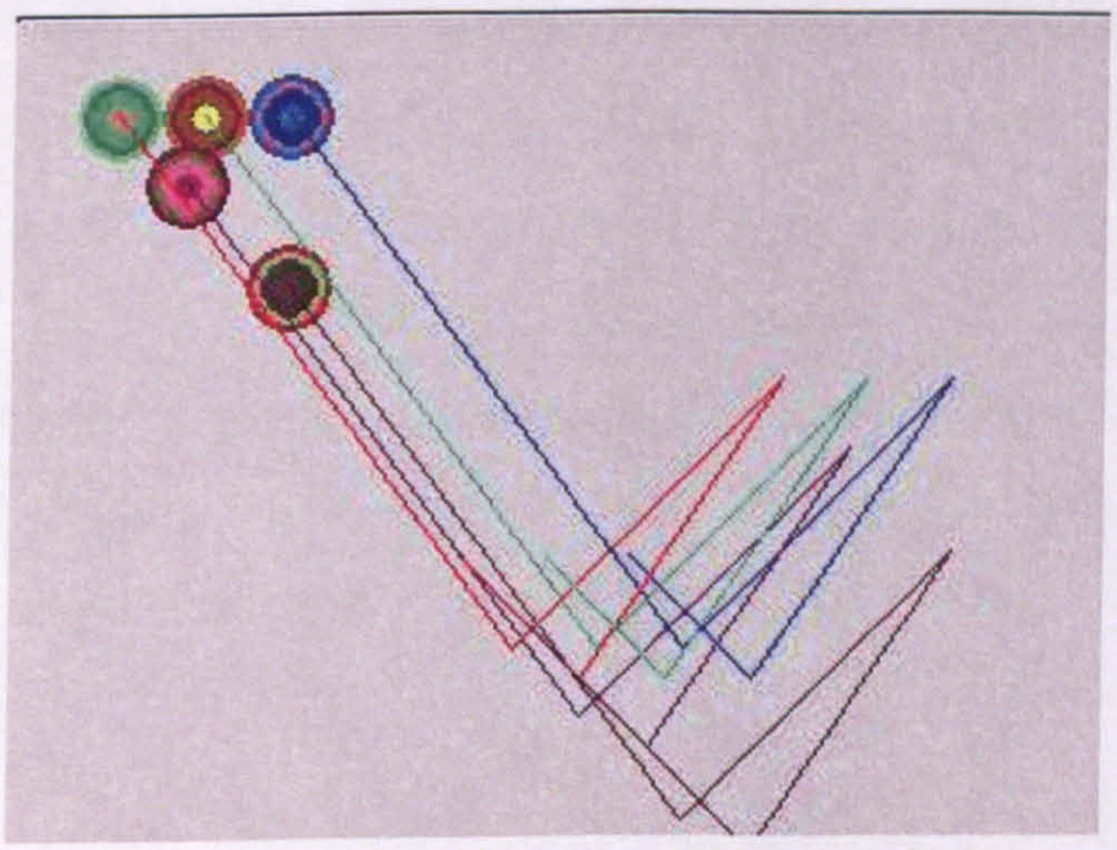
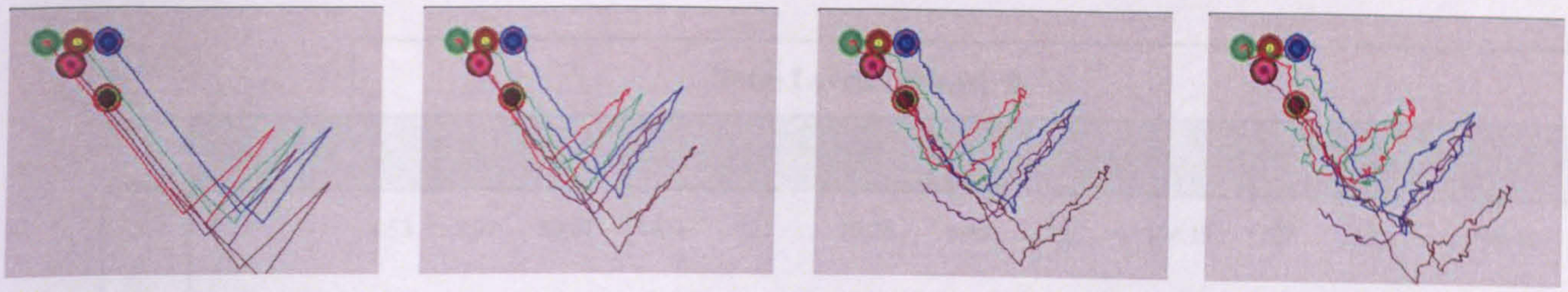


Figure 5.3: Original Video with no noise added. All original paths are drawn for all the five objects.

The only difference between each video is the level of noise introduced to the targets' x and y coordinates at each frame. Noise is generated by multiplying the output of a Gaussian random function with a constant. As we increase the constant value, the range gets bigger hence increasing the noise levels. The Gaussian function returns a real value by picking out random values from a Gaussian distribution of mean 0 (μ = 0) and standard deviation squared of 5 (σ^2 = 5). So by increasing the multiplying constant, and keeping the Gaussian random range the same, random noise is increased.

With zero added noise the occlusions are very minor and only partial, but collisions between trackers will occur as most of the objects touch each other during motion and the trackers jitter over their targets, colliding with each other occasionally. As noise is increased, the objects paths become increasingly random (figure 5.4) and the number of occlusions is bound to increase. An increase is seen in both partial and complete occlusions.



Noise 0x Noise 1x Noise 2x Noise 3x

Noise 4x Noise 5x Noise 6x Noise 7x

Noise 8x Noise 9x Noise 10x Noise 12x

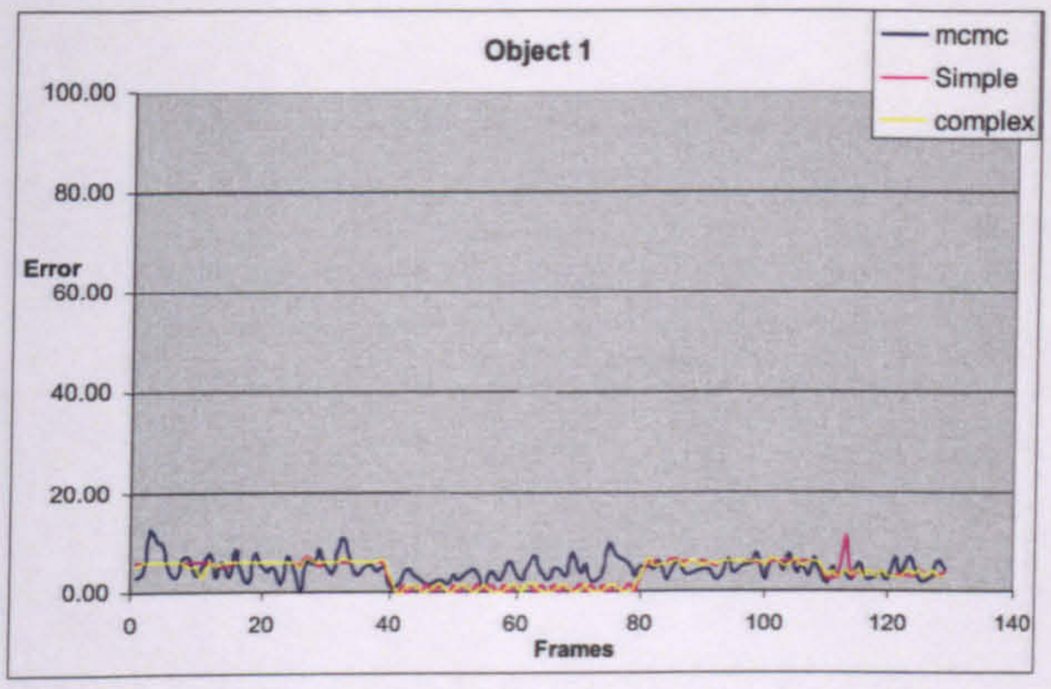
Figure 5.4: The effects of varying the random noise added to each object's path at each frame

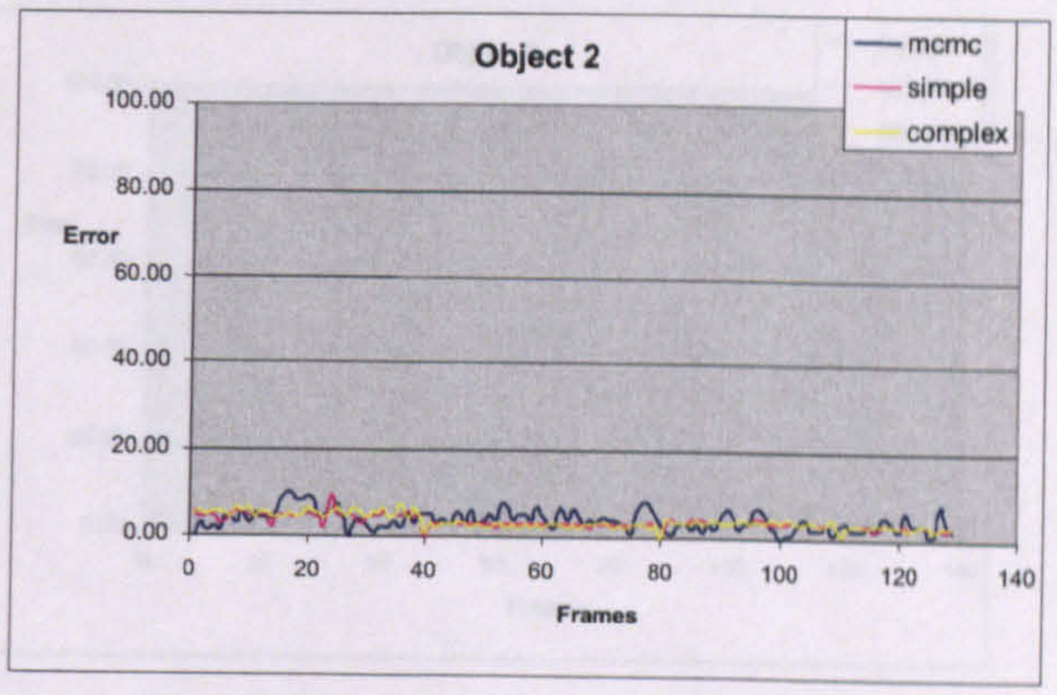
Objects in each video were tracked using Khan's (Khan et al 2004) MCMC and the two versions of MKAMS (complex and simple). Estimates of the position of each object from all the 3 algorithms were compared with ground truth and RMS errors were computed. Table 5.1 shows RMS errors associated with each object when tracked by each algorithm in each video. Here average RMS error for each individual object is reported for each video and for each algorithm, and the total accumulated average RMS error per frame for all the objects in each video is also reported. All distances are in pixels.

		Noise Levels (videos) →											
	Object #	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	10x	12x
MCMC	1	4.53	8.07	80.61	1631	727	19.26	6.85	23	104.15	109	110.7	158.94
	2	4.02	8.51	5.66	4.84	19.75	15.91	376	2210	169.34	1309	2822	459
	3	5.88	4.77	381.9	211	30.23	2026	2337	117	1458	3330	7.36	1333
	4	6.13	6.20	6.28	1403	7.74	2750	70	278	635	31.44	53.18	122
	5	3.70	3.94	4.20	5.11	7.03	6.41	7.45	8.26	1032	1676	11.97	440
	Average	4.85	6.30	95.75	651	158	964	559	527	679.81	1293	746.83	502
Simple MKAMS	1	3.97	5.60	4.61	6.07	6.23	7.21	8.24	9.52	94.19	97.16	77.59	120.31
	2	3.66	4.02	5.56	5.14	6.02	6.82	8.11	9.01	10.33	11.50	30.25	15.05
	3	4.03	4.55	6.24	5.95	6.53	7.83	9.08	10	10.55	11.36	54.50	15.43
	4	4.30	4.46	5.07	5.81	7.09	9.22	9.37	206	10.83	11.94	11.57	134.33
	5	5.04	4.83	5.23	6.17	6.38	7.03	8.76	9.73	9.67	11.17	12.97	30.36
	Average	4.20	4.69	5.34	5.83	6.45	7.62	8.71	49	27.11	28.63	37.38	63.10
Complex MKAMS	1	3.83	5.77	4.60	6.25	6.22	7.41	8.39	9.52	10.84	11.63	13.29	41.41
	2	3.82	4.43	5.34	5.56	6.24	6.93	8.05	9.50	10.48	48.72	12.90	14.99
	3	4.51	4.60	5.35	6.12	7.08	8.06	9.43	10	10.57	13.01	14.36	131.09
	4	4.32	4.14	4.88	6.35	6.88	8.31	8.83	73	9.70	11.44	12.29	32.62
	5	5.05	4.66	5.21	6.09	6.93	8.15	9.38	10	10.08	11.90	13.13	132.67
	Average	4.31	4.72	5.08	6.07	6.67	7.77	8.82	22.72	10.34	19.34	13.20	70.56

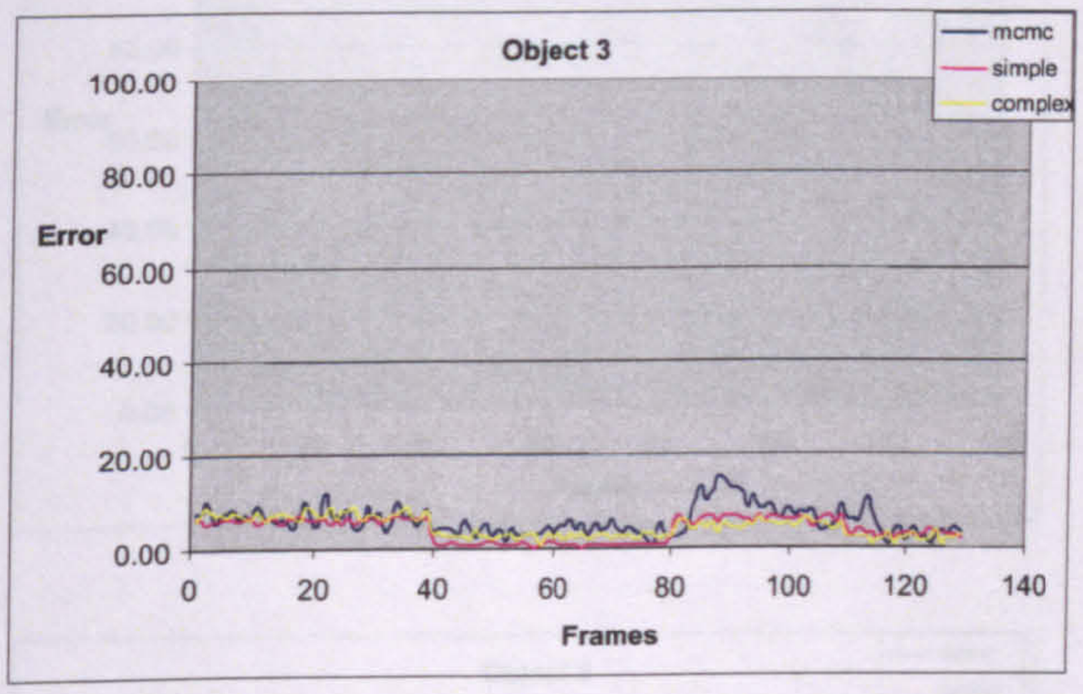
Table 5.1: RMS error of each object in all the tested videos, average error is also shown in the yellow fields.

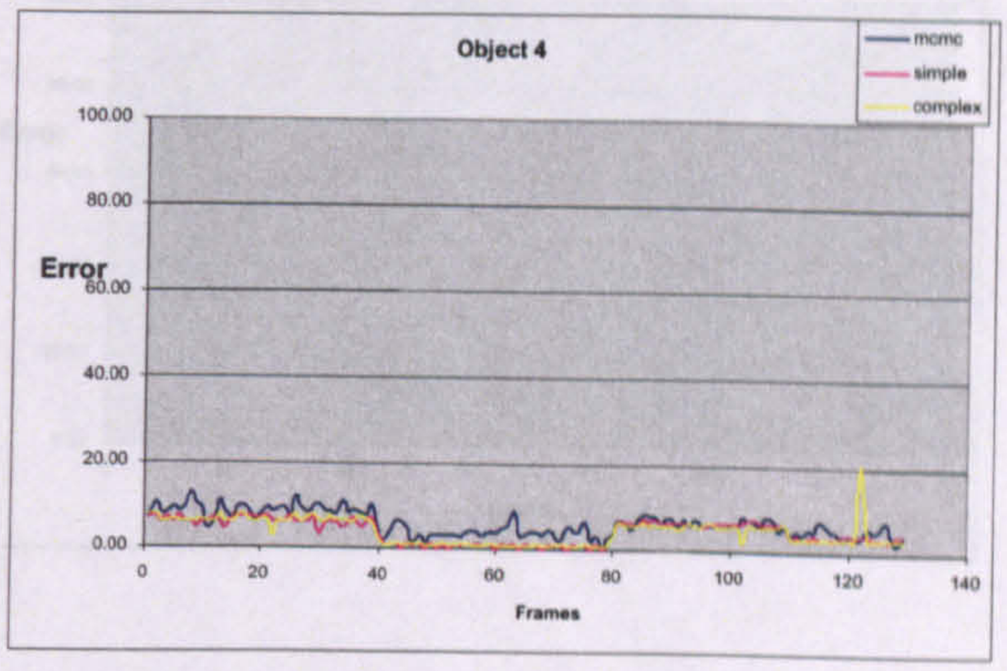
The individual object errors from the three algorithms in each video are plotted separately in figures 5.5 to 5.16.





Noise 0x





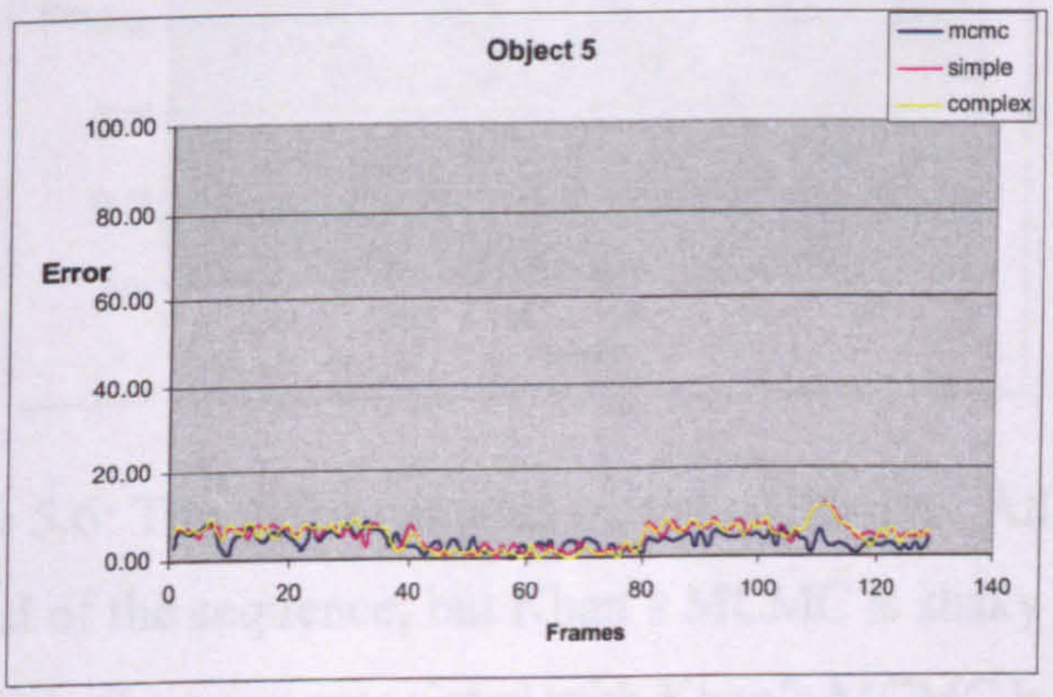
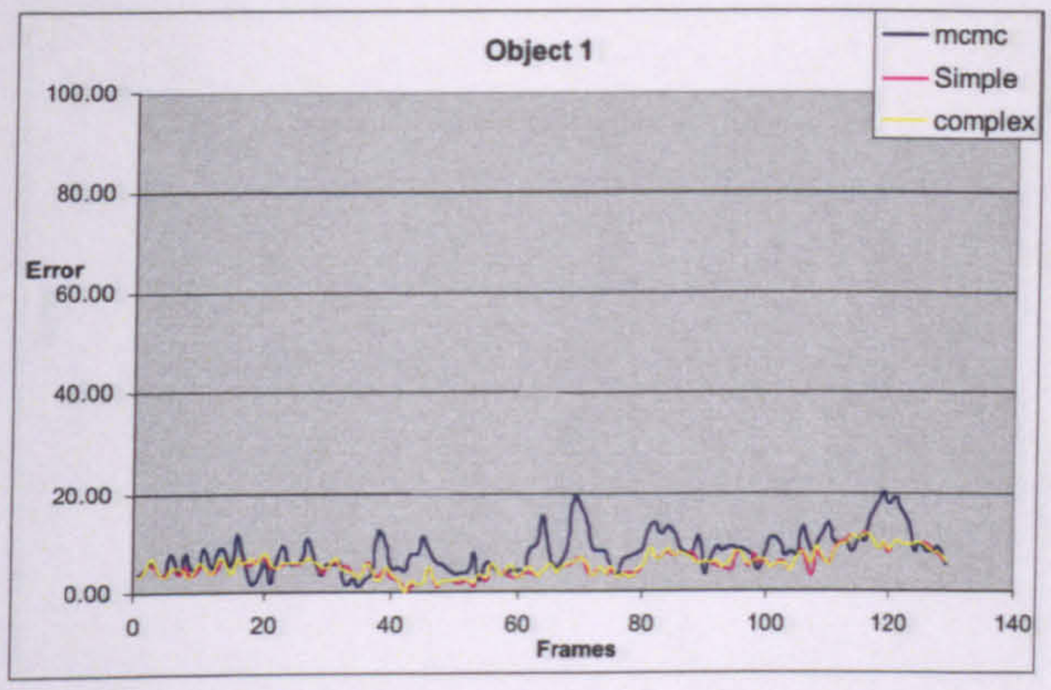
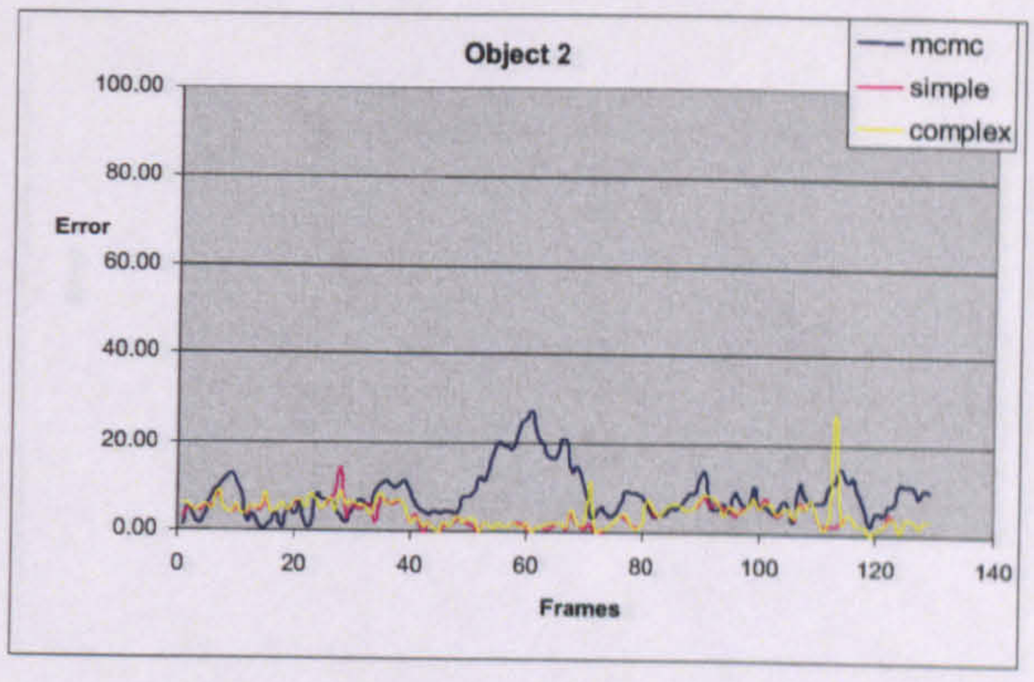
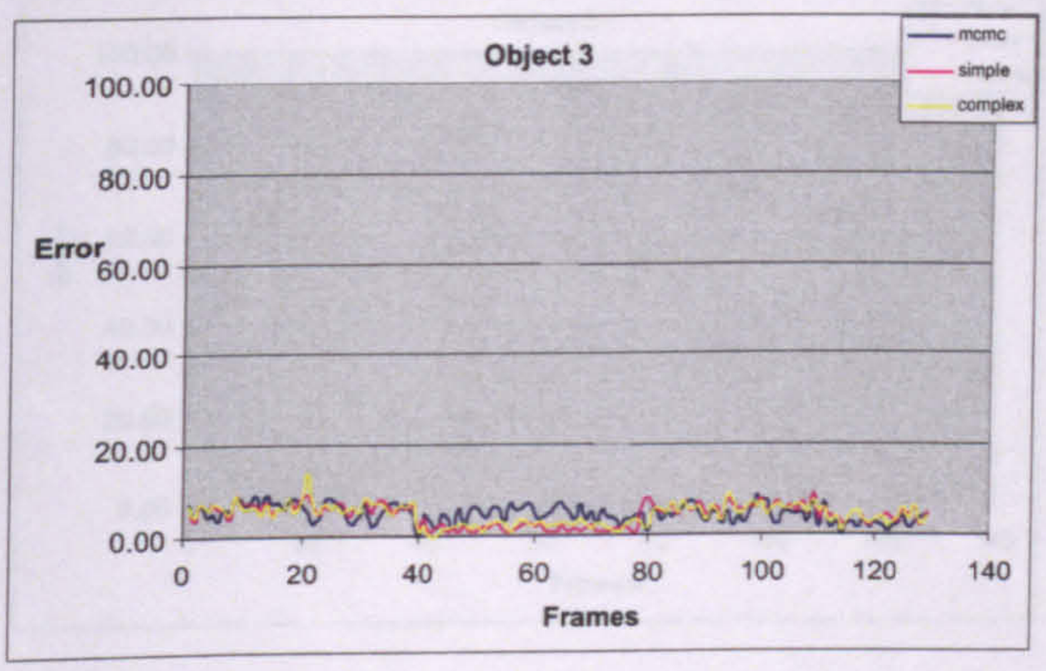


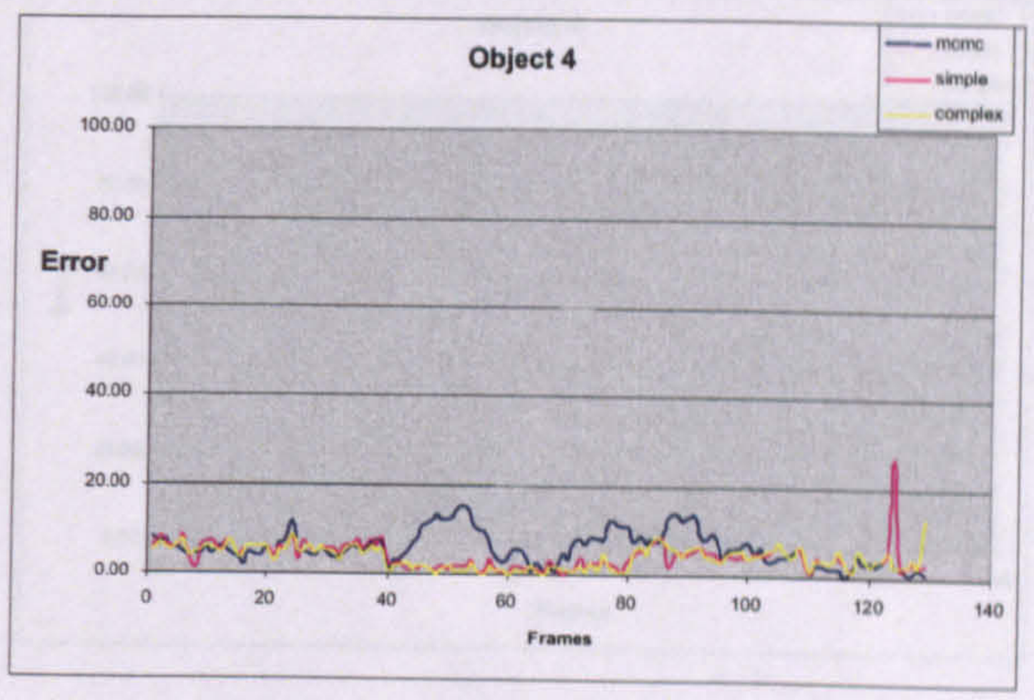
Figure 5.5: No noise was added to the five objects in this video. The occlusions are minimal and only partial. All the three algorithms successfully tracked the 5 objects to the end of the sequence.





Noise 1x





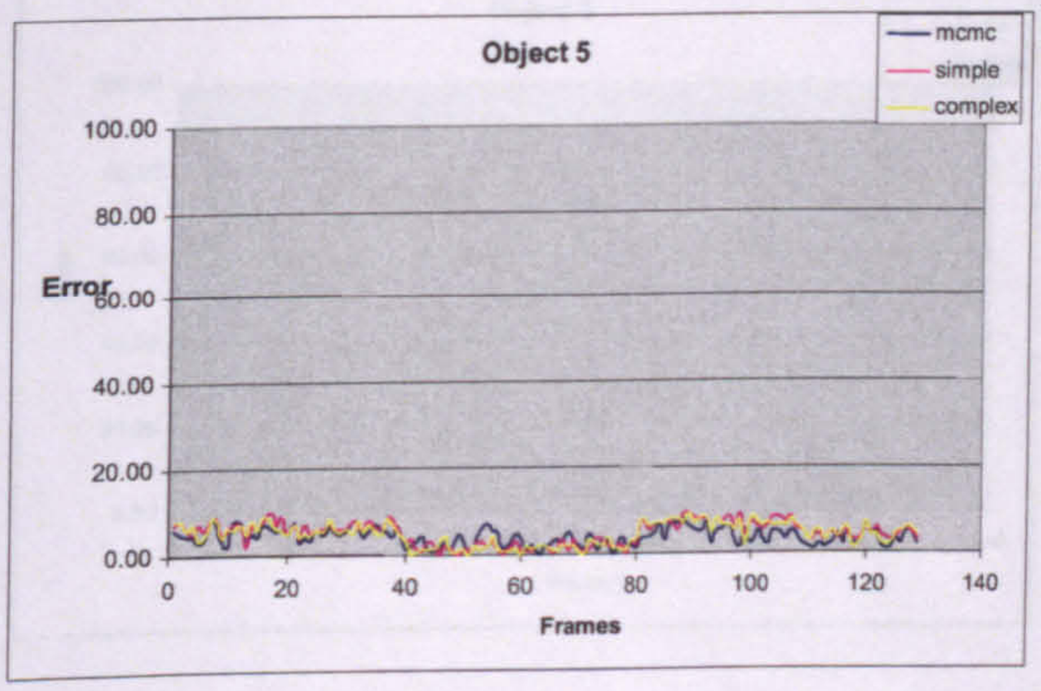


Figure 5.6: The video contains partial occlusions. All algorithms tracked the objects successfully to the end of the sequence, but Khan's MCMC is shaky while tracking objects 1, 2 and 4 .In several situations the error associated with Khan's MCMC is noticeably higher than that obtained from MKAMS.

Noise 2x

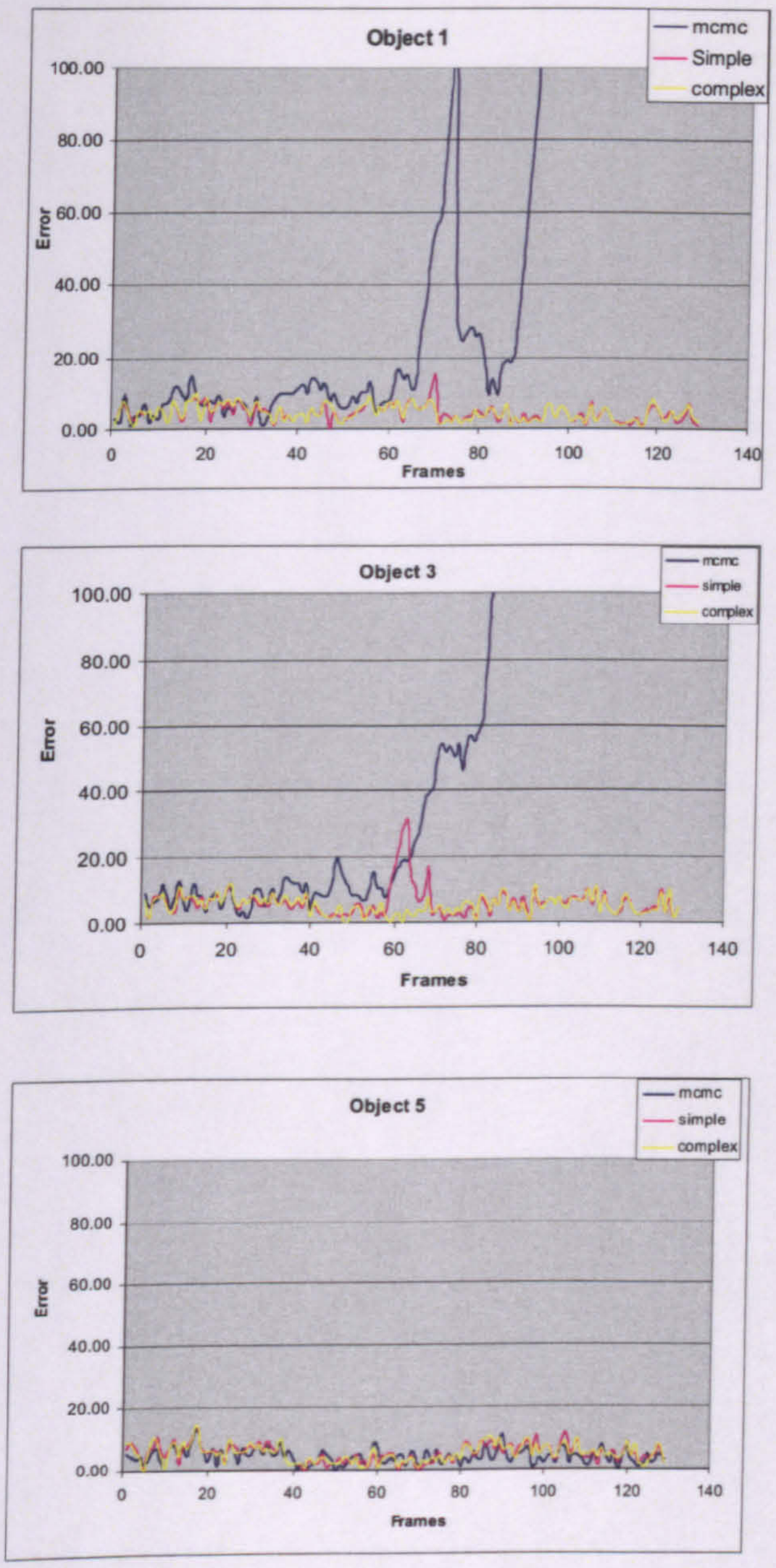
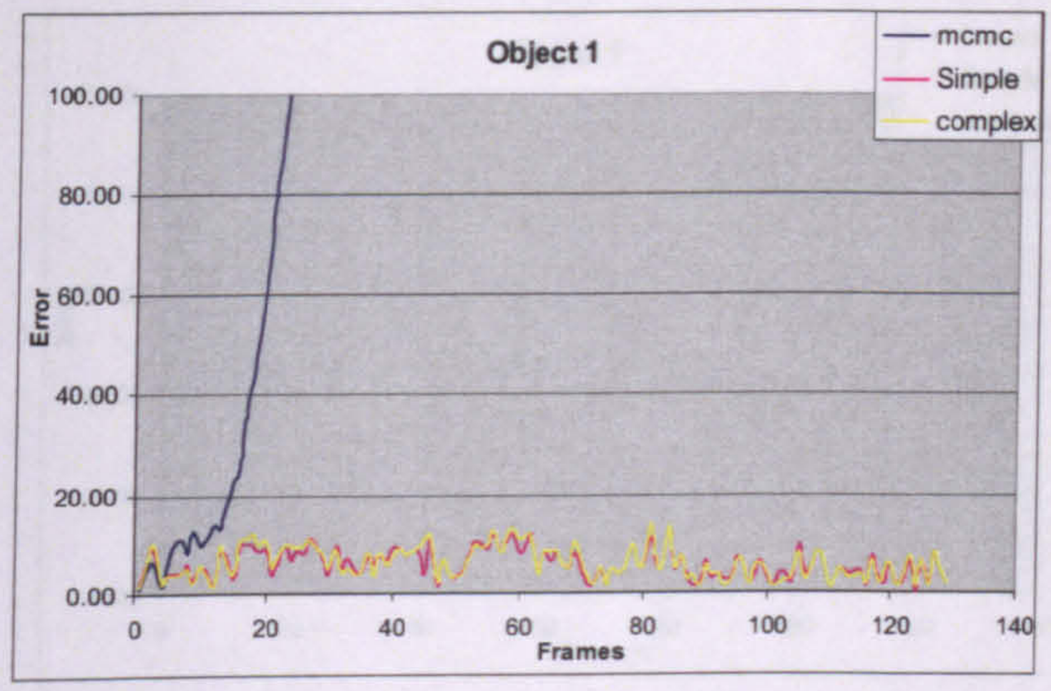
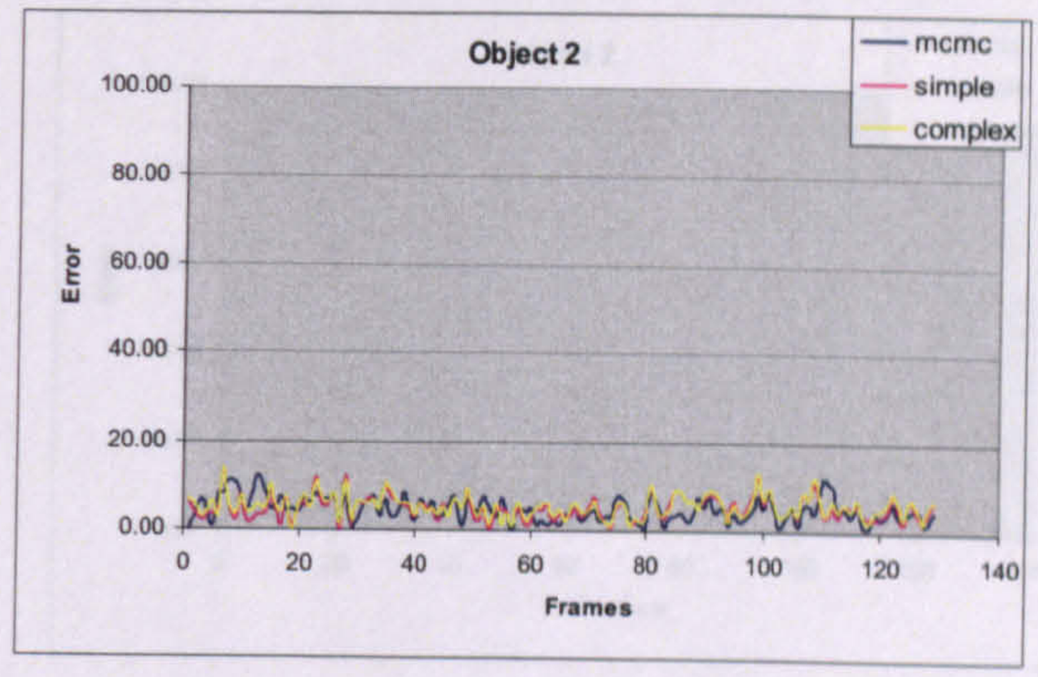
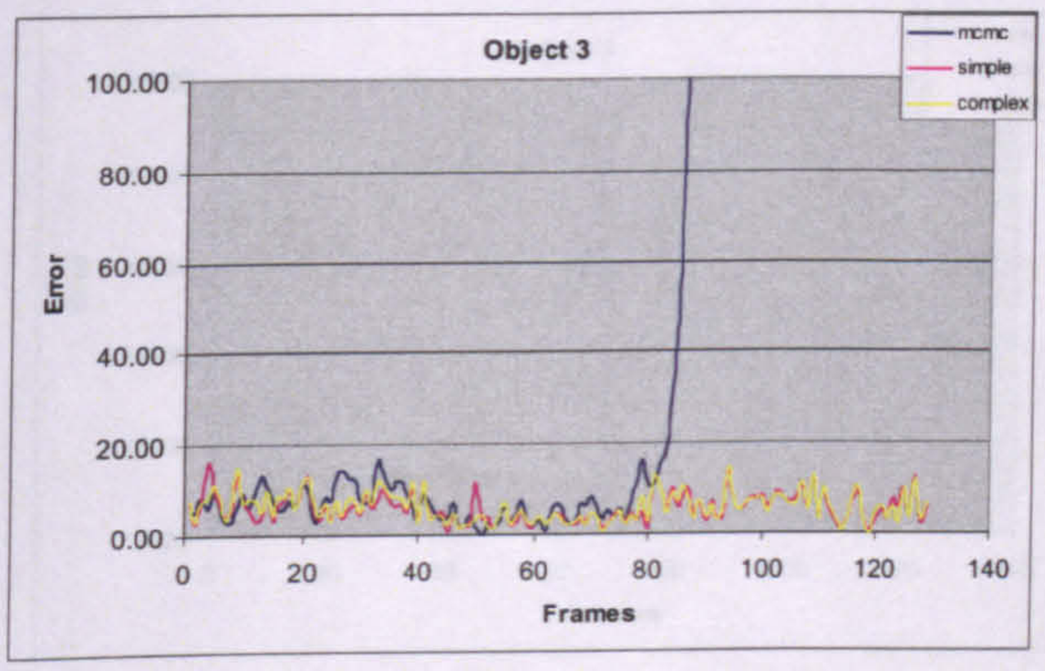


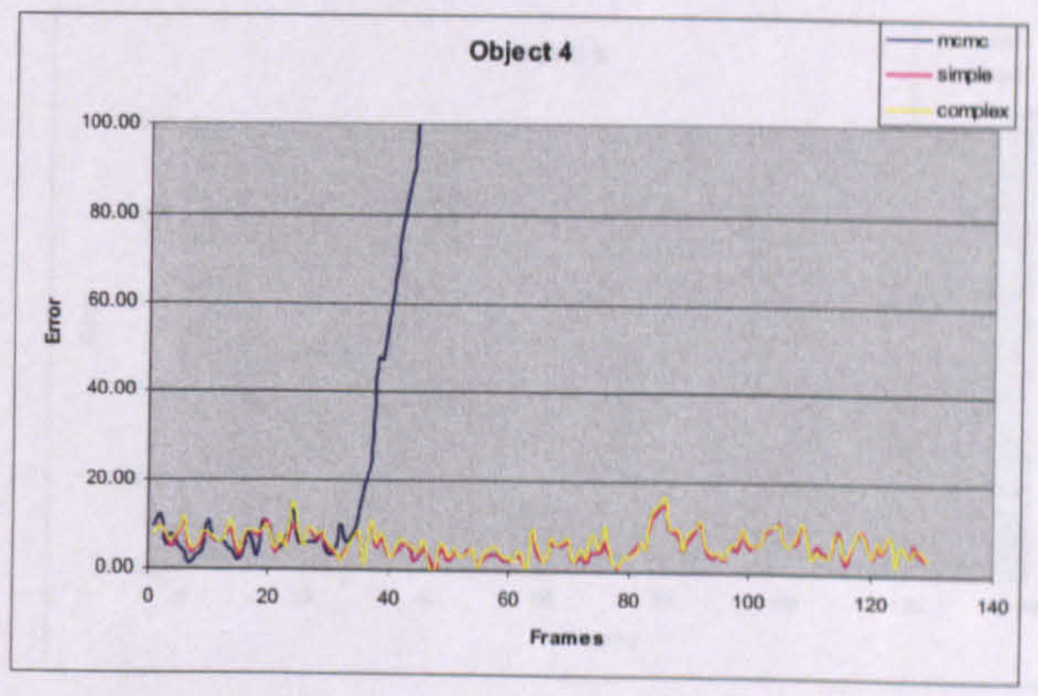
Figure 5.7: The video contains many partial occlusions, Khan's MCMC failed to track objects 1 and 3 to the end of the sequence. Both MKAMS algorithms track all 5 objects well.

Noise 3x









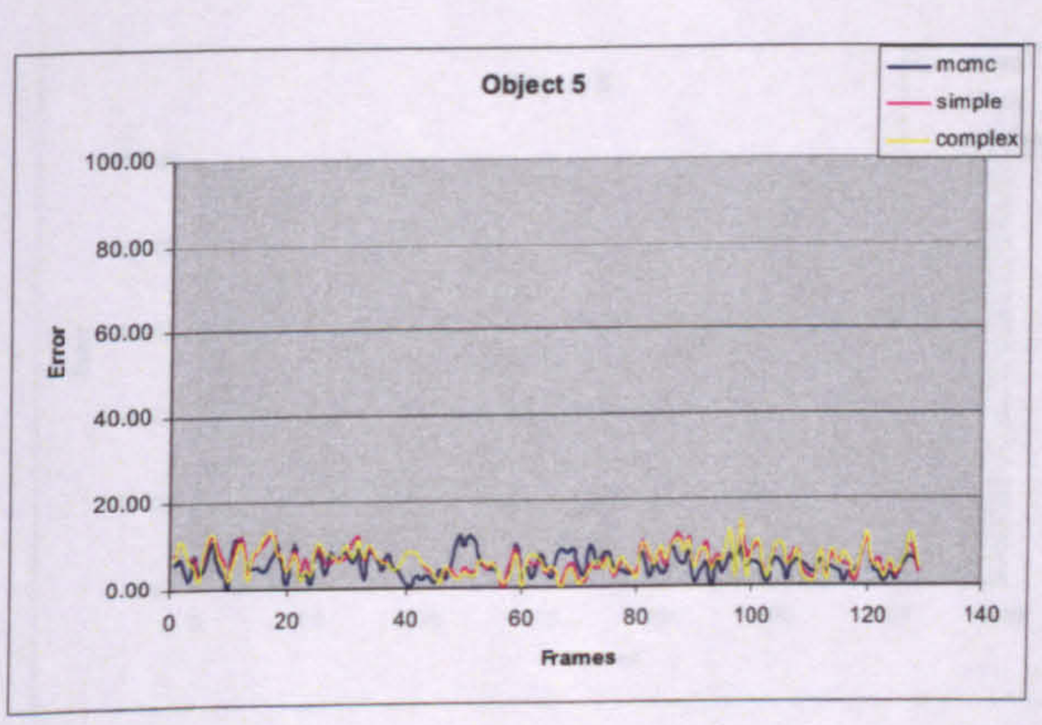
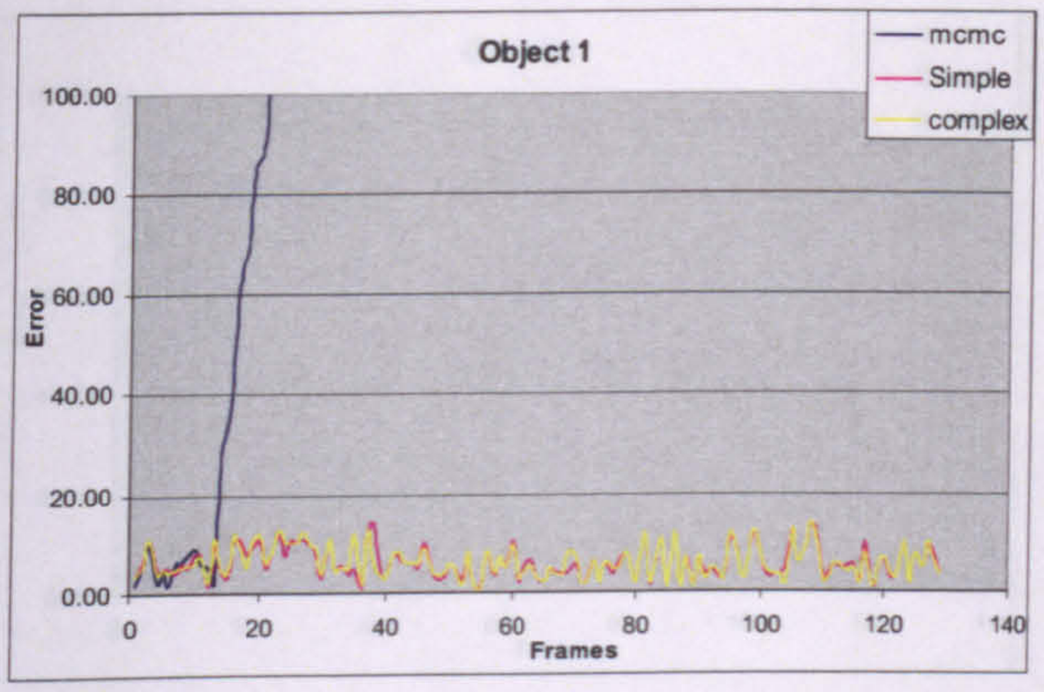
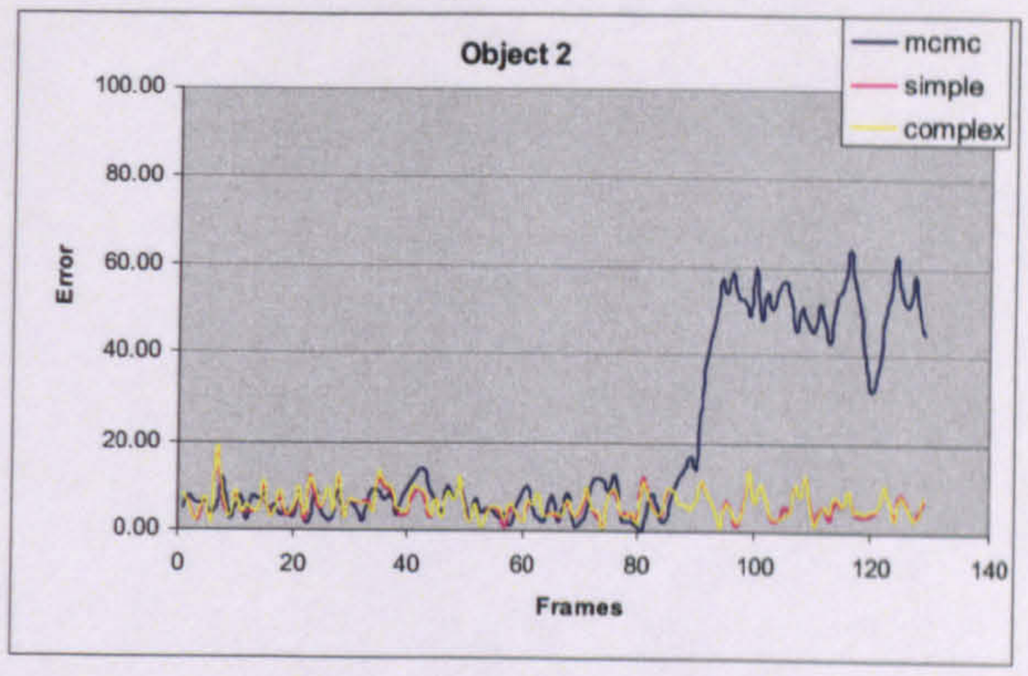
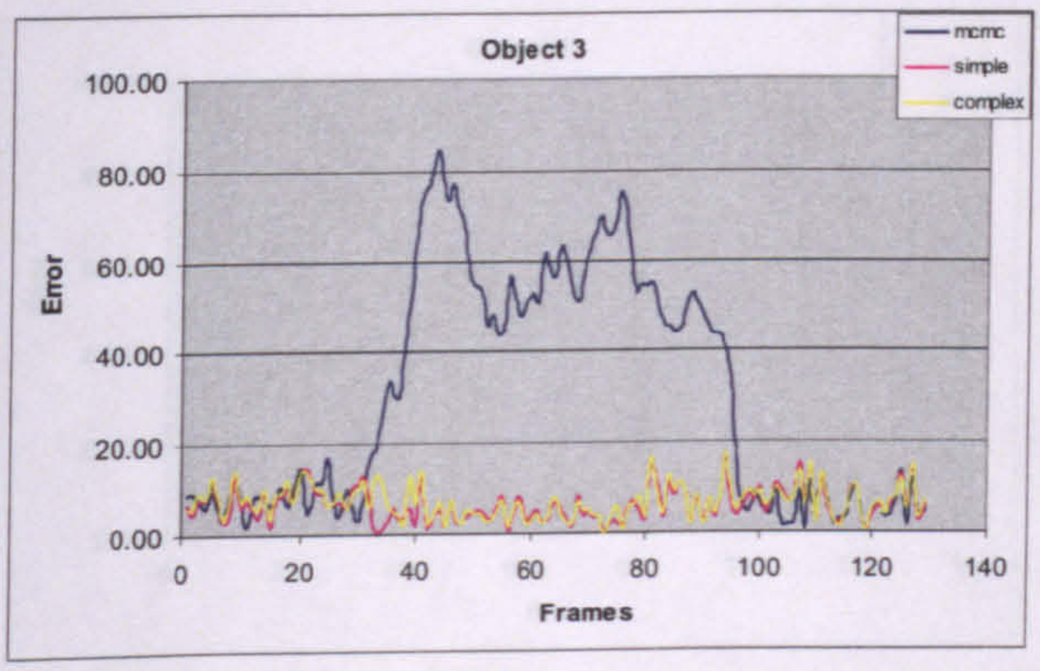


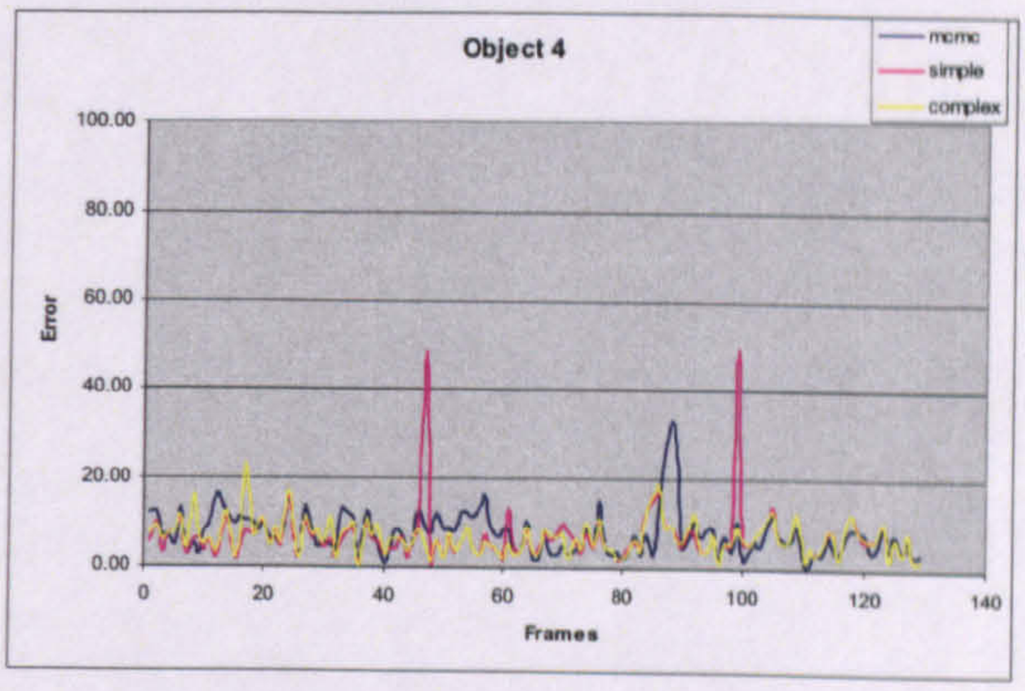
Figure 5.8: This video contains both partial and complete occlusions. Khan's MCMC fails to track objects 1, 3 and 4 to the end of the sequence. Both MKAMS algorithms track all 5 objects well.

Noise 4x









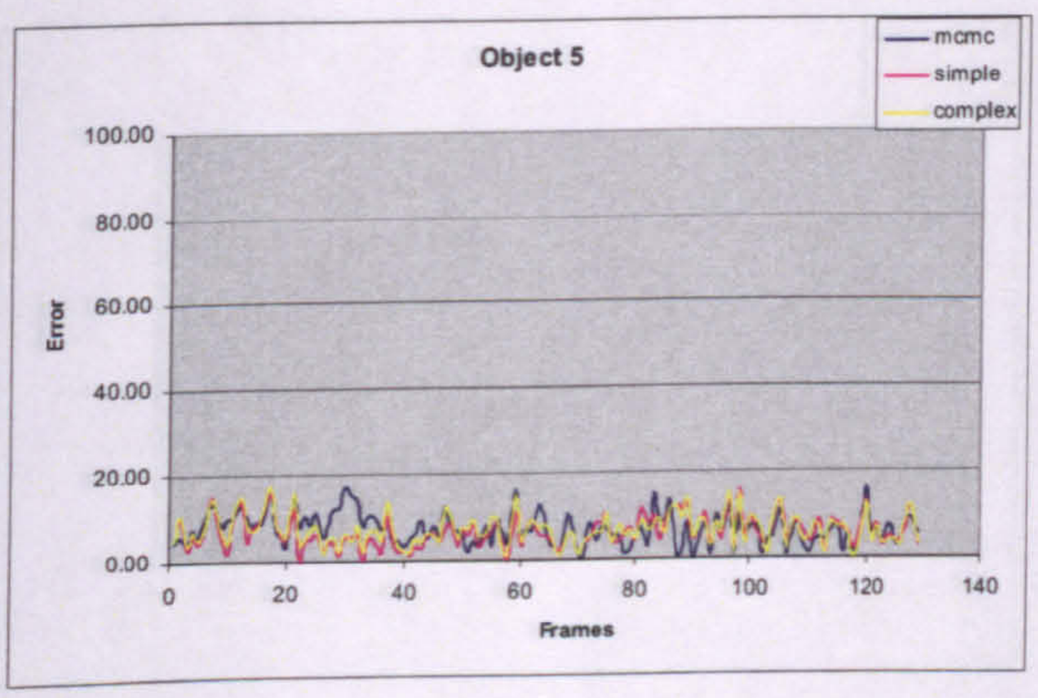


Figure 5.9: Khan's MCMC loses lock on objects 1, 2 and 3. Simple MKAMS begins to show greater errors than Complex MKAMS, although both algorithms successfully track all the 5 objects to the end of the sequence.

Noise 5x

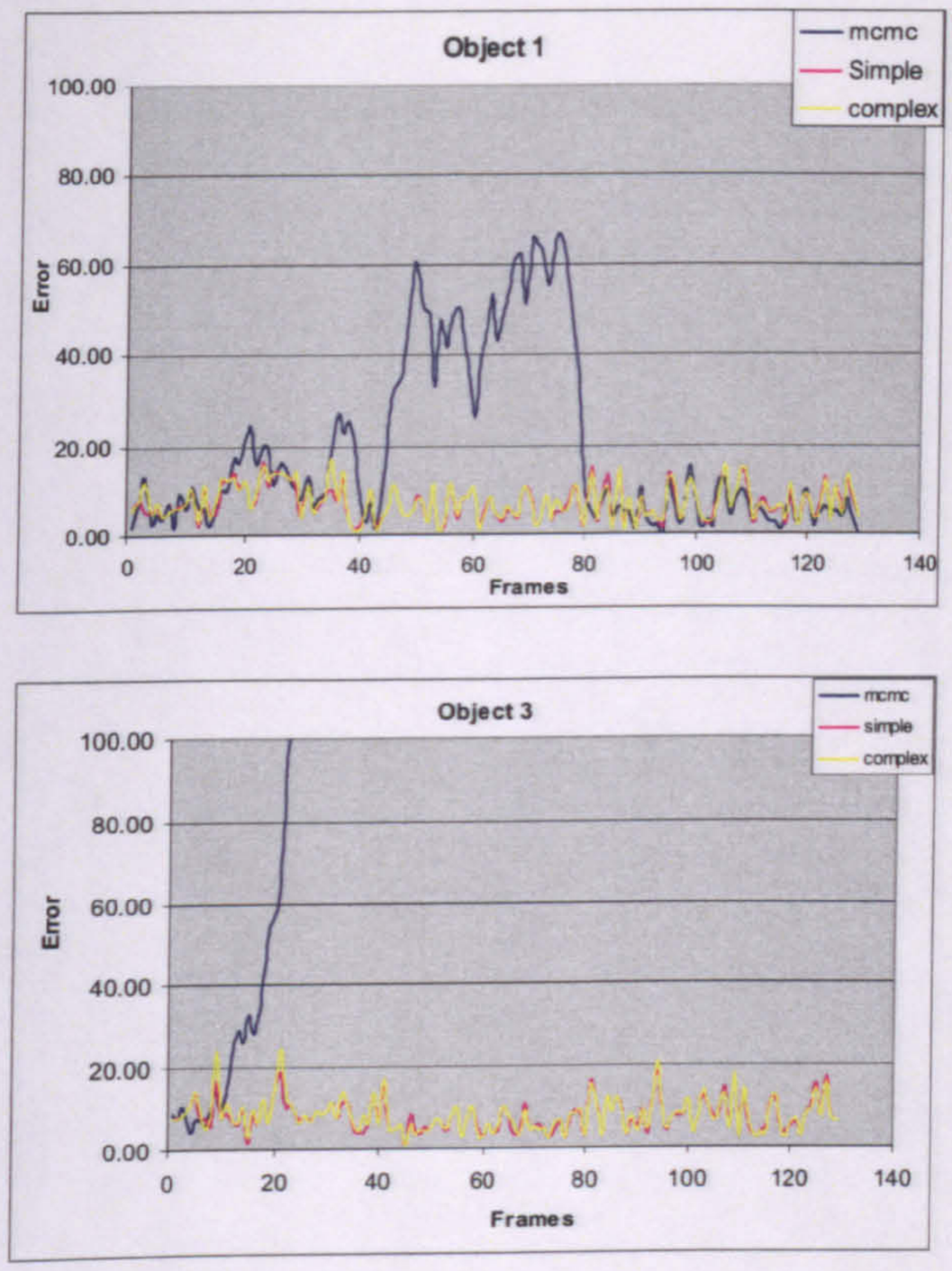
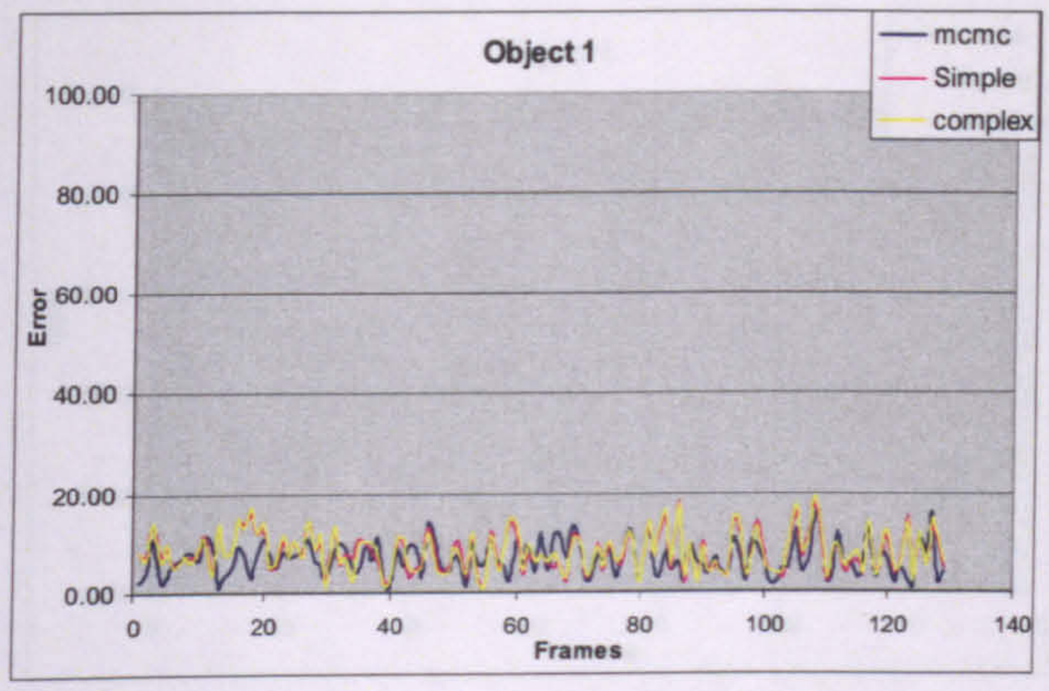
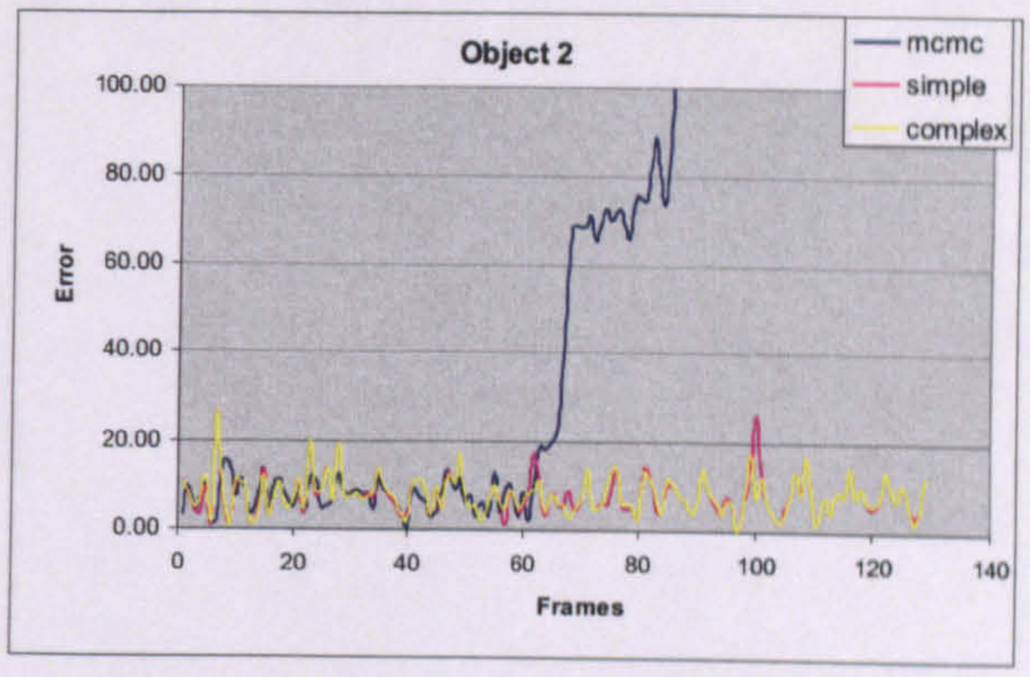
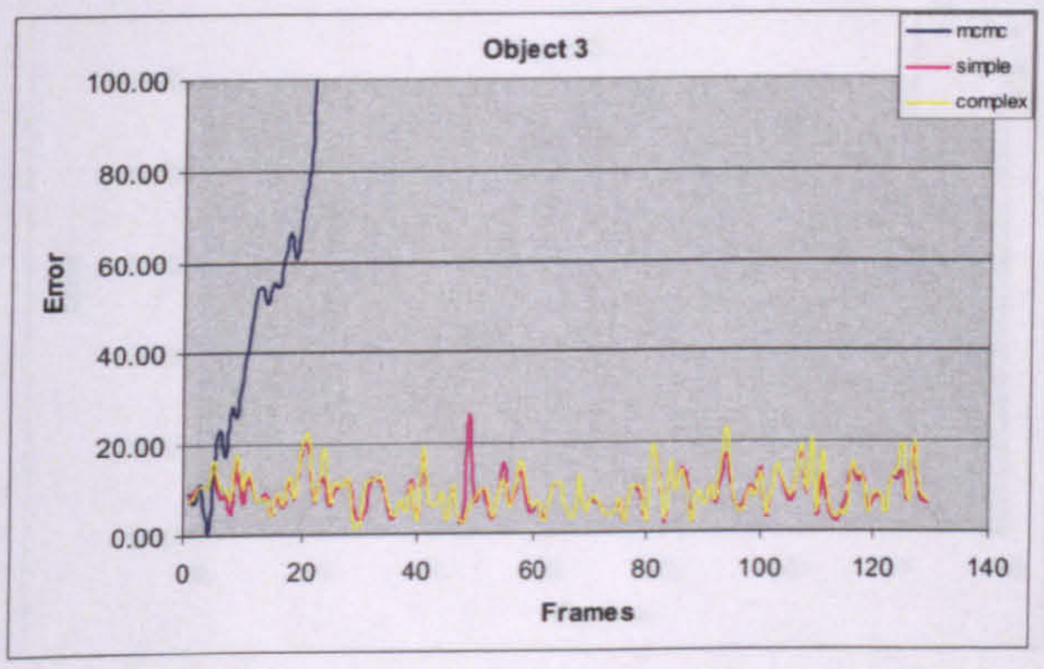


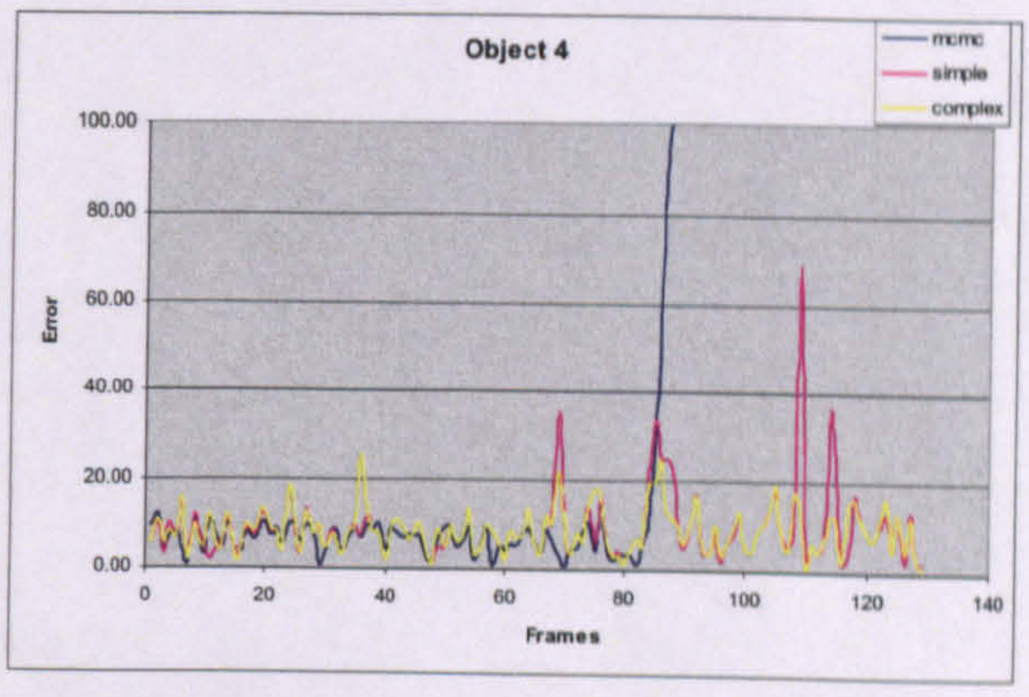
Figure 5.10: Khan's MCMC loses lock on objects 1, 2, 3 and 4 to the end of the sequence. Simple MKAMS loses lock on object 4, while Complex KAMS performs well.

Noise 6x









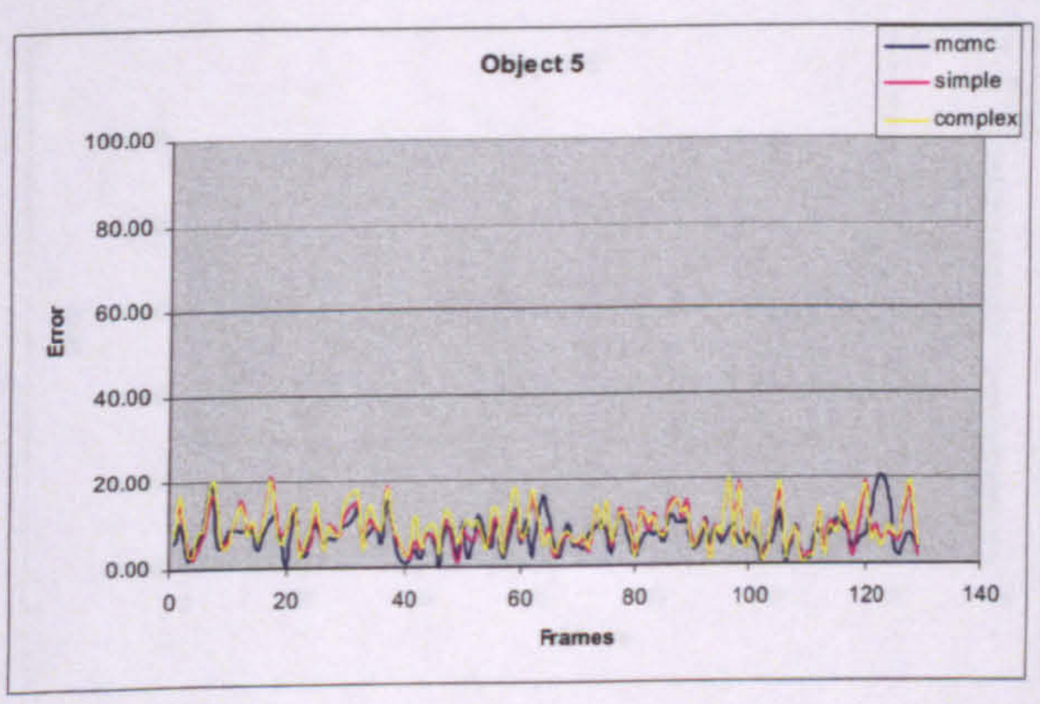
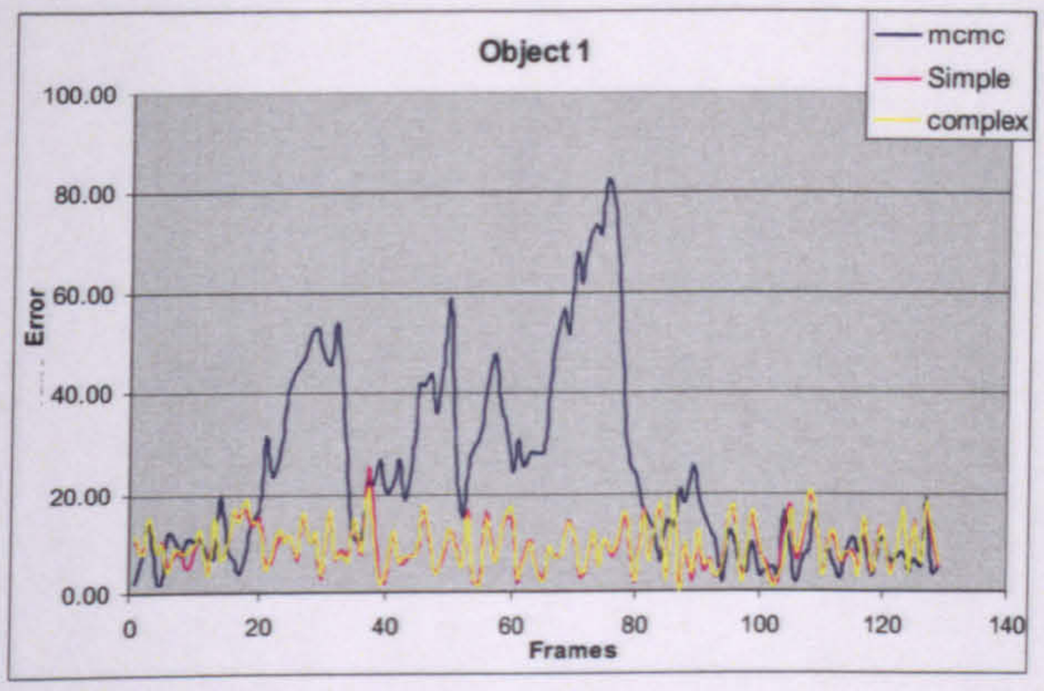
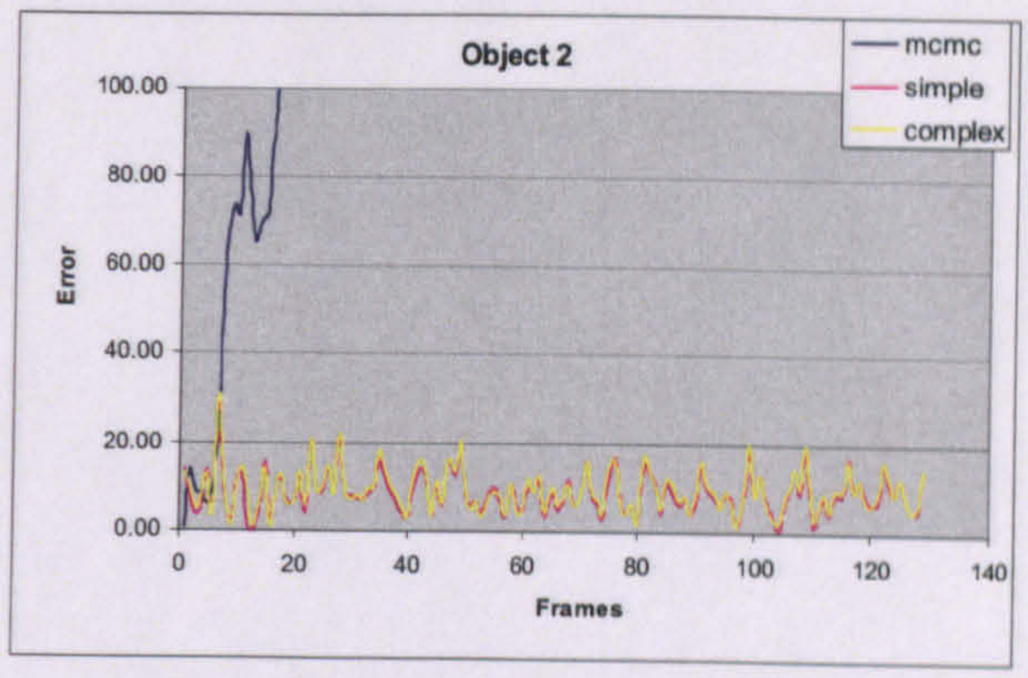
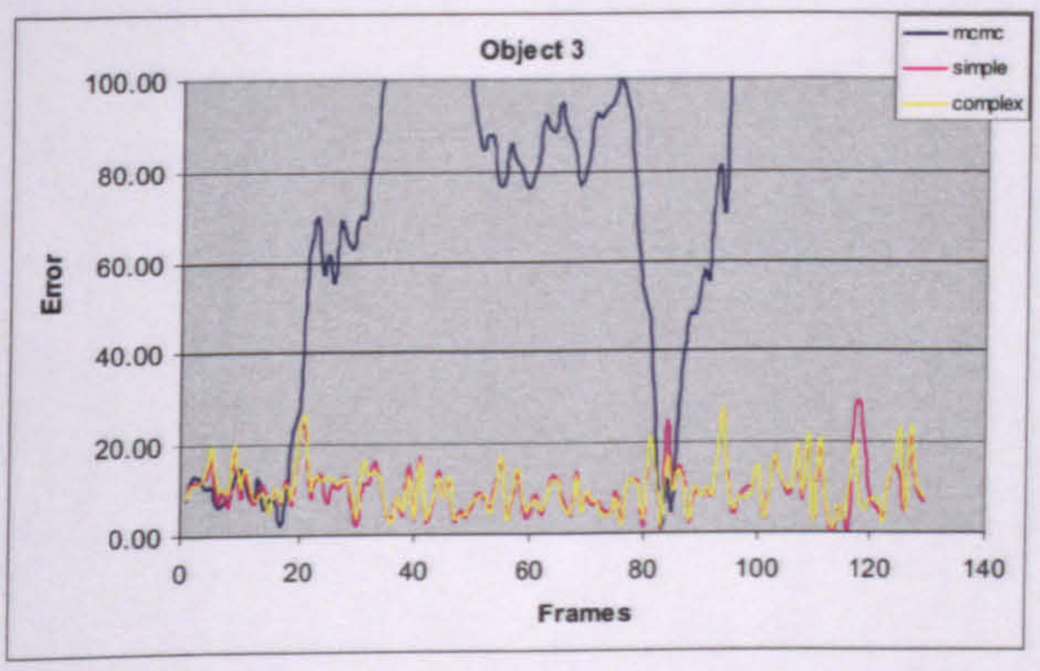


Figure 5.11: Khan's MCMC fails to track objects 2, 3 and 4 and Simple MKAMS loses lock on object 4 briefly. Complex MKAMS tracks all the objects to the end of the sequence.

Noise 7x







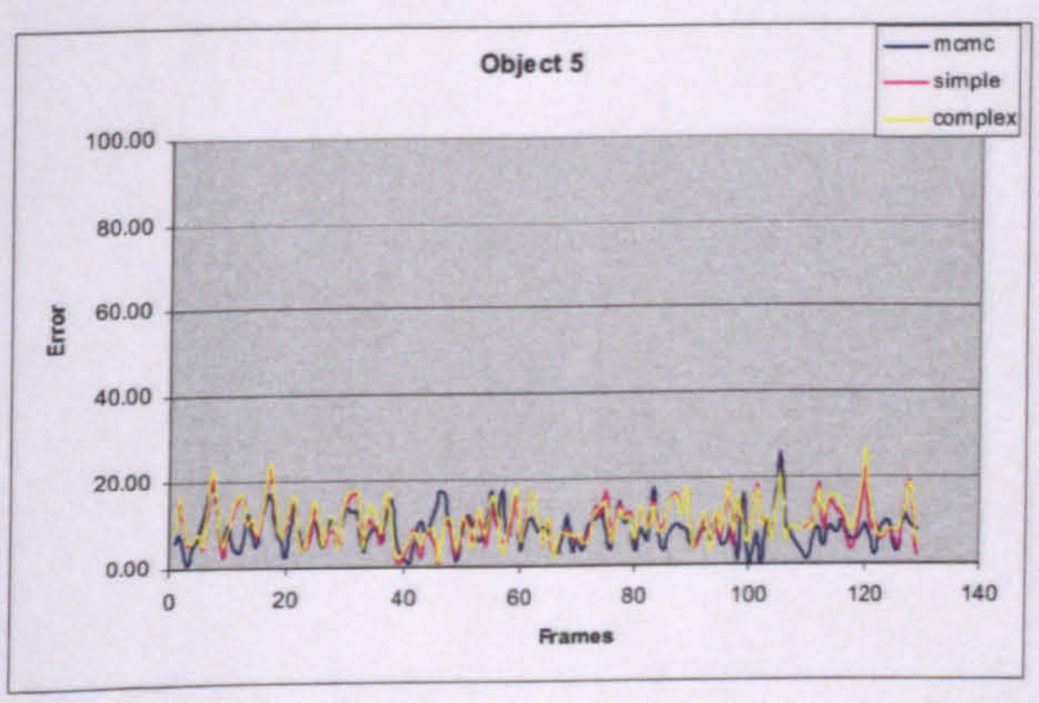


Figure 5.12: Khan's MCMC loses track of object 1, 2, 3 and 4. Simple MKAMS loses lock on object 4 near the 5th frame, Complex MKAMS fails 4th as well but after the simple MKAMS at around 35th frame.

Noise 8x

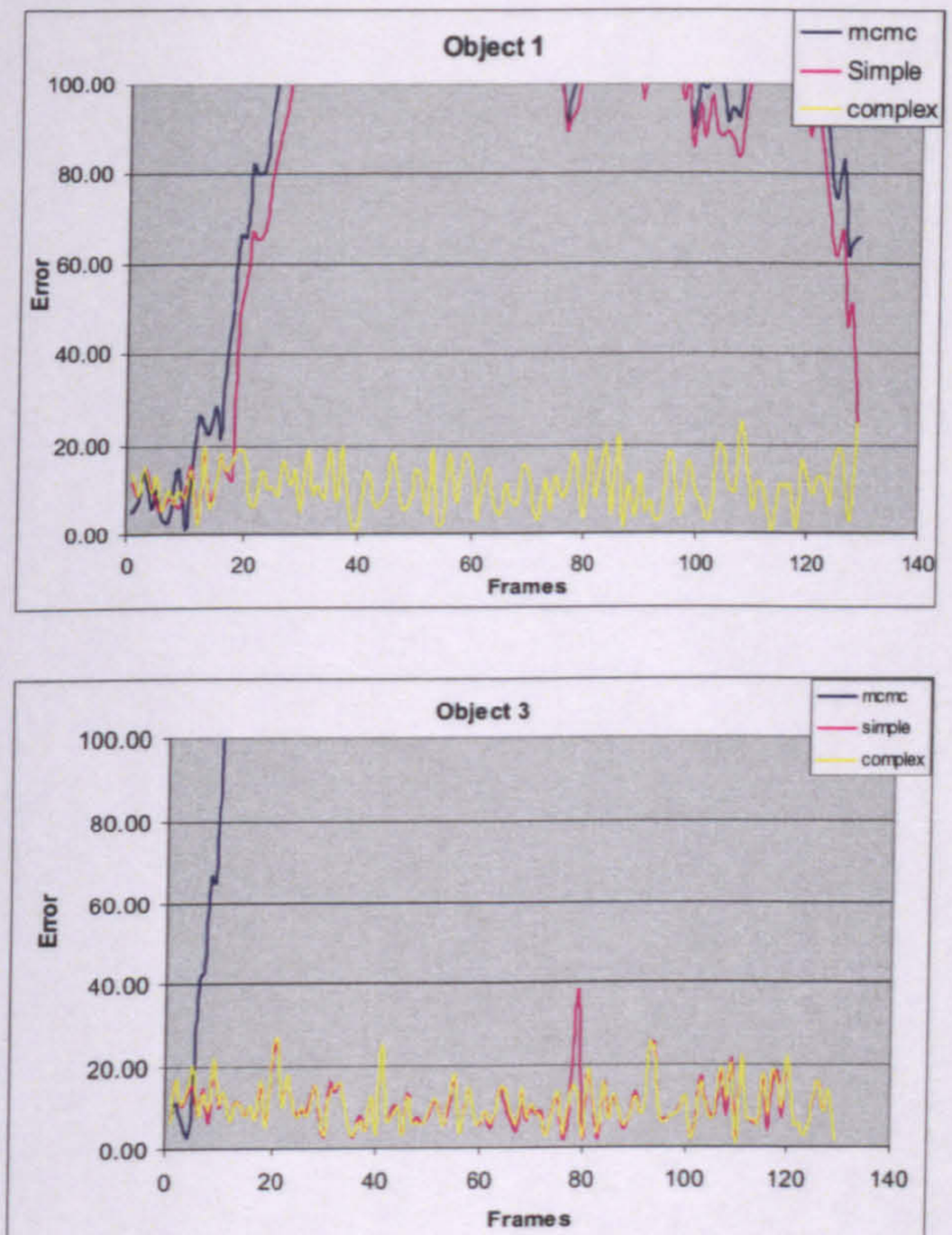


Figure 5.13: Khan's MCMC loses track of all the objects. Simple MKAMS fails to track objects 1 and 4. Complex MKAMS tracks all the objects successfully to the end of the sequence. Although we note that the RMS is increasing with the increasing noise levels.

Noise 9x

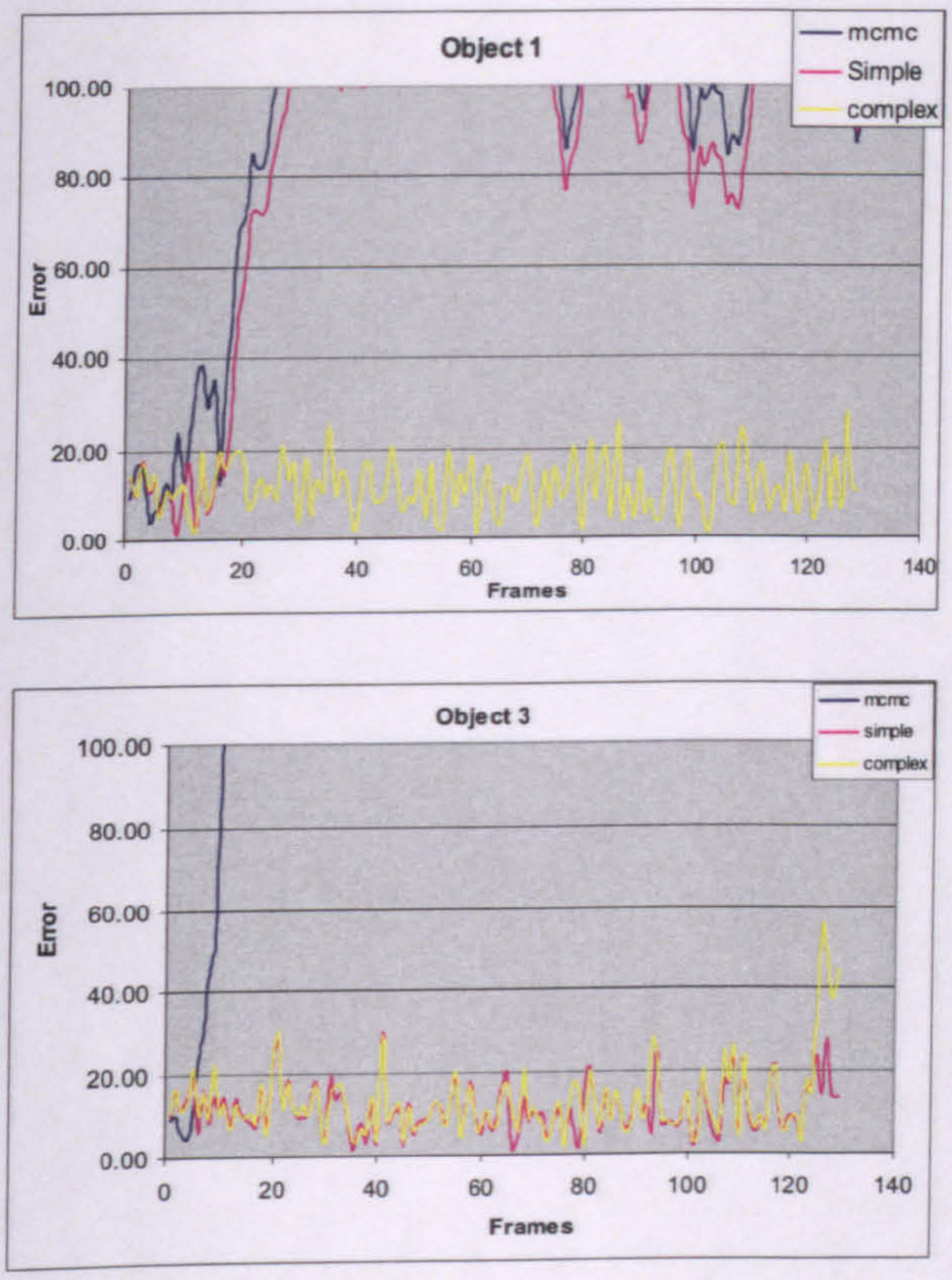
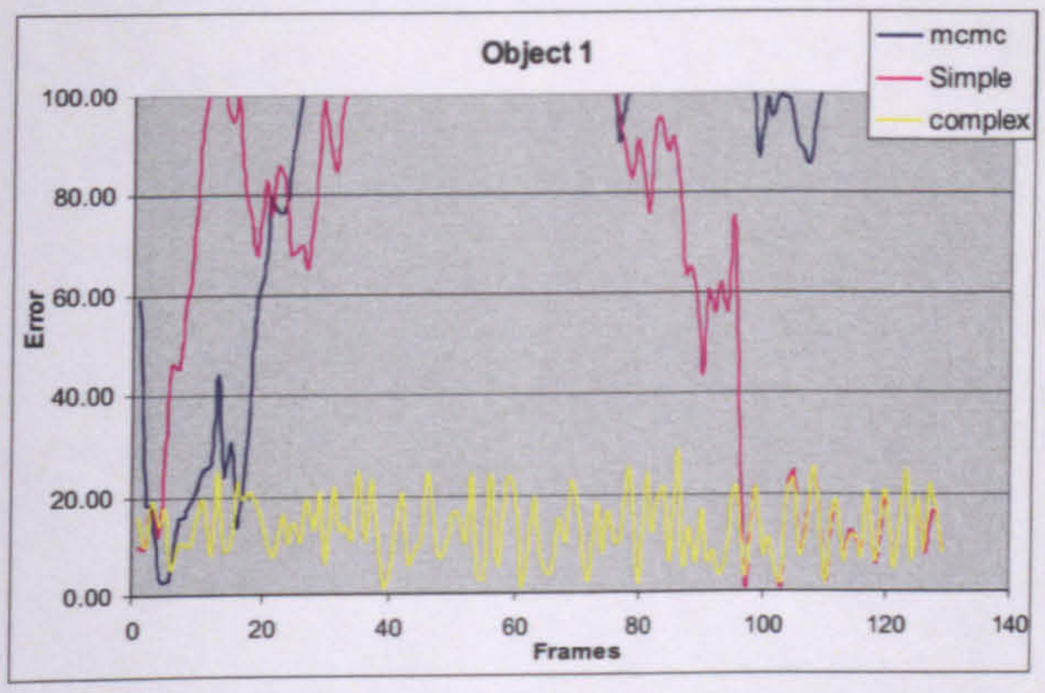
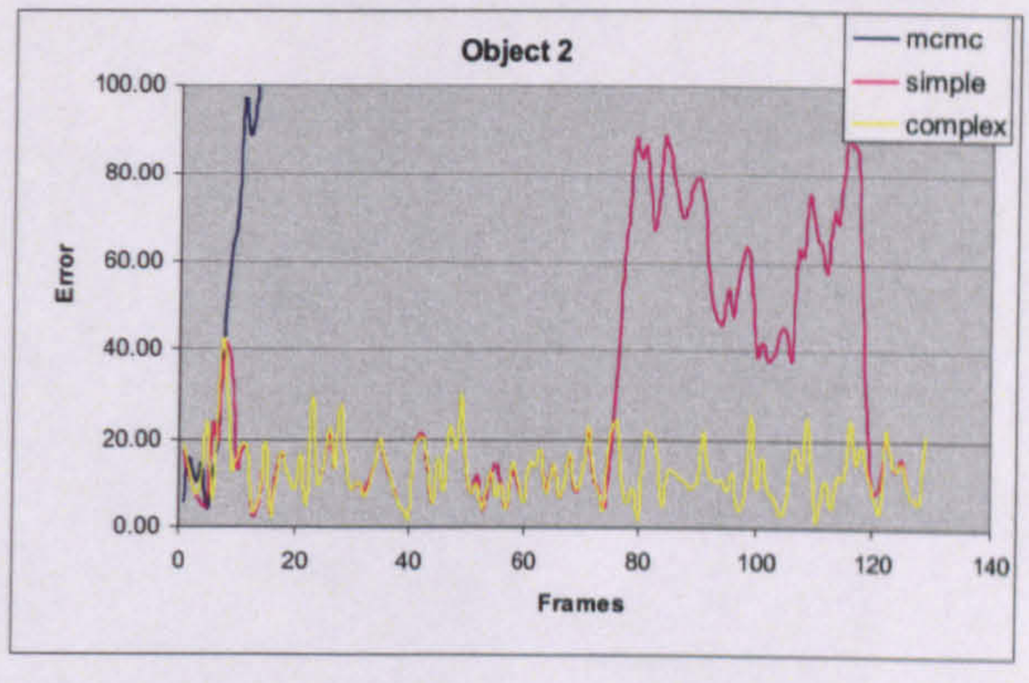
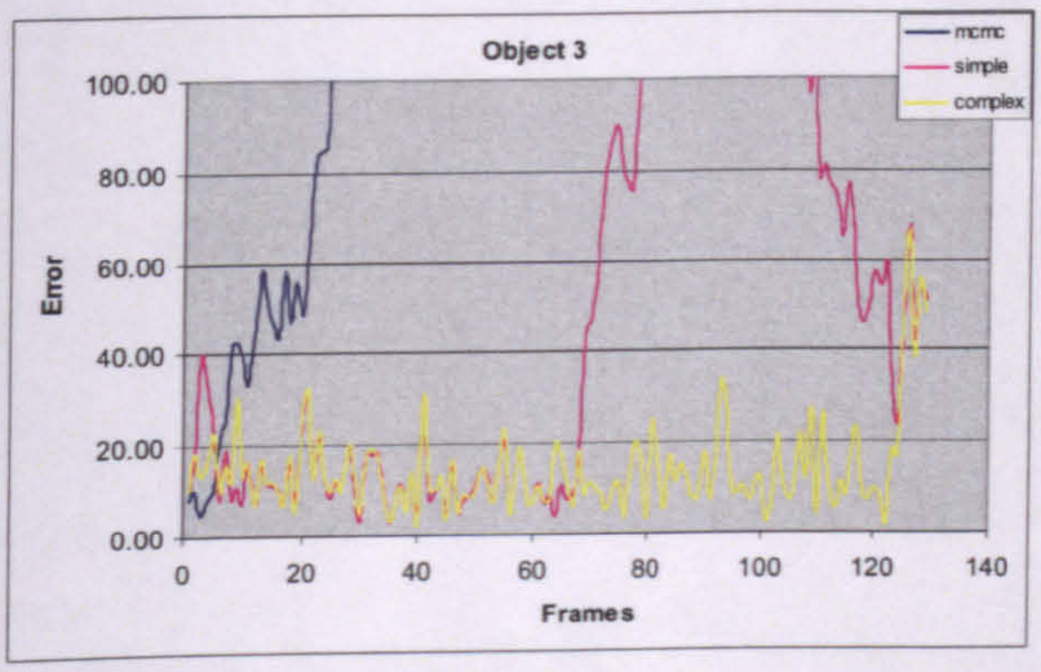


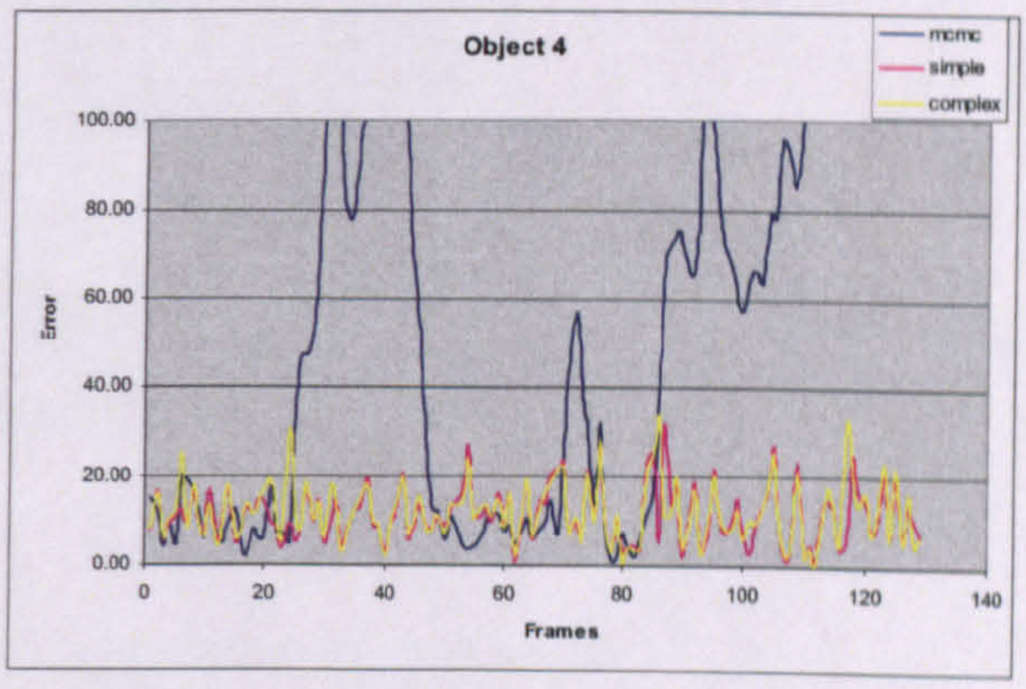
Figure 5.14: Khan's MCMC loses track of all the objects. Single MKAMS failed to track objects 1 and 4 while Complex MKAMS lost lock on objects 2 and 6 before the end of the sequence.

Noise 10x









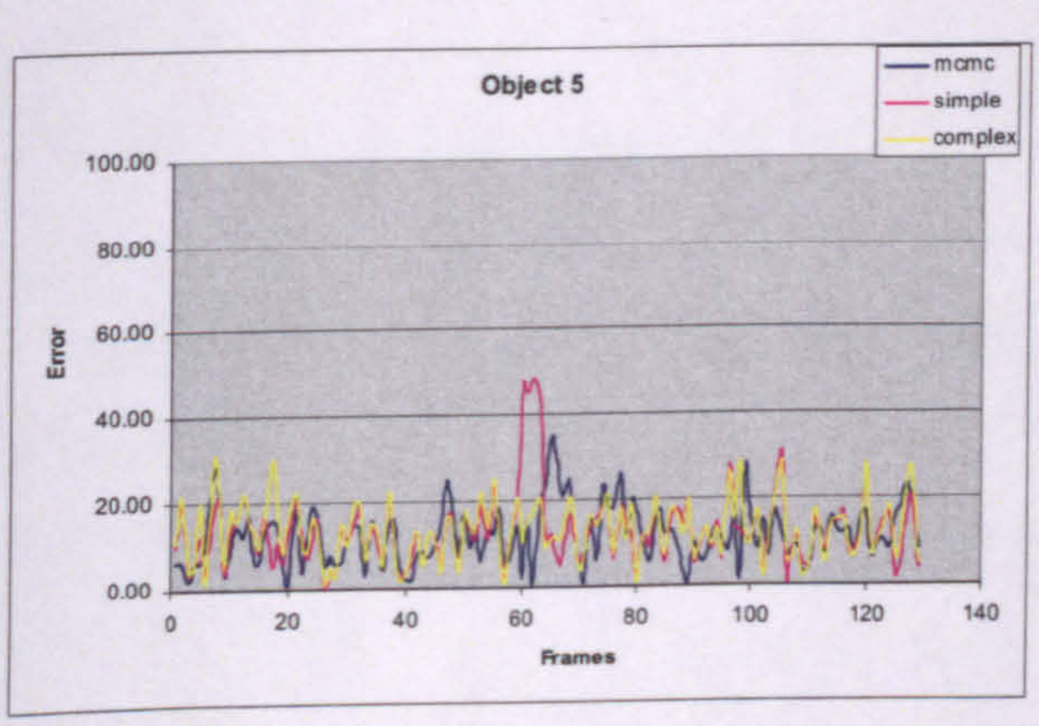
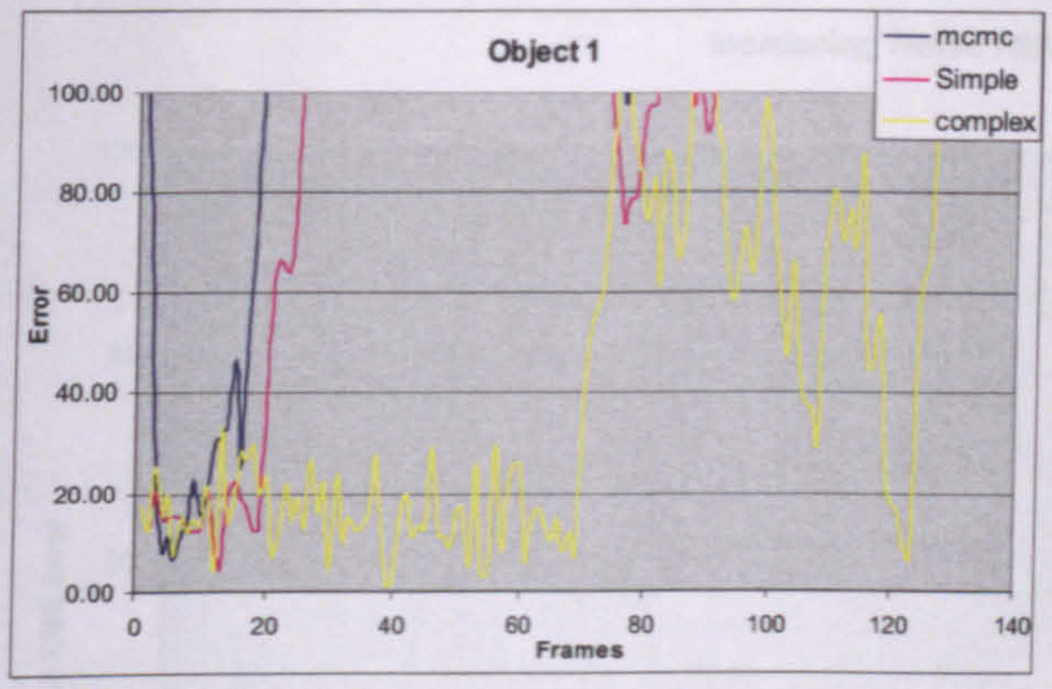
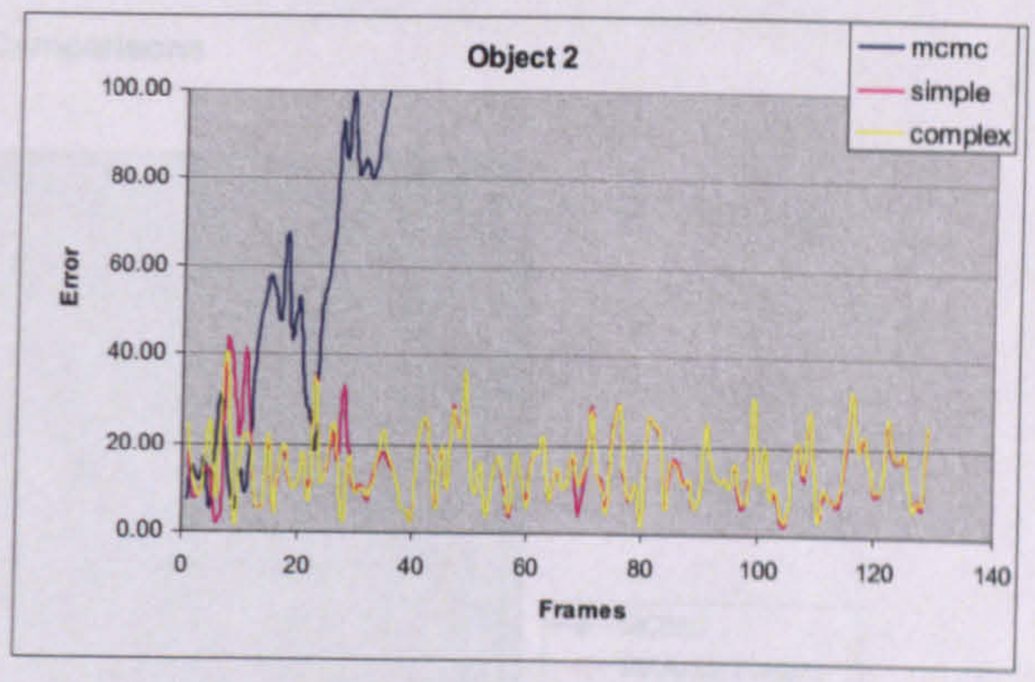
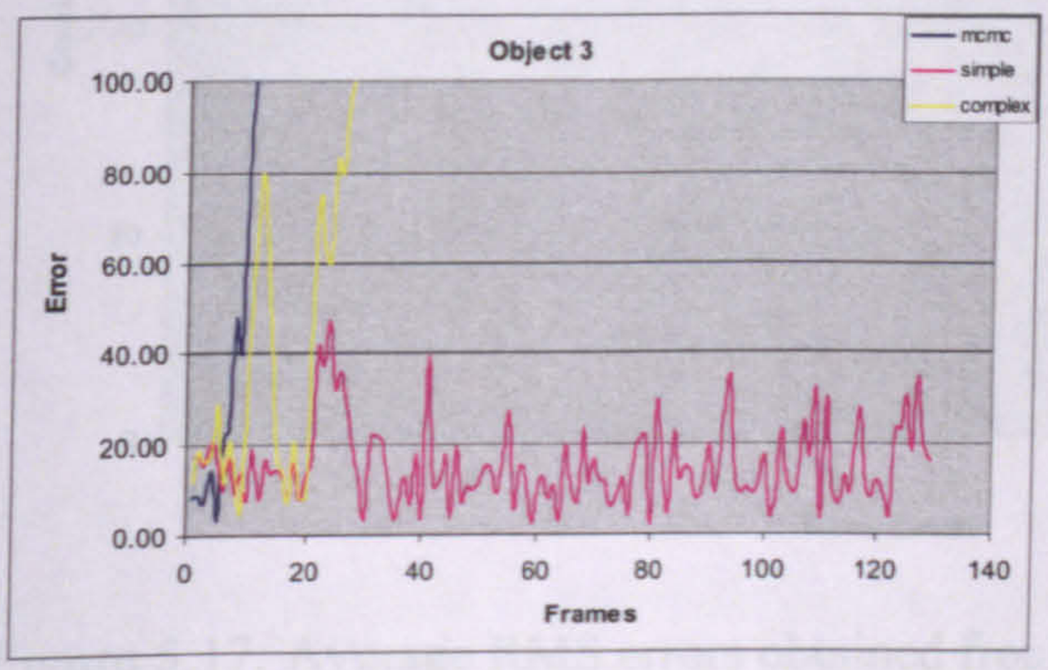


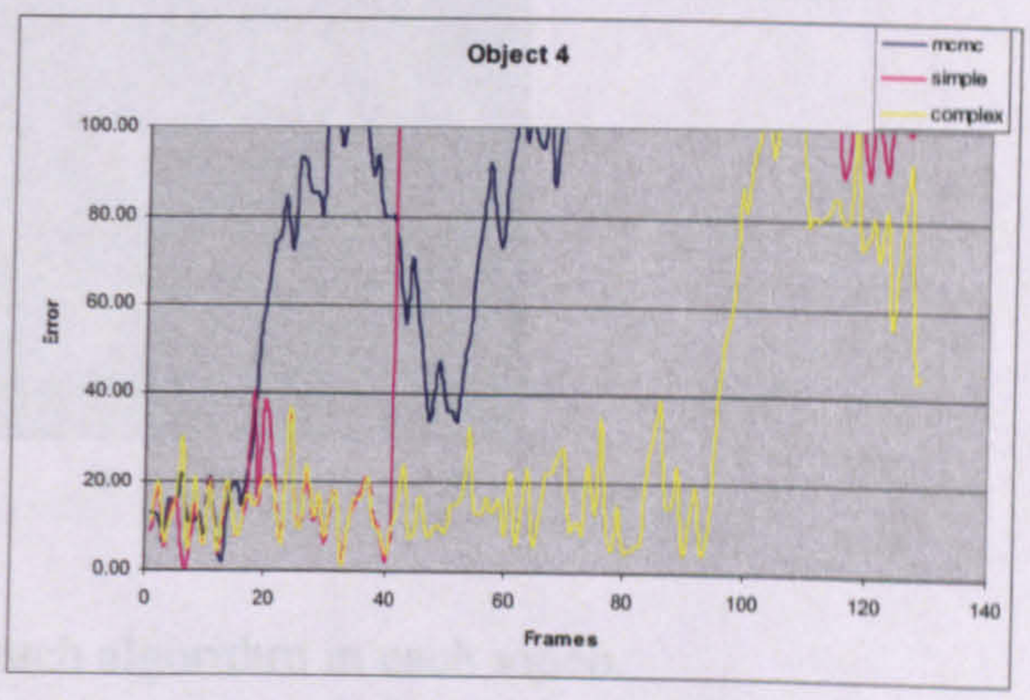
Figure 5.15: Khan's MCMC fails to track objects 1, 2, 3 and 4. Simple MKAMS lost lock on objects 1, 2, 3 and 4th object, while Complex MKAMS only failed to track object 3 before the end of the sequence.

Noise 12x









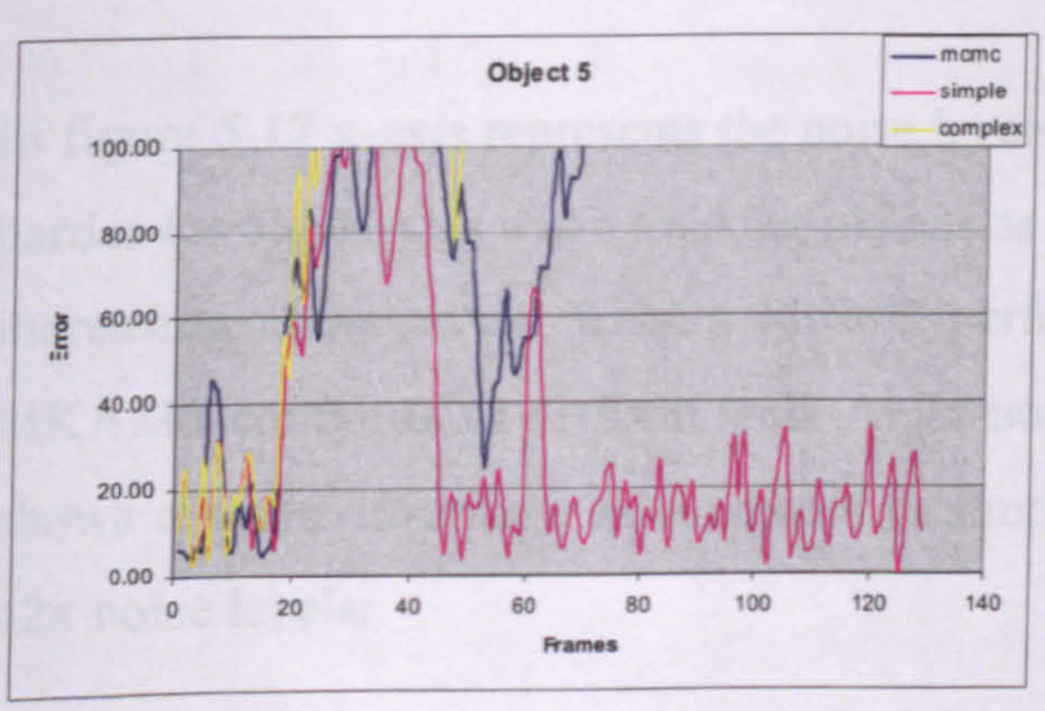


Figure 5.16: Khan's MCMC loses all five objects near the start of the sequence. Complex and Simple MKAMS track object 2 successfully till the end of the sequence but show

As noise levels are increased, the tracking problem becomes harder, and all three algorithms show some deterioration Figure 5.17 shows the average errors arising from each algorithm, for all tested videos. These are plotted in ascending order of noise level.

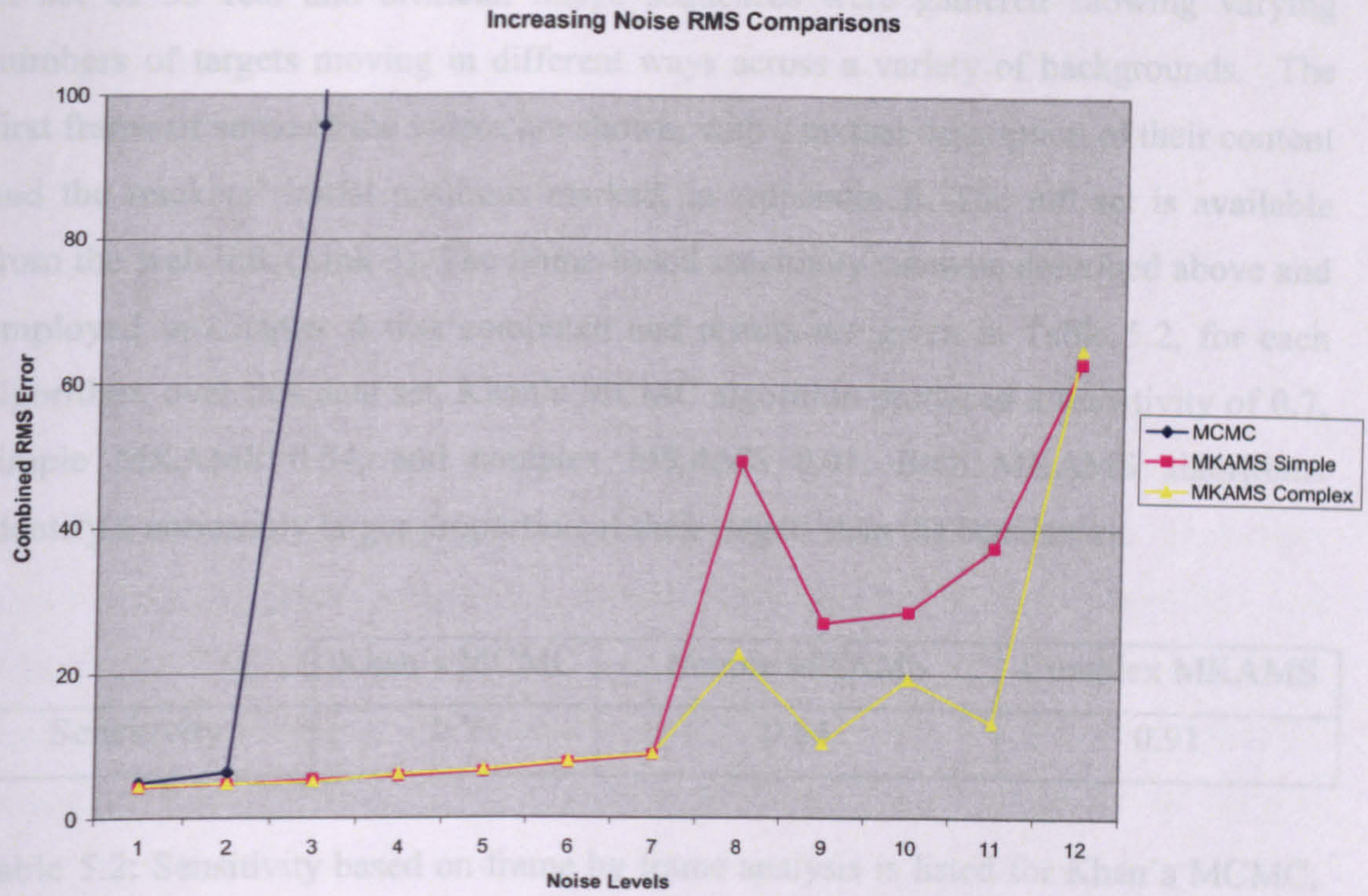


Figure 5.17: Average RMS errors obtained from each algorithm in each video.

In figure 5.17 x-axis represents the noise levels 1,2,3,4 … 12x. In theory it should get harder for algorithms when tracking objects as x increases and this is of course due to increasing noise levels. Khan's MCMC performs badly after 2x noise levels, but MKAMS continues to perform well. At 7x noise levels we see that complex KAMS shows a slight advantage as compared to simple KAMS. All three algorithms fail at 12x noise levels.

5.4.2 Robustness

As in Chapter 4, robustness is assessed via sensitivity estimates and McNemar's statistic. Measurement of sensitivity requires definitions of true positive and false positive outcomes. In single target tracking these are clear; a true positive is recorded at a given frame when the tracker is positioned over the target it was initialised to track, and a false positive otherwise. This measure can also be used in the evaluation of multi-target tracking, to provide an estimate of the proportion of correct target identifications across a data set.

A set of 33 real and artificial image sequences were gathered showing varying numbers of targets moving in different ways across a variety of backgrounds. The first frame of some of the videos are shown, with a textual description of their content and the trackers' initial positions marked, in Appendix B. The full set is available from the web link (Link 3). The frame-based sensitivity estimate described above and employed in Chapter 4 was computed and results are given in Table 5.2, for each algorithm, over this data set. Khan's MCMC algorithm produced a sensitivity of 0.7, simple MKAMS 0.84, and complex MKAMS 0.91. Both MKAMS algorithms identify a noticeably larger proportion of their targets than the benchmark.

	Khan's MCMC	Simple MKAMS	Complex MKAMS
Sensitivity	0.7	0.84	0.91

Table 5.2: Sensitivity based on frame by frame analysis is listed for Khan's MCMC, Simple MKAMS and Complex MKAMS algorithms.

What this data does not give, however, is any indication of how many targets were tracked successfully, i.e. for the entirety of their trajectory, and for how many tracking failed. To assess this, a second definition of true and false positive was employed. A true positive was recorded when a given target was tracked successfully from the beginning to the end of one of the test sequences. When tracking failed, that target generated a false positive for that sequence. Computed in this way and reported in Table 5.3, Khan's MCMC algorithm displays a sensitivity of 0.69. Simple and complex MKAMS produce values of 0.83 and 0.90 respectively. Both MKAMS algorithms track considerably more targets to completion.

	Khan's MCMC	Simple MKAMS	Complex MKAMS
Sensitivity	0.69	0.83	0.90

Table 5.3: Sensitivity based on trajectories is listed for Khan's MCMC, Simple MKAMS and Complex MKAMS algorithms.

Note that, while the two measures are highly correlated, the sensitivities computed from frame by frame analysis are slightly higher than those based on trajectories. This is because frame by frame analysis is more sensitive than trajectory-based analysis. In trajectory-based analysis, if an object is tracked till almost the end of the sequence, and tracking then fails, it is not counted as a true positive. The frame by frame analysis, however, counts all the frames in which the tracking was successful as true positives, and only the last few are marked as failures.

To provide a clear statement of relative robustness McNemar's test is again employed. McNemar's statistic (Chapter 2) requires a definition of success. When evaluating pairs of single target algorithms, the tracker which lost its lock on the target first was considered to have failed, and the competing algorithm was considered to have succeeded (Chapter 3 and 4). A logical way of comparing two multiple target tracking algorithms using McNemar's test is that the one tracking the most objects till the end of the specific video sequence is considered to be successful, and the other is considered to have failed. Using this definition, McNemar's test was employed to gauge the robustness of the 3 algorithms (Khan's MCMC, Simple MKAMS and Complex MKAMS) over the 33 test sequences described in Appendix B. Over 30 are required to make sure the central limit theorem starts to apply and z-score formula becomes valid.

Results are given in table 5.4.

		Objects Tracked Successfully			Winners during comparisons		
Video Name	**Total Objects**	**MCMC**	**KAMS Simple**	**KAMS Complex**	**Simple vs MCMC**	**Complex vs MCMC**	**Simple vs complex**
0x	5	5	5	5			
1x	5	5	5	5			
2x	5	3	5	5	Simple	complex	
3x	5	2	5	5	Simple	complex	
4x	5	4	4	4	Simple	complex	
5x	5	3	4	4	Simple	complex	
6x	5	3	4	5	Simple	complex	Complex
7x	5	3	4	4	Simple	complex	
8x	5	2	3	5	Simple	complex	Complex
9x	5	2	3	4	Simple	complex	Complex
10x	5	1	2	5	Simple	complex	Complex
12x	5	1	3	4	Simple	complex	Complex
11	5	3	4	4	Simple	complex	
12	5	3	5	5	Simple	complex	
13	3	1	3	3	Simple	complex	
14	4	0	4	4	Simple	complex	
15	3	1	2	3	Simple	complex	Complex
16	5	4	5	5	Simple	complex	
17	10	7	7	9		complex	Complex
18	10	9	9	10		complex	Complex
19	9	8	7	7	mcmc	mcmc	
20	2	1	2	2	single	complex	
21	5	5	4	4	mcmc	mcmc	
22	3	3	3	3			
23	7	7	7	7			
24	18	15	16	16	single	complex	
25	3	3	2	2	mcmc	mcmc	
26	4	2	3	3	single	complex	
27	3	2	3	3	single	complex	
28	10	8	8	8			
29	3	2	0	1	mcmc	mcmc	Complex
30	3	3	3	3			
31	6	5	6	6	single	complex	

Table 5.4: Results of tracking 33 videos with Khan's MCMC, Simple MKAMS and Complex MKAMS algorithms. Note that cells in the three rightmost columns are left blank when the algorithms concerned track the same number of targets successfully.

Based on this data McNemar's test was performed and the results of McNemar's test are listed in Table 5.5.

	Simple vs MCMC	Complex vs MCMC	Complex vs Simple
N_{sf}	21	23	9
N_{fs}	4	4	0
Z-score	3.20	3.46	2.67
McNemar Results	>99.5%	>99.5%	>99%

Table 5.5: Table shows the results of Mcnemar's test, where N_{sf} is the number of times Algorithm A succeeds and Algorithm B fails, and N_{fs} is the number of times Algorithm A fails and B succeeds.

Table 5.5 shows the results of the pair wise comparison of the three algorithms. The results show that

- Simple MKAMS is significantly more robust than Khan's MCMC algorithm with a confidence of 99.5%
- Complex MKAMS is significantly more robust than Khan's MCMC with a confidence of 99.5

Complex MKAMS significantly more robust than Simple MKAMS with a confidence of 99%

5.4.3 Execution Time Comparison

Khan's MCMC and the two MKAMS algorithms described above were used to track objects in the set of 33 image sequences described in Appendix B. The average time taken by each to process one frame was calculated and is reported here. Given the region-based appearance model used by the three trackers, it is expected that computational cost will increase with target size. In chapter 4 this was taken into account by plotting average execution time against target radius. As there are multiple

objects with varying radii in each of image sequences used here, the total area of the objects tracked in each image sequence was computed and used instead of the radius measure. The results are shown in Figure 5.18 in increasing order of total area of tracked objects.

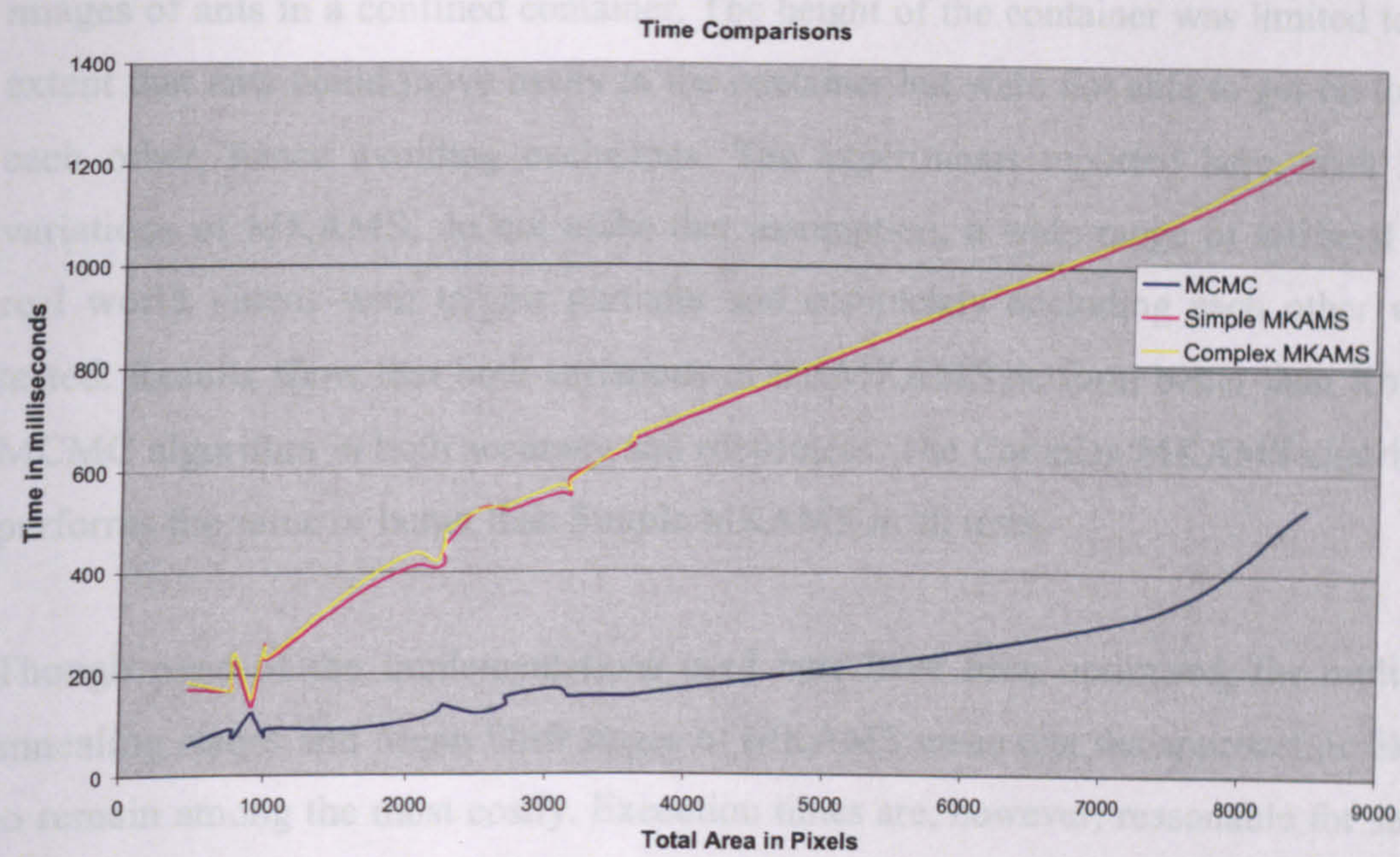


Figure 5.18: Average time in milliseconds taken to process one frame in increasing order of total area tracked.

Figure 5.18 shows complex MKAMS to be slightly more expensive on average as the simple MKAMS algorithm; this is because the complex version processes all four stages even if an intermediate annealing stage fails, while simple MKAMS abandons processing of the object in that frame as soon as a stage fails. Both MKAMS algorithms are computationally much more expensive than the benchmark Khan's MCMC algorithm.

5.5 Conclusion

The KAMS algorithm developed and presented in Chapter 4 has been extended to produce a multi-target Kernel Annealed Mean Shift (MKAMS) algorithm in which each target is tracked by an individual tracker and interactions are handled using an updated version of Khan's interaction filter (Khan et al 2004). Use of a multi-level interaction filter allows the trackers to retreat to an alternative local maxima if

occlusion by another target is noted. Two versions of the MKAMS were identified, based on the type of interaction filter used.

The experiments that Khan (Khan et al 2004) performed employed sequences of images of ants in a confined container. The height of the container was limited to the extent that ants could move easily in the container but were not able to get on top of each other, hence avoiding occlusions. The experiments reported here, with both variations of MKAMS, do not make that assumption; a wide range of artificial and real world videos with targets partially and completely occluding each other were tested. Results show that both variations of the MKAMS perform better than Khan's MCMC algorithm in both accuracy and robustness. The Complex MKAMS algorithm performs the same or better than Simple MKAMS in all tests.

Though none of the implementations used here have been optimised, the multiple annealing stages and Mean Shift stages of MKAMS mean that the approach is likely to remain among the most costly. Execution times are, however, reasonable for small numbers of small targets, and the improved performance of these algorithms will make them attractive in applications that are not time-critical.

Chapter 6: Conclusions and Future Work

6.1 Overview

While tracking objects using particle filter-based algorithms, management of the spread of the particle set is a key issue. For effective tracking in real-world environments the particle set must sample widely enough to represent all reasonable alternatives in areas of ambiguity. It must not, however, become diffused, spreading across the image plane rather than clustering around the object of interest. When this happens particles tend to migrate towards local maxima in their evaluation function, becoming caught on clutter and losing track of the target.

This thesis has addressed the particle management problem in visual tracking. Specifically, it has investigated the ability of hybrid kernel mean shift/particle filtering algorithms to control particle sets and provide accurate and robust tracking of single and multiple targets.

The first kernel mean shift/particle filter hybrid was proposed by Maggio and Cavallaro (Maggio and Cavallaro 2005). This combines Condensation with mean shift tracking to produce a system in which particles are alternately diffused by Condensation and clustered towards the local maxima by mean shift. Maggio and Cavallaro's hybrid typically shows the performance expected of Condensation, but requires noticeably fewer particles, greatly reducing computational cost. The hybrid tracker typically requires 80-90% fewer particles than regular condensation to achieve similar results (Maggio and Cavallaro 2005).

Two approaches to the construction of hybrid tracking algorithm have been considered here. The first (SOK, Chapter 3), makes explicit the iterative diffuse-cluster structure implicit in Maggio and Cavallaro's original hybrid algorithm, only diffusing when necessary and then carpeting a large, fixed area around the prior with a small number of carefully placed particles. The second (KAMS, Chapter 4), integrates kernel mean shift into the annealed particle filtering algorithm. The use of annealing allows the process noise employed in the particle filter to be increased, spreading the particle set over a larger area, with mean shift again drawing them back together at each stage of the annealing process. The KAMS algorithm was subsequently extended to multi-target tracking (MKAMS, Chapter 5) by incorporating the interaction filter successfully employed by Khan et. al. (2004).

6.2 Contributions

The contribution of this thesis is to show that:

- Manipulation of the natural diffuse-cluster structure of particle filter/mean shift hybrids can result in more robust and efficient tracking. In particular, larger search areas should and can be used, in the expectation that the mean shift stage will draw the particles towards the true target.
- Careful particle placement within an enlarged search area improves performance still further; annealed particle filtering is a suitable mechanism for managing that placement.
- Integrating Khan et al's interaction filter with annealed particle filtering and mean shift both allows the hybrid approach to be extended to multiple targets and supports a wider range of more informed responses to target interaction.
- Integrating kernel mean shift and particle filtering generally results in improved performance over the component algorithms involved.

6.3 Future Work

Visual tracking is a longstanding and challenging problem in computer vision, and the need to produce techniques and algorithms that are less sensitive to the classic problems of noise, occlusion and clutter will continue for some time yet. A number of specific developments of the methods investigated here are, however, apparent:

- Like the annealed particle filter, KAMS is not a Bayesian tracker. Annealing and mean shift steps change the particle distribution so that the new sample set is not drawn from the original. Although it performs very well against other algorithms, it would be interesting to attempt to develop a fully Bayesian version of KAMS.

- Though the MKAMS algorithm displayed superior tracking performance, MKAMS processing time is prohibitive in many multi-target tracking applications. One way to reduce this cost may be to combine hill-climbing and interaction filtering with an annealed particle filter in a different way. It may be possible to introduce annealing and multi-level interaction filters into the MCMC sampling-based algorithm, replacing the mean shift with the MCMC algorithm's hill climbing patterns. This may reduce processing overheads and make MKAMS a useable option for real world applications.

References:

Ali and Aggarwal 2001	Ali, A. and Aggaewal, J. 2001. Segmentation and recognition of continuous human activity. In IEEE Workshop on Detection and Recognition of Events in Video. 28–35.
Arulampalam et al 2002	S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear non-Gaussian Bayesian tracking," *IEEE Trans. Signal Processing*, 55 (2), pp. 174-188, Feb 2002.
Ballard and Brown 1982	Ballard, D. and Brown, C. 1982. Computer Vision. Prentice-Hall.
Bar-Shalom and Fortmann 1988	Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association.* Academic Press, 1988
Bar-shalom and Li 1993	Y. Bar-Shalom and X.R. Li, Estimation and Tracking: Principles, Techniques and Software, Artech House, Boston, MA, 1993.
Berg et al 2005	A. Berg, T. Berg, and J. Malik, Shape Matching and Object Recognition Using Low Distortion Correspondence, CVPR,2005.
Biesiadecki et al 2001	J. Biesiadecki et al. "The Athena SDM Rover: a Test bed for Mars Rover Mobility", Proc. International Symposium on Artificial Intelligence and Robotics & Automation in Space, St-Hubert, Canada, June 2001
Bradski 1998	G.R. Bradski, Computer vision face tracking as a component of a perceptual user interface, Proc. 4th IEEE Workshop on Applications of Computer Vision (1998) 214-219.
Brown et al 2005	L. M. Brown, A. W. Senior, Ying-li Tian, Jonathan Connell, Arun Hampapur, Chiao-Fe Shu, Hans Merkl, Max Lu, "Performance Evaluation of Surveillance Systems Under Varying Conditions", IEEE Int'l Workshop on Performance Evaluation of Tracking and Surveillance, Colorado, Jan 2005.
Canny 1986	Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986
Carpenter et al 1999	J. Carpenter, P. Clifford, and P. Fearnhead, "Improved particle filter for nonlinear problems," Proc. Inst. Elect. Eng., Radar, Sonar, Navig., 1999.
Chang and Ansari 2005	C. Chang and R. Ansari. Kernel particle filter for visual tracking. IEEE Signal Processing Lett., 12(3):242 – 245, March 2005.
Chen et al 2007	Chia-Chih Chen; Bhargava, M.; Ryoo, M.S. ; Aggarwal, J.K., Detection of abandoned objects in crowded environments, Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on 5-7 Sept. 2007, page 271-276
Clark et al 2008	A Clark and C. Clark, Performance Characterisation in Computer Vision – A Tutorial, http://peipa.essex.ac.uk/benchmark/tutorials/essex/tutorial.pdf last visited 8th August 2008.
Collins 2003	R.T. Collins, Mean Shift blob tracking through scale space, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (2003) 234-240.
Comaniciu et al 2003	D. Comaniciu, V. Ramesh, and P. Meer, Kernel-based object tracking, IEEE Trans. Pattern Analysis and Machine Intelligence, 25, 5 (2003) 564–577.
Courtney and Böttcher 2003	www.eucognition.org/ecvision/industrial_liaison/Industrial_Applications_of_Cognitive_Vision.pdf
Crisanand et al 1999	D. Crisan, P. Del Moral, and T. J. Lyons, "Non-linear filtering using branching and interacting particle systems," Markov Processes Related Fields, vol. 5, no. 3, pp. 293–319, 1999.
Deguchi et al 2004	K. Deguchi, O. Kawanaka and T. Okatani, Object tracking by the mean-shift of regional colour distribution combined with the particle-filter algorithm, Proc. ICPR (2004) 506-509.
Deutscher et al 2000	J. Deutscher, A. Blake, and I. Reid, Articulated body motion capture by annealed particle filtering, Proc. IEEE Conf. Computer Vision Pattern Recognition (2000).
Dornaika and Ahlberg 2004	Fadi Dornaika, Jörgen Ahlberg: Model-Based Head and Facial Motion Tracking. ECCV Workshop on HCI 2004: 221-232

Doucet et al 2001	A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," Statist. Comput., vol. 10, no. 3, pp. 197–208.
Doucet et al 2001	A. Doucet, J. F. G. de Freitas, and N. J. Gordon, "An introduction to sequential Monte Carlo methods," in Sequential Monte Carlo Methods in Practice, A. Doucet, J. F. G. de Freitas, and N. J. Gordon, Eds. New York: Springer-Verlag, 2001.
Edwards et al. 1998	Edwards, G., Taylor, C., and Cootes, T. 1998. Interpreting face images using active appearance models. In International Conference on Face and Gesture Recognition. 300–305.
Ellis 2002	T. Ellis, "Performance Metrics and Methods for Tracking in Surveillance", Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, June, Copenhagen, Denmark, 2002, pp26-31.
Evans and Naeem 2007	Evans, D. and Naeem, A. (2007). "Using visual tracking to link text and gesture in studies of natural discourse", Online Proceedings of the Cross Disciplinary Research Group Conference 'Exploring Avenues to Cross-Disciplinary Research', November 7, University of Nottingham.
Evans et al 2007	Evans, D. and Naeem, A. (2007). "Using Visual Tracking to Link Text and Gesture in Studies of Natural Discourse", 3rd International Conference on e-Social Science, October 2007, Univ. of Michigan Ann Arbor, USA: ESRC/NSF.
Fieguth and Terzopoulos 1997	Fieguth, P. and Terzopoulos, D. 1997. Colour-based tracking of heads and other mobile objects at video frame rates. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 21–27.
French et al 2007	Andy French, Asad Naeem, Ian Dryden, Tony Pridmore , "Using social effects to guide tracking in complex scenes", AVSS 2007 London
French et al 2008	French, A., Bennett, M., Howells, C., Patel, D. and Pridmore, T. 2008. A probabilistic approach to root measurement in images. International Conference on Bio-inspired systems and Signal processing, Madeira, Portugal, 2008.
Gordon et al 1993	N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to nonlinear and non-Gaussian Bayesian state estimation," Proc. Inst. Elect. Eng., F, vol. 140, pp. 107–113, 1993.
Green et al 2005	Green, J., Pridmore, T.P., Benford, S.D. and Ghali, A., 2004. Location and recognition of flashlight projections for visual interfaces. In: Proceedings of the 17th International Conference on Pattern Recognition, 23-26 August 2004.
Grenander et al 1991	Grenander, N., Chow, Y., and Keenan, D. M. (1991) HANDS. A Pattern Theoretical Study of Biological Shapes. Springer-Verlag. New York
Grzegorz et al 2007	Grzegorz Cielniak, Tom Duckett and Achim J. Lilienthal, "Improved Data Association and Occlusion Handling for Vision-Based People Tracking by Mobile Robots", IEEE/RSJ 07, International conference on intellegent robots and systems, Sandiago, CA, USA, November 2007"
Hahn and Duncan 2006	Daniel V. Hahn, Donald D. Duncan "Digital Hammurabi: design and development of a 3D scanner for cuneiform tablets"Three-Dimensional Image Capture and Applications VII. Edited by Corner, Brian D.; Li, Peng; Tocheri, Matthew. Proceedings of the SPIE, Volume 6056, pp. 130-141 (2006).
Horn and Schunk 1981	B.K.P. Horn and B.G. Schunk, "Determining optical flow." Artificial Intelligence, vol 17, pp 185-203, 1981
i-LIDS 2007	Home Office Scientific Development Branch, Imagery library for intelligent detection systems (i-LIDS), http://scienceandresearch.homeoffice.gov.uk/hosdb/cctvimaging- technology/video-based-detection-systems/i-lids/
Isard and Blake 1998a	M. Isard and A. Blake, "Condensation -- conditional density propagation for visual tracking," International Journal of Computer Vision 29(1), pp. 5--28, 1998.
Isard and Blake 1998b	M. Isard and A. Blake, "ICondensation: Unifying low-level and high-level tracking in a stochastic framework", *Proc 5th European Conf. Computer Vision*, 1, pp 893-908, 1998
Isard and Blake 1998c	A Mixed state condensation tracker with automatic model switching. Proc 6th Int. Conf. Computer Vision 107-112.
Julier and Uhlmann 1997	S. J. Julier and J. K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL, 1997.
Kalman 1960	Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems," Transactions of the ASME - Journal of Basic Engineering Vol. 82: pp. 35-45 (1960).
Kanazawa et al 1995	K. Kanazawa, D. Koller, and S. J. Russell, "Stochastic simulation algorithms for dynamic probabilistic networks," in Proc. Eleventh Annu. Conf. Uncertainty AI, 1995, pp. 346–351.

Kerhet et al 2007	Aliaksei Kerhet, Francesco Leonardi, Andrea Boni, Andrea Boni, "Distributed Video Surveillance Using Hardware-Friendly Sparse Large Margin Classifiers" AVSS 2007 London.
Khan et al 2003	Z. Khan, T. Balch, and F. Dellaert, Efficient Particle Filter-Based Tracking of Multiple Interacting Targets Using an MRF-based Motion Model, Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03), 2003.
Khan et al 2004	Khan, Z., Balch, T., and Dellaert, F., An MCMC-based particle filter for tracking multiple interacting targets, Proc. ECCV 2004, Prague, Czech Republic (2004).
Knight et al 2008	Brundell, P., Knight, D., Tennent, P., Naeem, A., Adolphs, S., Ainsworth, S., Carter, R., Clarke, D., Crabtree, A., Greenhalgh, C., O'Malley, C., Pridmore, T. and Rodden, T. "The Experience of using Digital Replay System for social science research" , NCeSS 2008, 19th June Manchester
Leung and Gong 2006	A.P. Leung and S. Gong, Mean Shift tracking with random sampling, Proc. BMVC 2005, pp.729-738, 2006.
Link 1	http://www.iis.ee.ic.ac.uk/~mpsha/ludwig/Video.html. Visited on Sept. 15th 2006,
Link 2	http://www.cs.nott.ac.uk/~azn/single_trgts_videos.htm
Link 3	http://www.cs.nott.ac.uk/~azn/multi_trgts_videos.htm
Lowe 2001	David G Lowe, Local Feature View Clustering for 3D Object Recognition, 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01), Vol. I (2001), pp. 682-688.
MacCormick and Blake 2000)	MacCormick, J. and Blake, A., A probabilistic exclusion principle for tracking multiple objects, International Journal of Computer Vision, 39, 1 (2000) 57-71.
Maggio and Cavallaro 2005	E. Maggio and A. Cavallaro, "Hybrid particle filter and Mean Shift tracker with adaptive transition model", *Proc. Int. Conf. Acoustics, Speech, and Signal Processing* 2005
McNeill 1992	McNeill, D. (1992) *Hand and Mind* Chicago: The University of Chicago Press
McNemar, 1947	McNemar Q. Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika, 1947, 12, 153-157.
Milstein et al 2002	A. Milstein, J.N. Sanchez, and E.T. Williamson, Robust global localisation using clustered particle filtering, Proc. AAAI (2002) 581-586.
Mortimer et al 2001	Mortimer D and Menkveld R (2001). Sperm morphology assessment--historical perspectives and current opinions. J Androl 2001 22, 192-205.
Musso and Oudjane 2001	C. Musso, N. Oudjane, and F. LeGland, "Improving regularized particle filters", in Sequential Monte Carlo Methods in Practice, A. Doucet, J.F de Freitas, and N.J. Gordon, Eds. Springer-Verlag. New York, 2001.
Naeem et al 2006	A. Naeem, S. Mills, and T. Pridmore, Structured Combination of Particle Filter and Kernel Mean-Shift Tracking, Proc. IVCNZ, Gt. Barrier Island, 2006
Naeem et al 2007	A. Naeem, T. Pridmore, and S.Mills, Managing particle Spread via Hybrid Particle Filter/Mean Shift Tracking. *In:* Proceedings of the British Machine Vision Conference. BMVA Press, 2007.
Nascimento et al 2005	J. Nascimento, J. Marques, "Performance evaluation of object detection algorithms for video surveillance", IEEE Transactions on Multimedia, 2005, pp761-774.
Needham and Boyle 2003	C.J. Needham, R.D. Boyle. "Performance Evaluation Metrics and Statistics for Positional Tracker Evaluation" ICVS 2003, Graz, Austria, April 2003, pp. 278 - 289.
Nickel and Stiefelhagen 2007	Kai Nickel, Rainer Stiefelhagen, "Visual recognition of pointing gestures for human-robot interaction" IVC , Volume 25 , Issue 12 Pages 1875-1884, 2007
Nixon and Carter 2004	Nixon, M. S. and Carter, J. N. (2004) Advances in Automatic Gait Recognition. In: IEEE Face and Gesture Analysis 2004, FG04, 2004, Seoul Korea. pp. 11-16.
Ohlander et al 1978	Ron Ohlander, Keith Price, and D. Raj Reddy (1978): "Picture Segmentation Using a Recursive Region Splitting Method", Computer Graphics and Image Processing, volume 8, pp 313-333

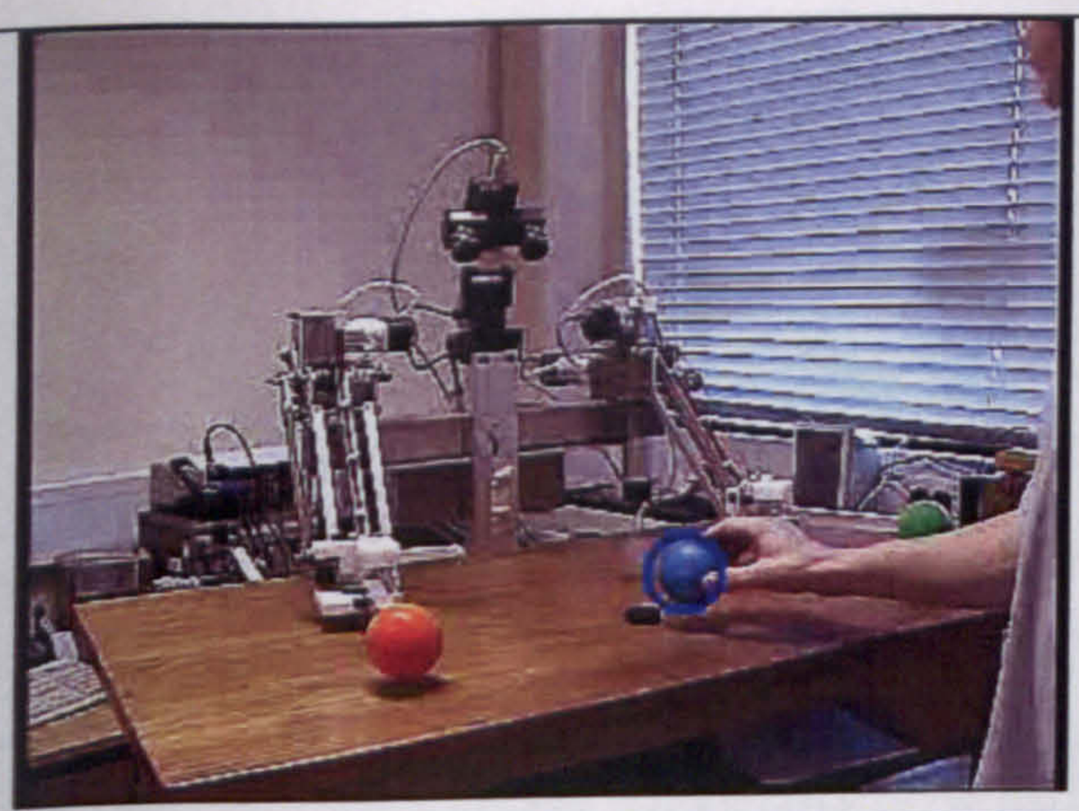
Okuma et al 2004	Kenji Okuma, Ali Taleghani, O De Freitas, James J. Little, David G. Lowe , "A boosted particle filter: Multitarget detection and tracking", ECCV 2004
Papoulis, A. 1990	Papoulis, A. (1990) Probability and statistics. Prentice-Hall
Park 2005	Jong-Seung Park, Interactive 3D reconstruction from multiple images: a primitive-based approach, Pattern Recognition Letters, v.26 n.16, p.2558-2571, December 2005
Pathegama and Gol 2004	Mahinda Pathegama & Ö Göl (2004): "Edge-end pixel extraction for edge-based image segmentation", Transactions on Engineering, Computing and Technology, vol. 2, pp 213-216, ISSN 1305-5313
Perš and Kovačič 2000	J. Perš, S. Kovačič, "A System for Tracking Players in Sports Games by Computer Vision." Electrotechnical Review - Journal for Electrical Engineering and Computer Science, 67(5):281-288, 2000
PETS Dataset 2007	PETS data sets, http://www.cvg.rdg.ac.uk/slides/pets.html, last visited 8th August 2008.
Pitt and Shephard 1999	M. Pitt and N. Shephard, "Auxiliary particle filters" J. Amer. Statist. Association, 94(446), pp. 590-599, 1999.
Porikli and Tuzel 2005	F. Porikli and O. Tuzel, Multi-kernel object tracking, Proc. IEEE Int. Conf. on Multimedia (2005) 1234-1237.
Ratnalingam et al 2006	R.A. Ratnalingam, D. Coates, J. Leong, D.O. Kessel, D. Magee, Y. Zhu, Computer assisted simulation of ultrasound guided procedures , Proc. Cardiovascular and Interventional Radiological Society of Europe Conference,, pp64, 2006
Reid 1979	D Reid, "An algorithm for tracking multiple targets", IEEE Trans. Automatic Control 1979.
Ristic et al 2009	B. Ristic, S. Arulampalam, N. Gordon, Beyond the Kalman Filter: Particle Filters for Tracking Applications, 2009
Salih et al 2004	Q. A. Salih, A. R. Ramli, R. Mahmud & R. Wirza : 3D Visualization For Blood Cells Analysis Versus Edge Detection . The Internet Journal of Medical Technology. 2004 Volume 1 Number 2
Serby et al 2004	Serby, D., Koller-Meier, S., and Gool, L. V. 2004. Probabilistic object tracking using multiple features. In IEEE International Conference of Pattern Recognition (ICPR). 184–187.
Shan et al 2007	C. Shan, T. Tam and Y. Wei, Real-time hand tracking using a Mean Shift embedded particle filter, Pattern Recognition, 40 (2007) 1958-1970.
Shi et al 1997	Jianbo Shi and Jitendra Malik (1997): "Normalized Cuts and Image Segmentation", *IEEE Conference on Computer Vision and Pattern Recognition*, pp 731-737
Somes, 1983	Somes G. McNemar test. Encyclopedia of statistical sciences, vol. 5, S. Kotz & N. Johnson, eds., pp. 361-363. New York: Wiley, 1983.
Starner and Pentland 1995	T. E. Starner and A. Pentland, "Visual Recognition of American Sign Language Using Hidden Markov Models," Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition, pp. 189-194, 1995.
Suyu et al 2007	S. Kong, C. Sanderson, B.C. Lovell. Classifying and Tracking Multiple Persons for Proactive Surveillance of Mass Transport Systems. IEEE Conference on Advanced Video and Signal based Surveillance, London, 2007.
Suyu et al 2008	Kong, S., Bhuyan, M. K., Sanderson, C. and Lovell, B. C. (2009). Tracking of Persons for Video Surveillance of Unattended Environments. In: 19th International Conference on Pattern Recognition (ICPR) 2008, Tampa, FL, USA, (1-4). 8-11 December 2008.
Tennant R. 1998	Tennant, Richard (1998). The American Sign Language Handshape Dictionary. Clerc Books. ISBN 1563680432.
Veenman et al 2001	Veenman, C., Reinders, M., and Backer, E. 2001. Resolving motion correspondence for densely moving points. IEEE Trans. Patt. Analy. Mach. Intell. 23, 1, 54–72.
Vermaak et al 2003	J. Vermaak, A. Doucet and P.Perez, Maintaining multi-modality through mixture tracking, Proc. ICCV, pp 1110-1116, 2003
Viola and jones 2001	Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE Conference on Computer Vision and Pattern Recognition (2001)

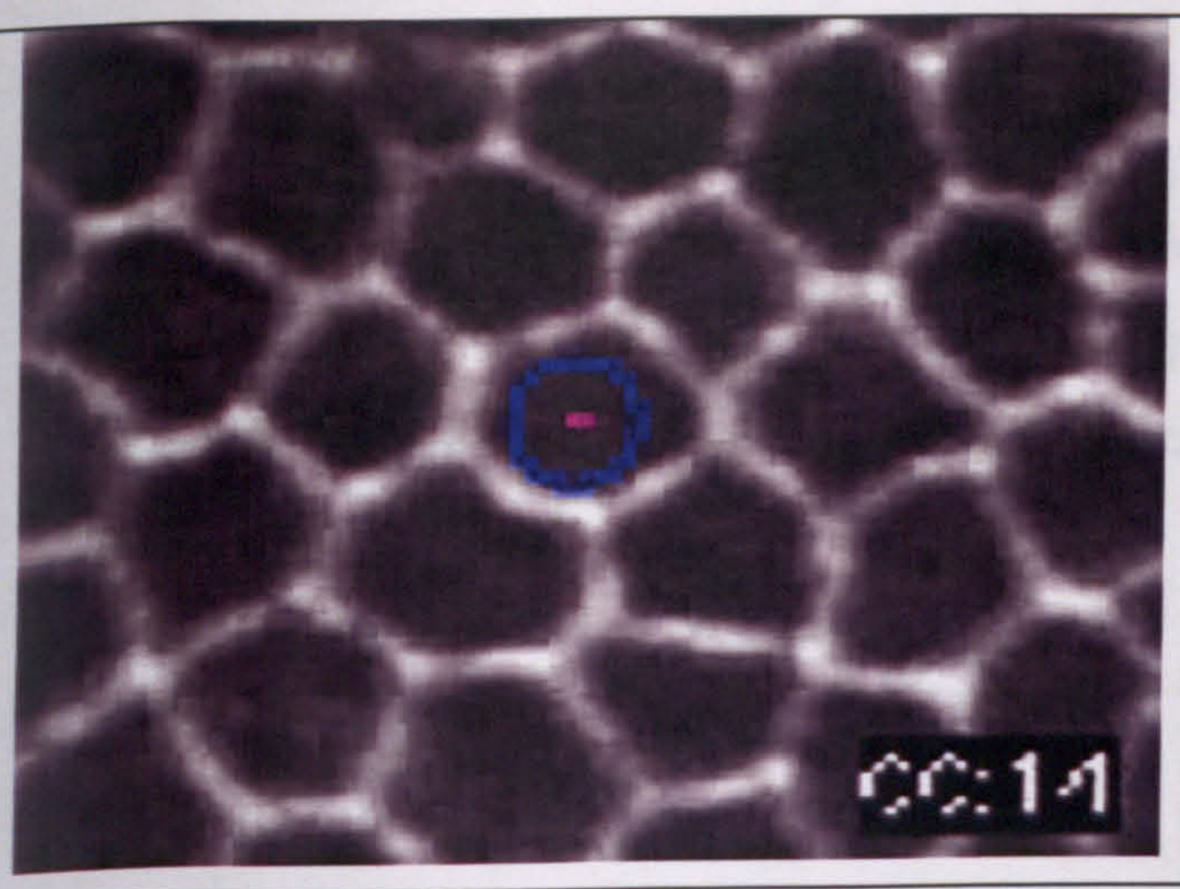
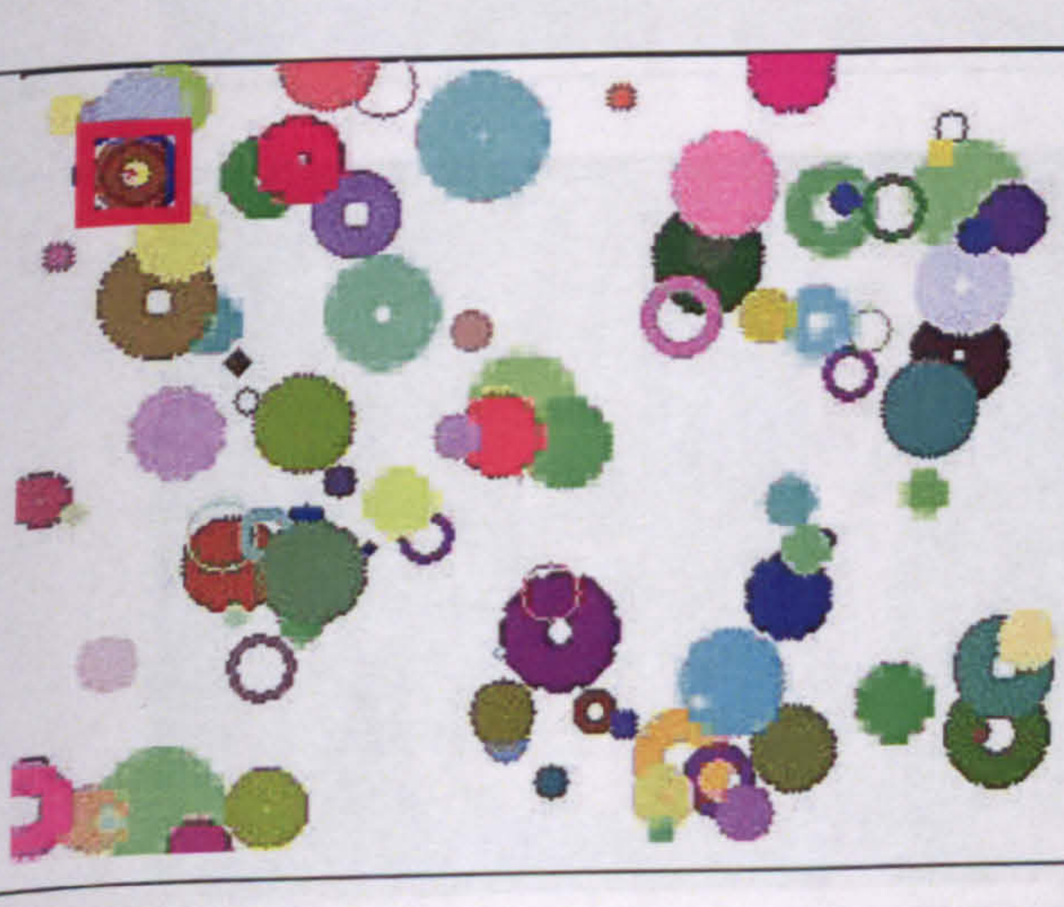
Viola and Jones 2003	M. Jones and P. Viola. Face recognition using boosted local features. In Proceedings of International Conference on Computer Vision, 2003
Wallace A. M. 1988	A.M. Wallace, "Industrial applications of computer vision since 1982", IEE Proceedings 135(3), 117-136 (1988).
Wan and Merwe 2001	E. A. Wan and R. van der Merwe, Kalman Filtering and Neural Networks, chapter 7 - The Unscented Kalman Filter, Wiley, 2001.
Wang et al 2007	Z. Wang, X. Yang, Y. Xu and S. Yu, CamShift guided particle filter for visual tracking, Proc. IEEE Workshop on Signal Processing Systems (2007) 301-306.
Welch and Bishop 2001	Greg Welch and Gary Bishop. An Introduction to the Kalman Filter, SIGGRAPH 2001 course 8. In Computer Graphics, Annual Conference on Computer Graphics & Interactive Techniques. ACM Press, Addison-Wesley, Los Angeles, CA, USA (August 12–17), 2001.
Yang et al 2005	C. Yang, R. Duraiswami and L. Davis, Efficient Mean Shift tracking via a new similarity measure, IEEE Conf. on Computer Vision and Pattern Recognition, 1 (2005) 176-183.
Yilmaz et al 2004	YILMAZ, A., LI, X., AND SHAH, M. 2004. Contour based object tracking with occlusion handling in video acquired using mobile cameras. IEEE Trans. Patt. Analy. Mach. Intell. 26, 11, 1531–1536.
Yilmaz et al 2006	Yilmaz, O. Javed and M. Shah, Object tracking: a survey, ACM Computing Surveys, 38, 4 (2006) 1-45.
Yin et al 2007	Fei Yin, D. Makris, S.A. Velastin, "Performance Evaluation of Object Tracking Algorithms", 10th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS2007), October, Rio de Janeiro, Brazil, (2007) Using Synthetic Ground Truth to Evaluate Computer Vision Techniques
Zhao and Nevatia 2004	T. Zhao and R. Nevatia, Tracking multiple humans in crowded environment, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (2004) 406-413.

Appendix A

This describes and provides a selection of sample frames from a set of test image sequences used through out this thesis to assess the performance of algorithms that track single targets. A short description of each image sequence is given in the table.

Video Frame	Description
	A tiger sprints through the forest, with trees occluding it partially. The face is tracked.
	A table tennis ball is bounced on a racket and tracked.
	A basketball player makes quick manoeuvres with the ball occluding his head twice. The head is tracked.

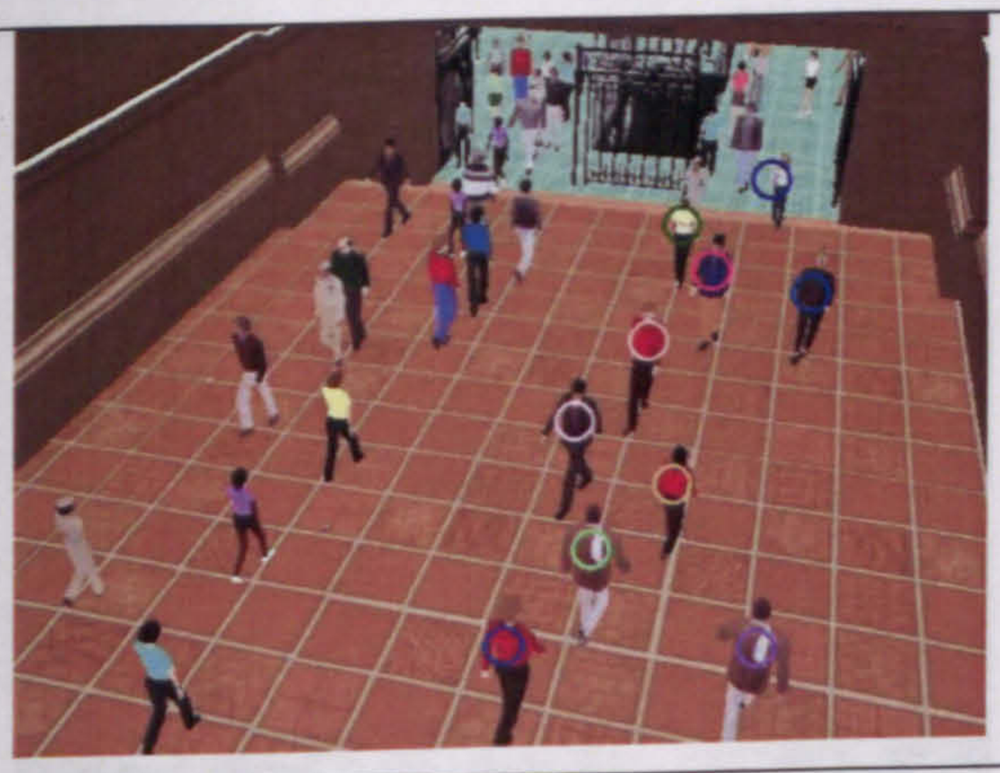
	The blue ball is tracked while the hand moves it around in the scene.
	The football is tracked while it is kicked around by the boy. The camera is not fixed and moves as well. The ball gets occluded by the legs a couple of times. The camera also zooms in and out a little.
	The girl is tracked while she runs and jumps around in the park. The camera is not stationary and moves with the girl as well.

	A single cell is tracked while they all move and change shapes.
	A girl is tracked while she runs around occluded once by the man. The camera is hand-held and not stationary.
	A ball is tracked in this artificial video while it moves randomly through the clutter. The clutter is static.

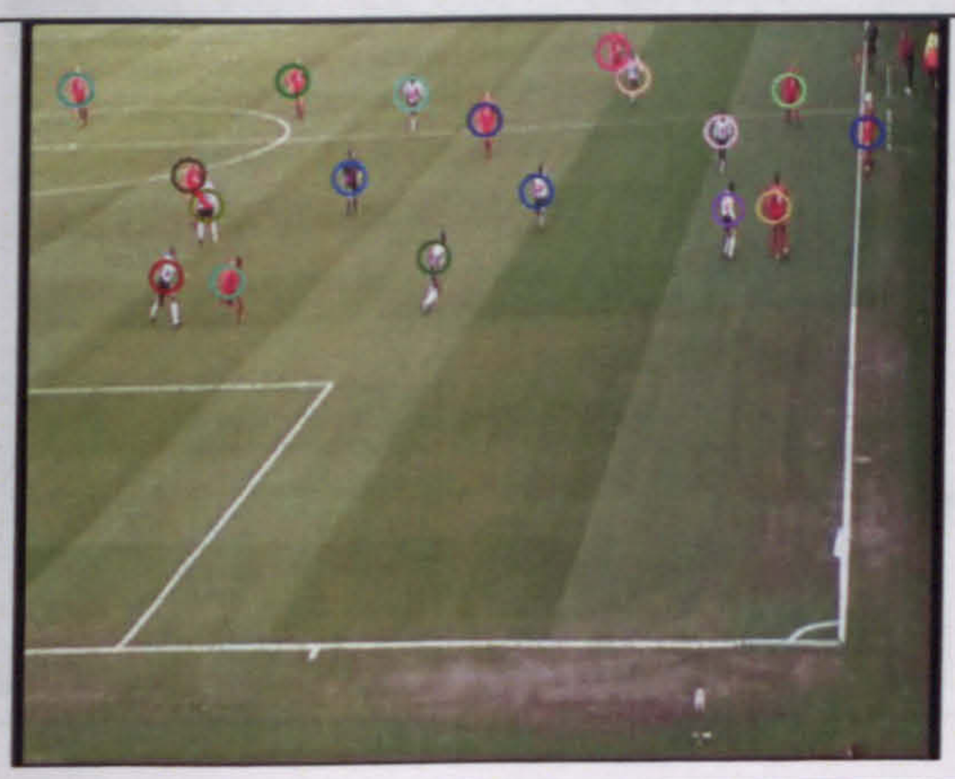
Appendix B

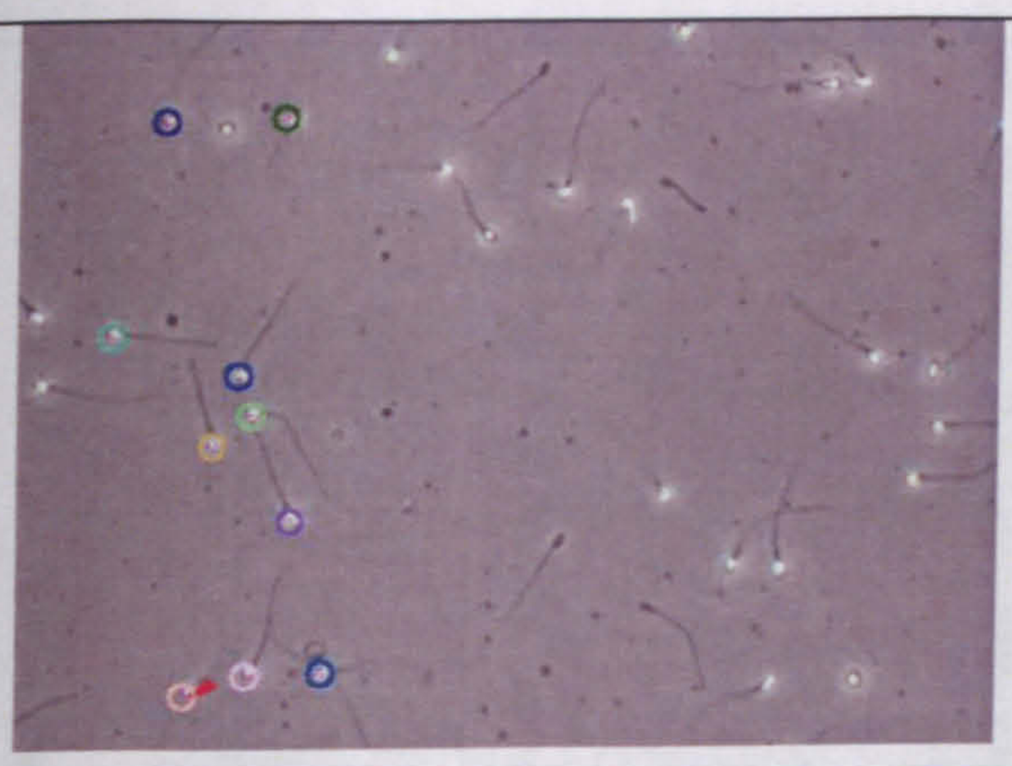
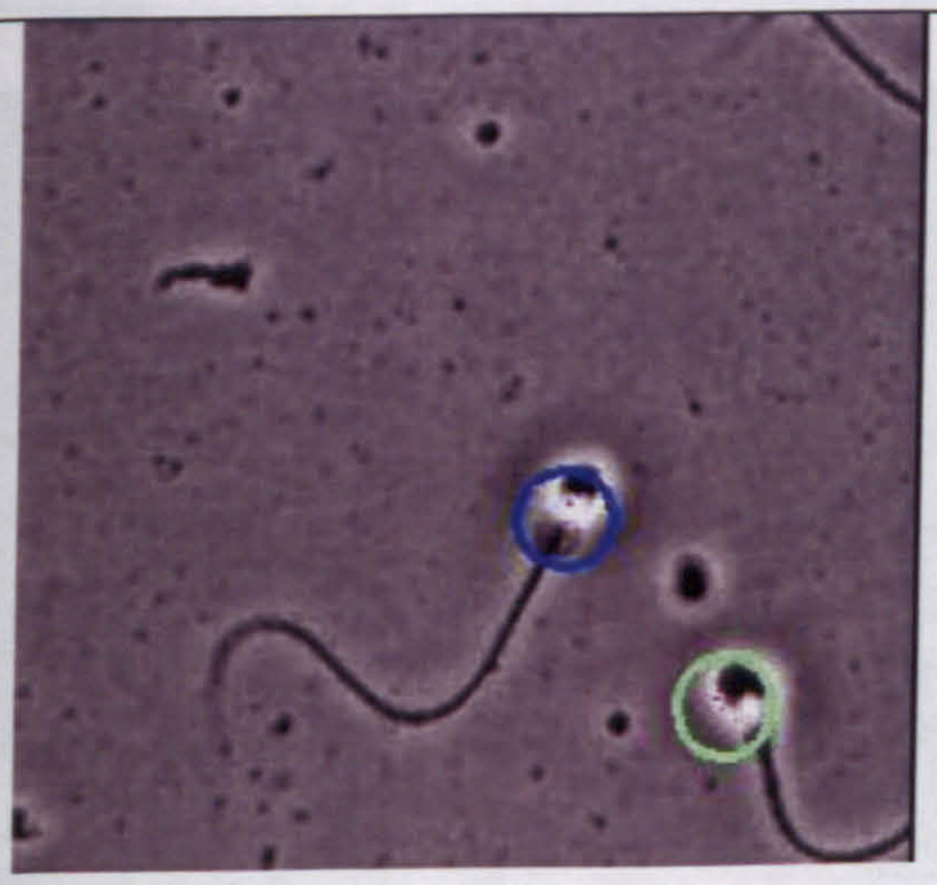
This describes and shows some selected sample frames from a set of test image sequences used to assess the performance of algorithms that track multiple objects. The number of targets and a short description of each image sequence are given in the table.

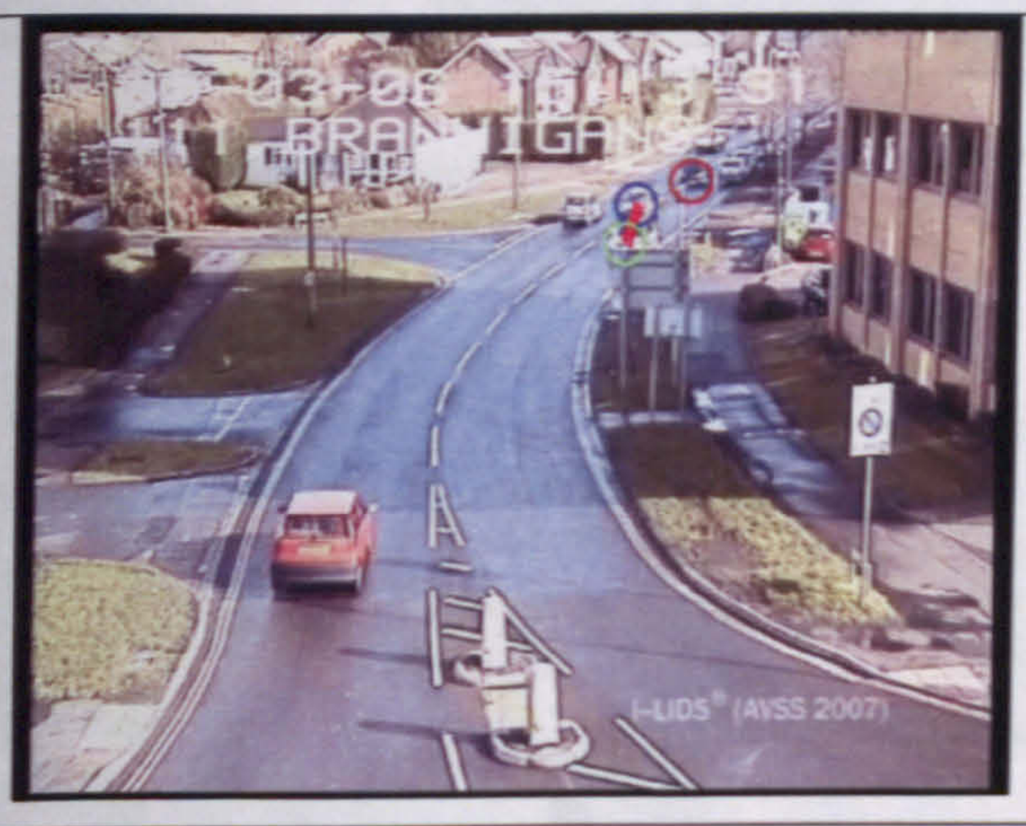
Video Frame	Number of Objects Tracked	Description
	5	Head movement is recovered by tracking eyes and mouth. Hands are also tracked while the subject explains how fishing net was thrown in a Vietnamese river.
	3	The host's hands and head are tracked while he waves them around very quickly with various hand gestures.

	5	5 people are tracked while moving away from the camera in a synthetic train station simulation.
	9	People moving in various directions causing collisions and occlusions in a simulated train station.
	10	10 people in various positions like sitting, walking, stopping, chatting and then walking are tracked.

	8	8 People moving in a mall are tracked in this synthetic video.
	2	2 people at a train station are tracked while they occlude each other having similar colour models to an extent, this was taken from PETS video data set.
	5	5 people are tracked while they move in the scene, causing occlusions, taken from PETS video dataset.
	7	7 football players are tracked in this high pace scene with occlusions and collisions with a moving camera.

	17	All players are tracked with a fixed camera while the game commences.
	3	The three players are tracked particularly because they form a close tackle occluding and colliding with each other.
	3	3 person's heads are tracked while moving in a train station, they move towards the train occluding each other both completely and partially.

	4	Fast hand movements are tracked during a PhD supervision meeting.
	10	10 sperm are tracked while they swiftly move in the scene causing collisions and occlusions.
	2	2 sperm are tracked up-close while colliding with and occluding each other.

	4	Eyes of both kids are tracked as they move around.
	3	3 cars are tracked while they move towards the camera, this was taken from PETS video data set.
	6	Eyes and mouths are tracked during a PhD supervision meeting.

	2	Head of two people are tracked during a bumping car drive.
	2	Two kids' heads are tracked during a funfair ride.