Probabilistic Graphical Models for Visual Tracking of Objects

Vijay Badrinarayanan

Project VISTA, INRIA Rennes Bretagne-Atlantique

University of Rennes 1

Abstract

This thesis puts forth graphical models for visual tracking in low and higher dimensional state spaces. For low dimensional tracking problems, such as object position tracking, a novel message switching/combination idea is introduced. Based on this concept and a new pseudo simulation viewpoint of point trackers a novel randomized feature point filter is developed. The message switching/combination ideas are then extended to construct multi-cue fusion based tracking with a set of simulation based filters. Employing pseudo simulation based point trackers and color based particle filters as their elementary filters these multi-cue fusion schemes track general objects in complex scenarios.

Moving to higher dimensions, general multi-part tracking schemes are introduced. A network of local patch trackers are put to play in a stochastic simulation framework to track the position and other attributes of arbitrary objects. The difficult task of updating the process prior online is also performed under this simulation framework. Dealing with online update of the process prior enlarges the scope of application of the multi-part tracking model. A simple interactive multi-part tracking scheme is also discussed in this context.

To the extent permitted by practicality, the contributions are evaluated quantitatively and/or qualitatively to convince the reader of their novelty, improvements and/or robustness. Detailed discussions of the highlights and drawbacks of the models are presented. Prospective extensions of the models based on empirical arguments and reflections on design are included.

Contents

List of Figures

List of Tables

Glossary

E	Edges on a graph.
$I_{0:n}$	Sequence of image data.
$V, \mathcal{V}$	Vertices on a graph.
X	Represents a set of hidden state random variable.
Y	Represents a set of instantiated/measured/observed random variables or data.
$\Psi(x_1, \ldots, x_n)$	The compatibility function between variables $x_1, \ldots, x_n$. It can be normalized (sum to unity) or unnormalized.
$\mathcal{N}(x; \mu, \Sigma)$	Gaussian distribution in random variable x, with mean parameter μ and co-variance parameter Σ.
θ	Represents a parameter controlling the prior probability distribution.
$\widetilde{w}$	Normalized importance sample weight.
e	Represents evidence, which can be the value of an instantiated random variable, measured or observed data.
$l(y\|x) \propto p(y\|x)$	The unnormalized likelihood function of x.
$m_{x_i \to x_j}(x_i)$	Message sent from hidden state x_i to hidden state x_j.
$p(x\|\theta)$	The prior probability distribution of x, parameterized by θ.
$p(x_{0:n}\|y_{1:n})$	The posterior distribution or simply, posterior of $x_{0:n}$.
$p(x_n\|y_{1:n-1})$	The effective prior distribution of x_n.
$p(x_n\|y_{1:n})$	The filtering distribution of x_n.
$p(y\|\theta)$	The evidence of prior model parameter θ. Equivalently, the likelihood function of θ.
$p(y\|x)$	The likelihood function of x, also termed the hidden data likelihood or model evidence.
$p(y_n\|x_{n-1})$	The predictive likelihood function of x_{n-1}.
$p(y_n\|y_{1:n-1})$	The measurement predictive distribution of y_n.
w	Unnormalized importance sample weight.
x	Represents a hidden state random variable, can be discrete or continuous.
$x_{0:n}$	Sequence of hidden random variables.
y	Represents an instantiated random variable, can be discrete or continuous. It is also termed the observed or measured data.
$y_{1:n}$	Sequence of observed/measured data.
BP	Belief propagation.
C	Represents a clique on a graph.
CPF	Color based particle filter.
DAG	Directed acyclic graph.
DP	Dynamic programming.
HMM	Hidden Markov model
IR	Importance resampling.
IS	Importance sampling.
LLN	Law of large numbers.
MAP	Maximum a posteriori.
MC	Monte Carlo.
MCMC	Markov chain Monte Carlo.
MLE	Maximum likelihood estimation.
NCC	Normalized cross correlation.

RFT-filter	Randomized feature point tracking filter.
SMC	Sequential Monte Carlo.
SSD	Sum of squared distances.
Z	Represents the partition function of a probability distribution.

1

General Introduction

1.1 Visual Tracking

Visual tracking is the problem of exploiting space-time consistencies of object motion and evolution to continuously determine the "state" of an object with additional aid from measurements extracted from a sequence of images capturing the object. The state could be the position with respect to some reference, size changes with respect to some initial known size, its three dimensional (3D) position or even a precise delineation of its contours. Wading through the vast literature on visual tracking, one cannot help but question the *relevance* and *state of the art* of this problem. From a system point of view, it is often argued that visual tracking is not a necessary part of a vision based system for video analysis. For instance, in a motion capture system, it can be argued that a tracking module may, in principle, be eliminated and replaced by a successive body pose detector module supported by a large computational power. Saying this, it would be an enormous waste to let go very useful kinematical cues of general objects for the very same task, which if benefitted from, can save a lot of computational power. So a visual tracker could intervene and save on the expensive task of independent successive analysis.

From a methodical or algorithmic point of view the developments in visual tracking frameworks have had considerable impact on several other problems in computational vision, notably the resurgence of Sequential Monte Carlo methods, which are now pervasive in problems like image and video segmentation. Active contours are another striking example. Likewise visual tracking has also borrowed considerably from developments in other vision related research or external fields and amalgamated them

to help serve the need for a sophisticated unobtrusive visual analysis tool. It is becoming increasingly common to employ visual tracking as an indispensable low level visual analysis tool for digital cinema post production tasks like color correction, Human Computer Interfaces for video game control, determining human body poses from video sequences for augmented virtual reality applications and to create graphic reincarnations (avatars), all at a small cost. Its implications in the area of security and surveillance are obvious. Therefore, to make a statement, visual tracking is now ubiquitous and its relevance in computational vision cannot be over emphasized.
The state of the art in visual tracking methods and systems is difficult to summarize over a few words. Over the years, beginning from the problem of point tracking it has diversified into several distinct paths of research. In the current day visual tracking includes problems like tracking positions of regions or objects, three dimensional poses of objects and humans, precise delineation of contours of deformable objects and special motion fields created by atmospheric flow or fluids. Each of these problems present a considerable challenge for research and even though there have been numerous attempts to meet these challenges, there remain several shortcomings which are driving current research. A more detailed overview of literature in these directions is presented in chapter 2 to give the reader a flavour of the problems involved and the outstanding frameworks which have been developed to tackle them.

1.2 Context and Contributions

The context in which this work was carried out was partly industrial and partly academic. At conception, the idea was to manouevre the research towards serving the needs of the digital cinema post production industry, the television broadcast production industry and for applications in retail networks. On hindsight, the contributions from this thesis can be justly regarded as serving the broad needs of the applications in these domains and even extending beyond the expected borders. The contributions from this thesis to visual tracking is briefly enumerated below.

1. **Probabilistic fusion of point trackers**
   The correlation based point tracking technique is presented as a pseudo simulation filter with a view to incorporate it in a Bayesian framework. Based on this model of point tracking, a message switching/combination scheme is introduced for probabilistic inference. Then on, a graphical model for the fusion of multiple

point trackers using message switching is presented and a novel tracker, termed the *randomized feature point tracker*, is derived from it. This tracker, constructed from a set of point trackers behaves like a simulation filter propagating abitrary filtering distributions.

2. **Probabilistic graphical models for message switching**
   A probabilistic graphical model is put forth for multi-cue fusion based tracking. On this model, the filtering distribution of the tracked state is approximated by choosing between incoming messages from one of several tracking nodes. The choice of which message is appropriate for arriving at the filtering distribution can be controlled by external parameters. More generally this model can also be used to linearly combine messages from multiple tracking nodes, with the combination weights determined *a posteriori*.
   To demonstrate the benefits of such a model, a novel multi-cue tracker is developed, which is capable of tracking generic objects like faces and vehicles in unconstrained environments. Specifically, a color based particle filter and a set of randomized feature point trackers are the elements used to make the probabilistic machinery work. The superiority of this tracker is adjudged by a test with several standard trackers in literature.

3. **Probabilistic graphical model for multi-part object tracking**
   Moving into a higher state space dimension, a Gaussian Markov Random Field (GMRF) model is utilised for tracking an arbitrary shaped spatial layout imposed on an object with multiple local patch trackers. The motivation for such tracking is to digress from rectangular box, ellipse or standard geometrical shape tracking and track moderately deformable layouts which can be used to deliver rough cut-outs of objects or produce a seed for batch based video segmentation. Another aspect in favour of multi-part layout tracking is that some parts can be updated or replaced online to act as an adaptation of the appearance model of the whole target.
   In numerous applications involving visual tracking some small user aid goes a long way in creating useful tracks. The proposed multi-part layout tracking is now leveraged in an interactive setup. The user is allowed to control the prior on the MRF model, which in intuitive terms simply determines how much the layout structure is allowed to change. It is quite difficult to establish priors on generic objects and layouts especially in the presence of strong motions in

three dimensions and scaling. Therefore, banking on a reasonable and intuitive interactive effort for adapting the prior is shown to be particularly efficient.

1.3 Organization of the manuscript

A taxonomic overview of the state of the art is provided in chapter 2. Under each *genre*, leading methods and frameworks, which have influenced numerous other methods are briefly discussed along with references to very recent propositions. Next, Chapter 3 provides an introduction to the probabilistic inference problem with detailed discussions on the elementary components. Readers unfamiliar with probabilistic inference concepts are advised to read through this introduction before moving on to other chapters. Chapter 4 is dedicated to probabilistic models for point tracking, in particular, models for single point tracking and concerted tracking with multiple points. Probabilistic graphical models for message switching/combination is then introduced. Subsequently, the novel randomized feature point tracker (RFT-filter) is derived and experimented with. The construction of a novel multi-cue fusion tracker based on the message switching/combination model is dealt with in detail in chapter 5. Results of qualitative and quantitative experiments are presented along with discussions on the pros and cons of the propositions. Multi-part geometric layout based tracking is introduced in chapter 6 with relevant probabilistic models and accompanying vision based elements used to put the probabilistic machinery to work. Sample results and detailed discussions of the advantages and drawbacks of the proposed algorithm are discussed. The case for interactive multi-part tracking with local patches and color models is then presented. The elements of user interaction are introduced. The results of interactive tracking are displayed and prospects arising from this technique is then discussed. General conclusions of the thesis are drawn in Chapter 7. Some prospective extensions of the graphical models are discussed in Chapter 8.
At the beginning of each chapter a focussed literature review pertinent to the contents of that chapter is presented. A recapitulative summary of the salient points discussed in the chapter and, where appropriate, motivation for the future chapters are also provided. To avoid frequent digression from the essence of the discussion appendices are provided to revise standard results and elucidate tangential points.

2

Overview of the literature

2.1 The nature of tracking algorithms

The key point which springs to mind about visual tracking algorithms is their non-universality, that is to say, there is no one tracking method which is currently able to track in high dimensional spaces (contours or articulated bodies for instance) very accurately over very long durations in unconstrained environments. It is becoming increasingly clear that a method capable of achieving this goal would be a technique incorporating several heterogeneous noisy measurements and multiple kinematic priors to deliver accurate tracks over long durations. In the process such a technique would ideally overcome difficult challenges like, dealing with noisy measurements, appearance changes of the target, partial and complete occlusions. This is indeed a difficult goal, nevertheless, the entire corpus of research in visual tracking is marching along in search of this goal. Meanwhile the vast body of existing literature ramifies into several techniques depending on the capabilities of the techniques, their view point of the tracking problem, the nature of their measurements and the characteristics of the application scenario for which they are designed. In tune with this fact, the review of the state of art is based on a taxonomical classification of the existing techniques. It is insisted beforehand that such a classification, although describing several important genres under which tracking algorithms can be classified, is by no means all encompassing. It would not be entirely incorrect to say the taxonomy is based on the subjective view of its author. Finally, this chapter provides only an overview of salient techniques in literature. The reader is provided with a more detailed review of salient techniques at relevant points in the forthcoming chapters.

2.2 Taxonomy of visual tracking

Visual tracking techniques may be broadly classified under one of the following genres.

2.2.1 Dimensionality of the state space

1. **Point tracking**

   Point tracking is a misnomer, it is usually a small patch of image around a specified point which is tracked. Nevertheless, overlooking the terminology, tracking the position of a point in a video sequence is one of the most fundamental low dimensional state space tracking problems. This is sometimes referred to as feature or feature point tracking. The early Kanade-Lucas-Tomasi (KLT) point tracker of [Tomasi and Kanade, 1991] is now a classic. The authors [J.Shi and C.Tomasi, 1994] propose not only a method for feature point tracking but in addition, techniques to select "good" features to track and monitor the quality of feature point tracking. Tracking is posed as an optimization problem to derive optimal motion model parameters. The cost $J(A,d)$ as a function of the affine motion parameters A and displacement d of a feature is taken as follows.

   $$J(A,d) = \int\int_{x\in W} \left[\mathcal{I}_{n+1}(Ax+d) - \mathcal{I}_n(x)\right]^2 w(x)\,dx, \qquad (2.2.1)$$

   where $\mathcal{I}_n, \mathcal{I}_{n+1}$ are consecutive images and $w(x)$ is a weighting function used to emphasize the central area inside the feature window W. $d = [d_x, d_y]$. A is defined below.

   $$A = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \qquad (2.2.2)$$

   The task now is to minimize the above cost function to retrieve the optimal motion model parameters A, d. Using a first order Taylor expansion around point x of $\mathcal{I}_{n+1}(Ax+d)$, the optimization problem is reduced to solving a linear equation,

   $$Tz = a, \qquad (2.2.3)$$

   with $z = [d_{xx}, d_{yx}, d_{xy}, d_{yy}, d_x, d_y]$ collecting the elements of the affine model and displacement vector. T and a turn out to be functions of the local image gradients. This equation is used as a basis to define what are optimal or good features to track. By construction, good features are defined as those for which the above equation can be reliably solved, which in turn implies the 2×2 matrix T must be non-singular (non-zero eigen values) and well-conditioned (the eigen values do not differ greatly in magnitude). Thus the condition for a feature to qualify as a good feature is that the minimum of the two eigen values must exceed a predefined threshold. The cumulative cost $\sum_{i=0:n} J_i\left(A, d\right)$ is shown to be useful to monitor the quality of track of a feature. If the increase in cumulative cost is not drastic then the track (and thereby the feature) is good. Experiments indicate that a displacement model alone is useful to track features from one frame to the next, but when the quality of track (feature) needs to be measured then an affine model is seen to be more useful (matching feature points between distant frames). The other standard feature point tracking method is the normalized cross-correlation [J.P.Lewis, 1995], closely related to sum of squared differences (SSD) [Nickels and Hutchinson, 2002] based approach. The idea follows from the following sum of squared differences equation.

$$d^2\left(u, v\right) = \sum_{x,y} \left[f\left(x, y\right) - t\left(x - u, y - v\right)\right]^2. \tag{2.2.4}$$

$$= \sum_{x,y} f\left(x, y\right)^2 + \sum_{x,y} t\left(x - u, y - v\right)^2 - 2\sum_{x,y} f\left(x, y\right) t\left(x - u, y - v\right), \tag{2.2.5}$$

where in the above, the feature point template t is moved to a position u, v on the image grid and the summation is done over the support of this template window positioned at u, v. The energy of the template is constant and if the energy of the image $\sum_{x,y} f\left(x, y\right)^2$ is assumed to be constant, then the cross-correlation term $\sum_{x,y} f\left(x, y\right) t\left(x - u, y - v\right)$ can be used as a *measure of similarity* between the feature template and test templates in the image. But in reality, the image energy varies over position, therefore, *normalized* cross correlation is necessary. The normalized cross-correlation coefficient, as a measure of similarity, is defined as shown below.

$$\gamma_{u,v} = \frac{\sum_{x,y}\left[f\left(x, y\right) - \bar{f}\left(u, v\right)\right]\left[t\left(x - u, y - v\right) - \bar{t}\right]}{\sqrt{\sum_{x,y}\left[f\left(x, y\right) - \bar{f}\left(u, v\right)\right]^2\left[t\left(x - u, y - v\right) - \bar{t}\left(u, v\right)\right]^2}}, \tag{2.2.6}$$

where $\bar{f}\left(u, v\right)$ is the mean of the image region under the template positioned at u, v and $\bar{t}$ is the mean of the feature template. It is the general view that using this coefficient is robust to illumination variations due to the zero-meaning operation. However, it is affected by affine or more general motion transformations. Recent work on feature point tracking include batch based point tracking by combining local search guided by a global motion prior of [Buchanan and Fitzgibbon, 2007].

A set of feature N points are selected and tracked over M frames. Each feature point track is a vector of $2M$ position elements. These vectors are arranged columwise to produce a $2M \times N$ *motion matrix*. It is assumed that this matrix has a rank R. Using a RANSAC procedure [Buchanan and Fitzgibbon, 2007], the best basis for this matrix (with rank R) is constructed. This basis is used to define an order M Markov chain motion prior with a Gaussian transition distribution. Given frame $M+1$, a Gaussian predictive distribution is constructed as in a Kalman Filter. This predictive distribution is multiplied with a Gaussian likelihood (defined using template SSD differences) to arrive at the posterior of the feature point. When all the feature point tracks reach frame $M+1$ the motion matrix is recomputed using the frame $M+1$ and the preceeding $M-1$ frames and the whole process repeated again. The highlight of their work lies in exploiting a global motion model constructed using tracks of distributed feature points to constrain the search for the new location of feature points. This decreases errors in the tracks due to clutter. Other recent and related work include tracking a bunch [Rosenberg and Werman, 1997; Badrinarayanan et al., 2007a; Grabner et al., 2007] or flock of feature points [Kolsch and Turk, 2004] to track larger and semantically significant objects. More discussions and analysis follow in the forthcoming chapter 4 dealing with probabilistic interpretations of point tracking.

2. **Region tracking**

   Tracking the position, scale and orientation of a region or a semantic object in a video sequence is by far the most common tracking problem of all. Although the state space is of low dimension, in the order of two to four dimensions, the problem is far from easy and to this day remains an active research problem. Looking back over the years, several approaches have been proposed to tackle this problem. These follow one of the two frameworks described below.

   (a) **Simulation based methods**

   Renowed among them is the CONDENSATION framework of [Isard and Blake, 1996], which casts the tracking problem in a Bayesian filtering framework [Doucet et al., 2001]. Without delving too deep into the details of their method (as forthcoming chapters discuss them in fair amount of detail), the main goal of Bayesian filtering is to evaluate the posterior density of the n dimensional state $x_{0:n}$ (trajectory) given a set of measurements $y_{1:n}$. Invoking the Bayes theorem [Papoulis and Pillai, 2002], this posterior distribution can be expressed as follows.

$$p\big(x_{0:n}|y_{1:n}\big) = p\big(y_{1:n}|x_{0:n}\big)p\big(x_{0:n}\big). \tag{2.2.7}$$

Given a state space model, for instance,

$$x_n = x_{n-1} + \sigma^2 \mathcal{N}\left(0, \mathbf{I}_{2\times 2}\right), \tag{2.2.8}$$

$$y_n = f\left(x_n\right) + \eta(.), \tag{2.2.9}$$

where the *hidden* states $x_{1:n}$ follow a first order Markov Chain model with parameter A, corrupted by Gaussian noise of variance magnitude σ^2. The measurement y_n is a non-linear function of the corresponding hidden state x_n, corrupted by noise process $\eta(.)$ whose statistics are generally unknown. The aim then is to compute the posterior distribution given the measurements. Under simplifying assumptions that measurements are conditionally independent of each other given the state process, one arrives at the following from Eqn. 2.2.7.

$$p\big(x_{0:n}|y_{1:n}\big) = \frac{p\big(y_n|x_n\big)p\big(x_n|x_{n-1}\big)p\big(x_{0:n-1}|y_{1:n-1}\big)}{\int p\big(y_n|x_n\big)p\big(x_n|x_{n-1}\big)p\big(x_{0:n-1}|y_{1:n-1}\big)dx_{0:n}}. \tag{2.2.10}$$

In general it is difficult to analytically evaluate the integral in the denominator as the conditional distribution of the measurements cannot be expressed in a convenient closed form due to the non-linearities governing the measurement process. Therefore, recourse is taken to statistical sampling methods like factored or importance sampling, Gibbs sampling or Markov Chain Monte Carlo sampling [Doucet et al., 2001] for estimating the posterior distribution. [Isard and Blake, 1996] choose importance sampling. To this end, one looks for an importance sampling distribution or proposal $q\big(x_{0:n}|y_{1:n}\big)$ from which sample trajectories $x^i_{0:n}, i = 1 : N$ can be easily drawn and assigned weights in such a way that the weighted sample set can approximate the posterior distribution. Ideally this distribution must be close to the posterior and have a support covering the support of the posterior [Arulampalam et al., 2002]. Say, at instant $n-1$ a weighted sample set $\big\{x^i_{0:n-1}, w^i_{n-1}\big\}, i = 1 : N$ approximating the posterior at that instant is available. If the proposal is factorized as shown below,

$$q\big(x_{0:n}|y_{1:n}\big) = q\big(x_n|x_{n-1}, y_n\big)q\big(x_{0:n-1}|y_{1:n-1}\big), \tag{2.2.11}$$

then the sample trajectories can be augmented with samples of the state at instant n as $x_n^i \sim q\big(x_n|x_{n-1}^i, y_n\big)$. Following which, the importance weights can be recursively updated as shown below [Arulampalam et al., 2002] (also see Chapter 3).

$$w_n^i = w_{n-1}^i \frac{p\big(y_n|x_n^i\big)p\big(x_n^i|x_{n-1}^i\big)}{q\big(x_n^i|x_{n-1}^i, y_n\big)}, i = 1:N. \tag{2.2.12}$$

From the theory of importance sampling, the new sample set approximates the posterior at instant n as:

$$p\big(x_{0:n}|y_{1:n}\big) = \frac{1}{\sum_{i=1:N} w_n^i} \sum_{i=1:N} w_n^i \delta_{x_{0:n}^i}\big(x_{0:n}\big). \tag{2.2.13}$$

A convenient, sub optimal, but ubiquitous choice for the proposal is the prior distibution $p\big(x_{0:n}\big)$ itself.

In filtering literature, the preceeding description is known popularly as the condensation filter or particle filter [Doucet et al., 2000; 2001]. This framework has spawned several key advancements over the years including the mixed-state condensation tracker of [Isard and Blake, 1998] for tracking objects displaying sudden and drastic motion, the auxiliary particle filters of [Pitt and Shephard, 1999], the unscented auxiliary particle filters of [van der Merwe et al., 2000], color based particle filters of [Perez et al., 2002; K.Nummiaro et al., 2003], a multi-modal approach fusing color and stereo sound/motion of [Perez et al., 2004], to the more recent approaches of tracking in low frame rate video by [Li et al., 2007] and the multi-cue tracking proposed in this thesis [Badrinarayanan et al., 2007b].

Although superior to the classical Kalman Filter [Kalman, 1960; Welch and Bishop, 2001], its variants including the Extended Kalman Filter (EKF) based on linearization of the state space models [Julier and Uhlmann, 1997] and the Unscented Kalman Filter (UKF) based on propagating second order statistics of the posterior using sigma-points [Wan and van der Menve, 2000], in view of its ability to propagate non-Gaussian and multi-modal densities, the particle filtering approach is not completely free from ailments. This density propagation scheme relies on importance sampling to result in a weighted sample based approximation of the posterior density. When a finite number of samples are used, in order to restrict the computational complexity, the density propagation is highly dependent on the quality of the importance sampling distribution. In that reagrd, the optimal choice of importance density is difficult to obtain [Arulampalam et al., 2002]. Therefore, adhoc choices dictated by, the computational power in hand, the expected nature, and range of motion of the targets are adhered to. Therefore, any misjudgement in the choice of the importance sampling density can lead to undesirable results. As remarked earlier, the general tendency is to choose any prior on the target motion as the importance sampling density for ease of implemenation. Unfortunately such a choice is far too simplified to accomodate all sorts of realistic scenarios. In summary, the general particle filtering approach is burdened by these practicalities, but nevertheless remains an important one.

(b) **Kernel based methods**

The principle among this branch of methods is to employ smoothly varying kernel functions, possessing properties of convexity and differentiability, for density gradient estimation. For example, if $\{x_1, \dots, x_n\}$ are data points in a d dimensional Euclidean space $\mathcal{R}^d$, the kernel density estimate at point x with a kernel $\mathcal{K}(x)$ of window radius h is given as follows.

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1:n} \mathcal{K}\left(\frac{x - x_i}{h}\right). \tag{2.2.14}$$

Now, if the kernel is so chosen that it is differentiable, then,

$$\hat{\triangledown} f(x) \equiv \triangledown \hat{f}(x) = \frac{1}{nh^d} \sum_{i=1:n} \triangledown \mathcal{K}\left(\frac{x - x_i}{h}\right), \tag{2.2.15}$$

which implies the estimate of the gradient of the density function can be derived from the gradient of the kernel density estimate. As a choice for the kernel, the Epanechnikov kernel [Comaniciu and Meer, 1999] shown below is frequently called upon.

$$\mathcal{K}_E = \begin{cases} \frac{1}{2c_d}(d+2)(1 - x^T x), \text{ if } x^T x < 1 \\ 0, \text{ otherwise,} \end{cases} \tag{2.2.16}$$

where c_d is the volume of the d dimensional unit sphere. With this kernel, the gradient estimate is derived as shown below.

$$\bigtriangledown\hat{f}(x) = \frac{1}{nh^d c_d}\frac{d+2}{h^2}\sum_{x_i \in S_h\{x\}} [x_i - x] \tag{2.2.17}$$

$$= \frac{n_x}{nh^d c_d}\frac{d+2}{h^2}\left(\frac{1}{n_x}\sum_{x_i \in S_h\{x\}} [x_i - x]\right), \tag{2.2.18}$$

where $S_h\{x\}$ is a hyper sphere of radius h and volume $h^d c_d$ containing n_x data points. The term in the brackets is known as the **sample mean shift** $\mathbf{M}_h(x)$. The factor $\frac{n_x}{nh^d c_d}$ is regarded as the kernel density estimate $\hat{f}(x)$ at point x with an uniform kernel. Therefore, the sample mean shift is a function of the normalized density gradient estimate.

$$\mathbf{M}_h(x) = \frac{h^2}{d+2}\frac{\bigtriangledown\hat{f}(x)}{\hat{f}(x)}. \tag{2.2.19}$$

The sample mean shift *vector* above points in the direction of the increasing density and thus can be used effectively for local "mode seeking". This is the basis of all mean shift methods for tracking. An example is color based object tracking using mean shift iterations [Fukunaga and Hostetler, 1975; Comaniciu et al., 2003] for minimizing the distance measure between a prototype kernel density estimate of the object color distribution and several candidate color distributions in the incoming images. Given an image of the object, centered at point x, a prototype color distribution or color model of the object is first constructed as follows.

$$\hat{q}(u) = C\sum_{x_i, i=1:n} \mathcal{K}\left(\frac{x - x_i}{h}\right)\delta\Big(b[x_i] - u\Big), u = 1:M, \tag{2.2.20}$$

where $\hat{q}(u)$ represents the probability of color u, C is a normalization constant and $b[x_i]$ maps the color at spatial point x_i to the quantized color space. The shape of the kernel determines how to weigh the contributions of each spatial point towards the construction of the color model. In a new frame of the video sequence, if tracking is initialised at some point y_0, then a color distribution $p_{y_0}(u)$ is constructed at that point. The similarity between any two non parametric distributions q, p_y can be measured by the following distance measure [Comaniciu et al., 2003].

$$\rho(q, p_y) = \sqrt{1 - \sum_{u=1:M}\sqrt{q(u)p_y(u)}}, \tag{2.2.21}$$

where ρ is the Bhattacharya coefficient (see [Comaniciu et al., 2000]). At some point y around y_0, the above distance measure can be appoximated by a first order Taylor expansion.

$$\rho\big(q, p_y\big) \approx T\big(y_0\big) + C^{'} \sum_{x_i, i=1:n} w_i \mathcal{K}\left(\frac{y - x_i}{h}\right), \tag{2.2.22}$$

where $T\big(y_0\big)$ represents the term without y, $C^{'}$ is a position independent normalization factor and,

$$w_i = \sum_{u=1:M} \delta\Big(b\big[x_i\big] - u\Big)\sqrt{\frac{q\big(u\big)}{p_{y_0}\big(u\big)}}, \tag{2.2.23}$$

is a weighting factor. Now, minimization of the distance measure implies minimizing the second term, which is of the form of a kernel density estimate. Therefore, the *mean shift iterations* can be used to effectively solve this local mode seeking (optimization) problem. The authors [Comaniciu et al., 2000] also propose a solution for automatically varying the *kernel bandwith h* and determining the scale of the target in [Comaniciu et al., 2001].

Apart from the density gradient estimation problem, kernel based methods have also found a place in a number of other instances. The kernel based Bayesian filtering approach [Han et al., 2005] which uses Gaussian mixture approximations of the likelihood and prior to propogate a Gaussian mixture over time. It is claimed that such a technique leads to more efficient sampling of the state space while being able to maintain a multi-modal posterior. A contemporary method to that of [Han et al., 2005] is the multiple collaborative kernels approach proposed by [Fan et al., 2005]. The authors linearize the optimization problem and argue that a single kernel may not be sufficiently "observable" to recover the true motion of the target due to rank deficiency in the observability matrix of the measurement model. Therefore, in an attempt to overcome this rank deficiency the authors suggest using a concatenation of kernels or a combination of kernels to extract more discriminative data. Extending this further, they also suggest using natural structural constraints when possible, for example when tracking a limb two kernels may be placed on either end with a constant length constraint between their positions, to obtain a full rank observability matrix.

More recently [Han et al., 2007] proposed a Bayesian fusion tracking approach using Gaussian mixture based filtering. The idea here is to do particle filtering with a way to distribute a fixed number of particles amongst different observation models. To each particle drawn from an importance sampling function an assignment probability is computed for each of the N different state space models. The particle is then assigned to the state space model under which it bears maximum weight. Thus this look ahead scheme is able to allocate a large number of particles to a potentially fruitful observation model. In their scheme, the kernel based representation of the posteriors plays an important role in enabling the computation of the assignment probabilities at arbitrary particle locations. Other recent kernel based approaches are [Leichter et al., 2007; Nguyen et al., 2007; Yilmaz, 2007].

3. **Multi-Feature or Multi-Part Object tracking**
Next in the hierarchy of increasing state space dimensions is tracking multiple interconnected parts of an object. Such trackers are motivated by the observation that different parts inside a region representing an object frequently have different appearances and if the object is non-rigid, then the parts have different motions too. Therefore, the need for local part trackers automatically arises. These local trackers are linked via a network and integrate "messages" received from other parts, thereby inducing global constraints on their estimates. It may so happen that a part is completely occluded and cannot be tracked reliably by a local part tracker, therefore its position can only be estimated from "predictive information" received from other unoccluded parts.
A *part* can be defined as an image area inside, near or even outside a target. It may or may not have semantic meaning. For example, a part could be an eye on the face or it could just be a patch stradling the forehead and hair. The selection of the parts is usually based on the nature of the object and the expertise of the user. [Perez et al., 2002] use a color based particle filters to track the head and the upper torso with a rigid interconnection between them and demonstrate increased robustness of object tracking over single region tracking. Another instance of multi part tracking is the approach of [Fan et al., 2005], described earlier. [Rasmussen and Hager, 1998] formulate the multi-part tracking problem in a Joint Probabilistic Data Association framework [Bar-Shalom and Fortmann, 1988] (this framework is discussed shortly) and demonstrate robust tracking of

face and hands through clutter and partial occlusions. [Sudderth et al., 2003] propose a non-parametric belief propagation approach with learnt geometric priors for robust tracking of facial features through partial occlusions. Recently, [Yang et al., 2006] proposed tracking of auxiliary objects in the background to achieve robust tracking of an target under temporary occlusions. Multi-part geometric layout based tracking is dealt with in detail in this thesis and [Badrinarayanan et al., 2008] is a related work on fusing patch tracking and online geometric priors. This thesis also investigates a simple interactive verison of multi-part tracking to enable tracking of complex objects in unconstrained videos. More on this topic can be found in the chapter on Multi-part tracking 6.

4. **Articulated body tracking**
The most frequently dealt with problem in articulated body tracking is the tracking of the head, torso, palm, fingers and limbs of a person. The general idea is to divide the object into multiple *semantic* sub-parts and track each sub-part separately. The results of sub-part tracking and the physical constraints on the interconnections or the so called compatibility functions between these sub-parts are all fed to an inference engine, such as the Belief propagation method [Yedidia et al., 2001], to arrive at the marginal posteriors of the sub-parts. This tracking methodology is equally useful for multi-part tracking.
Most instances of articulated body tracking usually assume a strong prior knowledge of compatibility between the variables. For example, the work by [Wu et al., 2003] uses a Gaussian prior on the interconnections and Mean field Variational approximation techniques [Jaakkola, 2000] to approximate their graphical model to ease the computation during inference. Contemporary work by [Sigal et al., 2003] encourage use of motion capture data to learn priors on their graphical model and employ non-parametric belief propagation [Wan and van der Menve, 2000] for inference. Another and a more recent work on similar lines which imposes a temporal constraint of smooth motion on each sub-part to reduce computation is one of [Han and Huang, 2005].
Analytical approaches have also been considered to perform articulated hand tracking; a well known technique is Eigen tracking wherein a view-based representation of the object is created using a set of basis images and a non-linear optimization problem is solved to determine an affine transformation which minimizes a special distance norm (which down weighs outlier matches) between the

matched image and the reconstruction from the basis images [Black and Jepson, 1998]. Replacing the brightness constancy constraint used in optical flow methods by a sub-space constancy constraint, wherein the warped matched image is similar to the reconstructed image from the basis images, this optimization problem assumes the look of a typical constrained optical flow equation. This equation is solved by iterative gradient descent, where at each step the warp parameters and the basis coefficients are refined. The authors also extend this to multi-scale search by pyramidical decomposition of the images. In recent times, such representations have become popular even in low dimensional tracking problems, see [Ho et al., 2004].

5. **Curve or contour tracking**

Tracking the deformable contours of an object has been an active subject of research ever since the seminal work of [Kass et al., 1987]. They begin by defining the contour as a parametric curve $v(s) = \big(x(s), y(s)\big)$ and associating an energy functional to this curve, which captures both the intrinstic characteristics of the curve like bending or its elasticity, and external influences on the curve from image data. The general form of the energy functional is shown below.

$$E\big(v\big) = E_{elastic}\big(v\big) + E_{bending}\big(v\big) + E_{image}\big(v\big). \tag{2.2.24}$$

Example definitions of intrinsic energies depending on the potential energy (stretching is seen to increase the potential energy of the curve) and curvature properties of the curve (relates to the bending of the curve) is shown below.

$$E_{elastic}\big(v\big) = \frac{1}{2}\int_S \alpha(s)\Big|\frac{dv(s)}{ds}\Big|^2 ds, \tag{2.2.25}$$

$$E_{bending}\big(v\big) = \frac{1}{2}\int_S \beta(s)\Big|\frac{d^2v(s)}{ds^2}\Big|^2 ds, \tag{2.2.26}$$

where $\alpha(s), \beta(s)$ are control parameters. The image based energy can be defined in multiple ways depending on the nature of local features extracted from the image parts close to the curve. An example, based on the local image gradients is shown below.

$$E_{image}\big(v\big) = -\int_S \Big|\nabla\mathbf{I}\big(v(s)\big)\Big|^2 ds. \tag{2.2.27}$$

The evolution of an initial contour to its new state, given an image, is then cast as an variational calculus problem of minimizing the total energy E. Employing the Euler-Lagrange equations leads to the following kind of differential equation law for the curve.

$$\alpha(s)\frac{d^2v(s)}{ds^2} - \beta(s)\frac{d^4v(s)}{ds^4} - \nabla E_{image}\big(v(s)\big) = 0. \tag{2.2.28}$$

In practice, equations of the above form are discretized and the resulting difference equations are solved to determine the location of the curve. Although restricted to small motions and closed curves, this technique paved the way for several other analytical techniques for deformable contour tracking. Among them is the Active shape models or smart snakes approach of [T.F.Cootes and C.J.Taylor, 1992], also an iterative optimization scheme which tries to fit best a shape model to image evidence by varying its pose and shape parameters under a global constraint on the shape parameters. [Xu and Prince, 1998] proposed to alleviate the problem of mifitting concave curves in the Kass approach by using the edge image based external energy term in the original snakes equation.
Another important strand of analytical methods for contour tracking are the ones based on the level set representation of contours. Based on the work of [Osher and Sethian, 1988] the idea is to represent the contour $v\big(x(s,t)\big), x(s,t) \in \mathcal{R}^N$ ($1 \leq s \leq S$, is the spatial position parameter and t is time) as a *zero level set* of a higher dimensional *embedding* function $\psi\big(\mathbf{x}(s,t)\big)$. That is to say, at any instant t, the evolving contour can be obtained by solving the following equation.

$$\psi\big(\mathbf{x}(s,t)\big) = 0, \tag{2.2.29}$$

with the property that at the instant $t = 0$,

$$v\big(\mathbf{x}(s,t=0)\big) = \big\{\mathbf{x}(s,t=0) | \psi\big(\mathbf{x}(s,t=0)\big) = 0\big\}. \tag{2.2.30}$$

It is now necessary to produce an equation for the surface $\psi\big(\mathbf{x}(s,t)\big)$ such that its zero level set would deliver the propagating contour $v\big(x(s,t)\big)$. For points $\mathbf{x}(s,t)$ on this propagating contour the following relation must hold by definition.

$$\psi\big(\mathbf{x}(s,t)\big) = 0. \tag{2.2.31}$$

By the chain rule of calculus,

$$\frac{\partial\psi}{\partial t} + \sum_{i=1}^{N} \frac{\partial\psi}{\partial x_i}\frac{\partial x_i}{\partial t} = 0, \tag{2.2.32}$$

where x_i is component in dimension i of $\mathbf{x}(s,t)$. This partial differential equation (pde), of the Hamilton-Jacobi type, is to be solved for the evolving surface ψ under the initial condition given by Eqn.2.2.30. This equation possesses several desirable properties including description of arbitrary motions of the contour topology and ease of discretization. Contour tracking methods based on such representations include [Malladi et al., 1995; Yilmaz et al., 2004].

Stochastic methods for contour/curve tracking based on measurements under clutter use the concept of iterative sampling to maintain multiple hypotheses of contours or curves [Storvik, 1994]. The CONDENSATION framework of [Isard and Blake, 1996] demonstrated the application of iterative sampling to track contours of heads and hands under dense clutter. Their filtering framework relies on a discretized stochastic diffusion equation which models the learnt parametric motion of a B-spline curve [Blake et al., 1995] and a measurement density which can be evaluated at each curve or contour sample. With these two components, a weighted sample based approximation of the posterior distribution for the evolution of the curve or contour is computed at each tracking step (see Chapter 3). The generality of their framework took it far beyond tracking curves or contours and Gaussian posterior distributions [A. Blake and Zisserman, 1993] to influence a generation of sequential Monte Carlo tracking techniques [Doucet et al., 2001].

2.2.2 Statistical paradigm for visual pattern modeling and interpretation

Statistical models of visual tracking follow one or a combination of three paradigms: the descriptive modeling paradigm, discriminative modeling paradigm and the descriptive-generative modeling paradigm. These paradigms represent different schools of thought representing the statistical idea to analyse the observed data, the hidden variable(s) in question and their relationships. Tracking algorithms derived from statistical models can therefore be seen to represent one or a combination of these paradigms.The interpretation of the visual tracking problem under each of these paradigms is discussed below.

1. **Descriptive models**

   The descriptive modeling of data (say, images captured by a sensor) is marked by the absence of hidden variables. It is fundamentally based on extracting several image feature statistics (like statistics of linearly transformed images using

Fourier, Gabor filters) and assuming these as the marginal statistics of an ensemble of images. A maximum entropy probabilistic model, usually from the exponential family, is derived whose marginal statistics match that of the image feature statistics. The higher the number of image features that are extracted, the more complex the descriptive model. An image then is a sample drawn according to this probability distribution. Following this and extending to sequential data (typically a tracking scenario), a maximum entropy model is estimated by statistical analysis.

$$p\big(y_{1:n};\Theta\big) = \frac{1}{Z(\Theta)}\exp\left\{-\sum_{k=1}^{K}\big\langle \lambda_k, h\big[\phi_k\left(y_{1:n}\right)\big]\big\rangle\right\}, \tag{2.2.33}$$

where $\Theta = \big\{\lambda_1, \ldots \lambda_K\big\}$ are the Lagrange multipliers in the optimization and $h\big[\phi_k\left(y_{1:n}\right)\big]$ are histograms of the features $\phi_k\left(y_{1:n}\right), k = 1 : K$. The histograms are assumed to be the marginal statistics of the true distribution of the data.
If it were possible to estimate such a model, of which the given data sequence is a sample, then in effect there is no tracking problem at all!, for any sequence of patterns can be "generated", instead of being "extracted" from some data sequence. Unfortunately such a situation is far from being reached. Even the laws governing the synthesis of simple textures remain an active subject of research [Zhu, 2003] today.

2. **Discriminative models**
The discriminative modeling paradigm is an "image processing" or bottom-up approach to modeling the relationship between the observed data and hidden variable of interest (in tracking, say, the position of an object). The relationship between grouping of image based features like key points,edge segments, curves and attributes of 2D objects (for instance, the position, shape and appearance of an object) is described in a statistical sense. The grouping itself is done using "visual dictionaries", which is basically a collection of all subjective and empirical knowledge of how to generate complex image structures from simpler building blocks. The parameters of a probabilistic model (which employs the visual dictionary within it) which relates the data (image features) and the hidden variable of interest (say, position of the object) is learnt by some form of training. An example of such a discriminative model is a conditional random field (CRF) model

[Qi et al., 2005], used for classification problems, shown below.

$$p(x_{0:n}|y_{1:n}, \mathbf{w}) = \frac{1}{Z(\mathbf{w})} \prod_{i\epsilon\mathcal{C}_i} \phi_i(\mathbf{x}_i, y_{1:n}; \mathbf{w}), \tag{2.2.34}$$

where $\mathcal{C}_i \subset x_{1:n}$ are cliques of hidden variables and the hidden variable $\mathbf{x}_i \epsilon \mathcal{C}_i$. The compatibility functions $\phi_i, i\epsilon\mathcal{C}_i$ encode the relationship between the data $y_{1:n}$ and the hidden variables $x_{0:n}$. $\mathbf{w}$ parameterizes these functions. $Z(\mathbf{w})$ is the normalization constant (usually difficult to compute). An example of a compatibility function is $\exp\left[\mathbf{w}\zeta_i(\mathbf{x}_i, y_{1:n})\right]$, where the function $\zeta_i(.)$ extracts features of the relationship between the data and the hidden variables in the clique $\mathcal{C}_i$.
The phrase "training the CRF model" implies learning the parameters $\mathbf{w}$. Once the model is trained, given a new data sequence $y^*_{1:n}$, estimates of the hidden variables (say, tracks) is made from the predictive distribution $p(x_{0:n}|y^*_{1:n}, \mathbf{d}, \mathbf{w})$, where $\mathbf{d} = \{x_{0:n}, y_{1:n}\}$ represents the complete training data (See [Qi et al., 2005] for more details). The CRF model is, in principle, capable of capturing relationships between the observed data at several instances (long range correlations) and hidden variables, thus avoiding standard Markovian assumptions which decrease modeling power.
Some deterministic methods regard the tracking problem as one of classification or equivalently prediction of a pattern in each frame of the sequence from extracted image features. These are also considered (for discussion's sake) follow this paradigm. Well known among such approaches is the Support Vector Tracking (SVT) of Avidan [Avidan, 2004] where an Support Vector Machine (SVM) classifier is trained to detect vehicles. In each new frame an energy functional is maximized to find the image region with the maximum support vector score. This optimization then delivers the image region closest to the training set. In order to introduce smoothness into tracking, the energy functional is made to include the brightness constancy constraint of standard optic flow formulations. The iterative optimization of this constrained problem results in the estimation of the motion parameters and maximizes the classication score based norm. The Eigen tracking approach of Black and Jepson [Black and Jepson, 1998] can also be classed here. It is similar to SVT except that the norm in the energy functional is a distance to the space of Eigen vectors instead of a SVM score. The Relevenace Vector Machine (RVM) based tracking of [Williams et al., 2005] is another semi-deterministic approach wherein a trained classifier outputs a displacement for a

given image region instead of a binary classification score. This displacement is then fused with a motion predictor using a standard Kalman Filter with a gain factor balancing the uncertainty of the so called displacement expert (the RVM output) and the stochastic diffusion equation. Other recent approaches using key point matches for tracking is the discriminative local features approach of [Grabner et al., 2007], where online boosting techniques are used to update a key point based appearance description of the object. Analytical approaches relying on the optic flow equation like the KLT tracker [Tomasi and Kanade, 1991] or kernel based approaches like Mean-shift tracking [Comaniciu et al., 2000] also belong to this group.

Although computationally efficient and intuitive, the argument against the foregoing methods (as compared to CRF models) is that they are more so a computational heuristic, than a model formally incorporating the uncertainties in the data and parameters. Further these methods usually require a top-down guidance (priors) to bring down the computational cost. However, these techniques can act to develop importance sampling proposals for statistical methods aiming to approximate the posterior $p\big(x_{1:n}|y_{1:n}\big)$.

3. **Descriptive-Generative models**

   In contrast the descriptive-generative model is a "top-down" approach where the top most layer in the conceptual hierarchy, usually attributed to a complex visual pattern or image, is taken to be synthesized based on a series of discoveries (inferences) of hidden variables. The fundamental idea is that the complex visual patterns or structures are generated by a hierarchy of hidden variables which need to be inferred based on image data. The hidden variables themselves are related by a descriptive model, like a Markov Random Field (MRF) model learnt from a statistical analysis of an ensemble of images or more frequently, simply postulated by the designer for convenience. The transition between layers in the model (intra-layer compatibilities) is parameterized by a component of a visual dictionary which is required to interpret higher structures from lower ones.

   From a tracking point of view, the distribution $p\left(y_{1:n};\alpha,\beta\right)$ is composed by the following integration.

$$p\left(y_{1:n};\alpha,\beta\right)=\int p\left(y_{1:n}|x_{1:n};\alpha\right)p\left(x_{1:n};\beta\right)dx_{1:n}, \qquad (2.2.35)$$

where α is a parameter in the link between the observed data and the hidden variables, while β is a parameter of the descriptive model relating the hidden variables. In Bayesian terms, the first component inside the integral is the data likelihood function and the second is the prior distribution.

The integral on the right hand side is either analytically evaluated (in feasible cases) or approximated using techniques like deterministic quadrature or Monte Carlo simulations. Tracking methods based on descriptive-generative models all vary in the way the data likelihood distribution and the prior are composed. Among the key methods is the CONDENSATION filter of Isard and Blake [Isard and Blake, 1996] using edge features to define the likelihood and a first order Markov stochastic diffusion model as the prior. The parameters of this prior distribution is learnt from ensemble of (labelled) image data (See [Blake et al., 1998; 1995]). The color based particle filter of [Perez et al., 2002] uses color histogram features to define the likelihood and a second order stochastic diffusion model as the prior. Likewise is the tracking approach based on fusion of multiple features like sound and color, as in [Perez et al., 2004]. Subspace based methods like the one of [Ho et al., 2004] use projections of an ensemble of the tracked pattern as features.

Although computationally friendly, under some standard assumptions, these techniques work only when consistent likelihood functions and priors can be defined. In several important and practical cases such definitions are hindered by dimensionality problems and mathematical intractabilities (see Chapter 3 for further discussions).

The reader is also referred to an indepth discussion, epistemological view point, analysis of these paradigms and their relations in [Zhu, 2003].

2.2.3 Modality of visual appearance description

The task of visual tracking may be equated to the task of recognizing some particular features or attributes of a particular visual pattern or object. Fundamental to this task is a description of the visual pattern by some statistical or deterministic feature analysis. Depending on the paradigm used for tracking, features can be classified into the following two categories.

1. **Descriptive features**
   Under the descriptive modeling paradigm, given an ensemble of images of a visual pattern and a pool of features, a descriptive model $p(y_{1:n};\Theta)$ can be derived (see earlier discussions) by using some subset of features from this pool. A new *most descriptive feature* from the pool can then be added to this subset using the minmax entropy principle, that is to say, choosing the feature which minimizes the KL divergence between $p^{+}(y_{1:n};\Theta)$ and $f(y_{1:n})$. This method then selects the most descriptive features among the pool by being as dissimilar as possible to the already existing features. An example is made by [Zhu, 2003] of the descriptive features such as the local image gradient, second order local image gradient and local curvature features used in descriptive modeling of contour priors in [Kass et al., 1987]. Another example is the local edge normal features used by [Isard and Blake, 1996].

2. **Discriminative features**
   In discriminative or descriptive-generative modeling methods, features are chosen to minimize the KL divergence between $p(x_{1:n}|y_{1:n})$ and $q(x_{1:n}|\phi_k(y_{1:n}))$. In simple terms, the chosen features try to minimize the loss in inferring the hidden variables (image structures) from image features rather than the image itself. These features are called discriminative features and are implicitly used even in descriptive-generative modeling methods which rely on discriminative modeling of importance proposals (distributions such as $q(x_{1:n}|\phi_k(y_{1:n}))$).
   More often than not selection of features are discriminative in nature and their selection is based on empirical experience rather than a rigorous statistical analysis as described above. Key point features and their online update is built into the tracking scheme of [Grabner et al., 2007]. Color histogram features used in [Comaniciu et al., 2000; Perez et al., 2002] and feature point templates used in KLT tracking [Tomasi and Kanade, 1991] are other examples of discriminative features.
   The importance of the nature of a feature(s) cannot be over emphasized. It is a fact which remains fundamental to the success of a tracking method. It is a general percept that using multiple complementary cues should lead to more robust tracking, but it is not always clear as what the nature of these cues must be. An even more challenging issue is to how to select, use and adapt these features over time. The selection of a feature must bear in mind the ensuing complexity in its

adaptation. Attempts have also been made to construct complex time varying discriminative features using combinations of time varying image templates [Jepson et al., 2003], online subspace modeling [Ho et al., 2004; Yang and Wu, 2005] and more recently, features of varying time spans [Li et al., 2007]. More details of this complex issue follow in Chapter 5. A recent work related to this discussion can be found in [Badrinarayanan et al., 2007b].

2.2.4 Single or Multiple target tracking

In single target tracking, the standing assumption is that a sensor (say, a video camera) obtains measurements (images) from which the attributes of the target (position, size and so on) can be inferred. However, it is not known as to whether these measurements must be associated to the true target or spurious clutter arising due to absence of the target in the range of the sensor/presence of occluding objects with similar appearance/sensor noise/errors. This problem is termed the **data association problem** or simply the problem of dealing with *unlabelled* measurements. It is interesting to note that single target tracking methods do not explicitly deal with this problem (see the pseudo data association based approach in Chapter 5 for an explicit formulation).
The data association problem is further vexed in the multiple target tracking scenario where there are multiple measurements arising from several targets. These measurements are assumed to be obtained from one or more (homogeneous) sensors. As in single target tracking, the measurement at each sensor can be associated to a target or spurious clutter, as earlier. In addition, when multiple sensors are brought into play, there is the issue of associating measurement to targets (this is a combinatorial problem). Therefore, multi target tracking methods must make way for these additional possibilities too.
Among the well known methods for multiple target tracking is the Expectation Maximization (EM) based approach of Probabilistic Multi Hypothesis Testing (PMHT) [Willett et al., 2002]. This so called optimal approach is a batch based iterative scheme to arrive at a Maximum a Posteriori (MAP) estimate of the joint state of K targets, represented as $X_{1:T} = \{x_{1,1:T}, \ldots, x_{K,1:T}\}$, given a sequence of observed data or measurements $Y_{1:T}$ from a sensor, taking into account the unknown target to measurement associations. Formally,

$$\hat{X}_{1:T} = \underset{X_{1:T}}{\operatorname{argmax}}\, p\big(X_{1:T}|Y_{1:T}\big), \tag{2.2.36}$$

where $\hat{X}_{1:t}$ is the sought MAP estimate of the joint trajectory of the K targets with the set of data association vectors $\Omega = \Big\{\Lambda_t = \{r_{j,t}\}, j = 1 : M\Big\}, t = 1 : T$ (with element $r_{j,t}$ representing the mapping of the measurement to target(s) at instant t as the hidden parameter).

This approach to multi-target tracking [Willett et al., 2002] is limited only to offline tracking due to its non-sequential nature. Further, the general assumption that a measurement can be associated to one or more targets and vice-versa is considered unrealistic.

The PMHT and the Probabilistic Data Association Filter (PDAF) [Bar-Shalom and Fortmann, 1988; Rasmussen and Hager, 2001] (seen as an extension to the Kalman Filter incorporating uncertainty due to unlabelled measurements) were initially designed for a multi target single sensor tracking scenarios. Naturally, then arose the multi target multiple sensor problems. Sequential Bayesian estimation methods have been regarded highly in this area for their generality. To restate and recaptiulate the problem in a Bayesian context, the issue is to infer the joint state of K targets, represented as $x_t = \{x_{1,t}, \dots, x_{K,t}\}$, given a sequence of observed data or measurements $Y_{1:t}$ from multiple sensors or observers in the scene. The observation at instant t is composed of a set of M_i measurements at observer i among the N observers (sensors);

$$Y_t = \Big\{\{y^1_{1,t}, \dots, y^1_{M_1,t}\}, \dots, \{y^N_{1,t}, \dots, y^N_{M_N,t}\}\Big\}. \tag{2.2.37}$$

The measurements made by each observer in general contain measurements arising from targets and/or clutter; whichever the case, the measurements are unlabelled. Here arises the data (measurement) association problem. For an observer i, at instant t, a measurement to target association vector $\Lambda^i_t = \{r^i_{j,t}\}, j = 1 : M_i$ maps the measurement j to a target or to clutter;

$$r^i_{j,t} = \begin{cases} k \in 1, \dots, K, \text{if the measurement} j \text{arises from target} \, k \\ 0, \text{if the measurement arises from clutter.} \end{cases} \tag{2.2.38}$$

This data association is equivalently represented with no loss of information as an target to measurement association vector $\tilde{\Lambda}^i_t = \{\tilde{r}^i_{k,t}\}, k = 1 : K$, the elements of which are,

$$\tilde{r}^i_{k,t} = \begin{cases} j \in 1, \dots, K, \text{if the measurement} j \text{arises from target} \, k \\ 0, \text{if the measurement goes undetected.} \end{cases} \tag{2.2.39}$$

The probabilistic filter must infer the posterior of the joint state bearing the uncertainty in the data association vectors. Even without the uncertainty in data association, filtering in the joint state space is a formidable problem; the dimension of the state space increases with the number of targets. To circumvent this problem, taking into account the data association uncertainty, the well known Joint Probabilistic Data Association Filter (JPDAF) [Bar-Shalom and Fortmann, 1988] disintegrates the problem into tracking individual targets and handles the data association problem by making a soft-assignment of measurements to targets. For a target k, the posterior is evaluated as shown below.

$$p_k\big(x_{k,t}|Y_{1:t}\big) \propto \prod_{i=1}^{N}\Big[\beta^i_{k0} + \sum_{j=1}^{M^i}\beta^i_{kj}p\big(y^i_{j,t}|x_{k,t}\big)\Big]\int p_k\big(x_{k,t}|x_{k,t-1}\big)p_k\big(x_{k,t-1}|Y_{1:t-1}\big)dx_{k,t-1}, \tag{2.2.40}$$

where in the term outside the integral (the likelihood function) β^i_{k0} is the posterior probability that target k goes undetected at observer i and $\beta^i_{kj} = p\big(\tilde{r}^i_k = j|Y_{1:t}\big)$ is the posterior probability of associating target k to measurement j at observer i. Therefore, the JPDAF *inserts* the uncertainty in the data association through the data likelihood. The posterior probabilities in the data likelihood are computed via marginalization as shown below.

$$\beta^i_{kj} = p\big(\tilde{r}^i_{k,t} = j|Y_{1:t}\big) = \sum_{\tilde{\Lambda}^i_t \in \Omega^i_t : \tilde{r}^i_{k,t}=j} p\big(\tilde{\Lambda}^i_t|Y_{1:t}\big), \tag{2.2.41}$$

where Ω^i_t is the set of all possible data associations at instant t at observer i. The posterior of the data association vector is decomposed as follows.

$$p\big(\tilde{\Lambda}^i_t|Y_{1:t}\big) = \frac{p\big(\tilde{\Lambda}^i_t\big)p\big(y_{i,t}|\tilde{\Lambda}^i_t, Y_{t/i}, Y_{1:t-1}\big)p\big(Y_{t/i}|\tilde{\Lambda}^i_t, Y_{1:t-1}\big)}{p\big(Y_{1:t}\big)}. \tag{2.2.42}$$

$$= \frac{p\big(\tilde{\Lambda}^i_t\big)p\big(y_{i,t}|\tilde{\Lambda}^i_t, Y_{1:t-1}\big)p\big(Y_{t/i}|Y_{1:t-1}\big)}{p\big(Y_{1:t}\big)}. \tag{2.2.43}$$

$$\propto p\big(\tilde{\Lambda}^i_t\big)\prod_{r^i_{j,t}=0} p\big(y^i_{j,t}\big)\prod_{r^i_{j,t}=k} p_{r^i_{j,t}}\big(y^i_{j,t}|Y_{1:t-1}\big). \tag{2.2.44}$$

The decomposition is performed by aid of assumptions that at any instant the measurements at different observers are independent of each other, the predictive likelihoods of a measurement at any observer is independent of predictive likelihoods of other measurements and measurements at any observer are independent of data associations at other observers. The fact that $\tilde{\Lambda}^i_t$ and Λ^i_t contain the same information is also used here. The computation above relies on a pre-defined prior on the data association vector and clutter probability. [Vermaak et al., 2005] provide some arguments for defining such priors. The remaining predictive likelihood component can be evaluated using the following equation.

$$p_{r^i_{j,t}}(y^i_{j,t}|Y_{1:t-1}) = \int p_{r^i_{j,t}=k}(y^i_{j,t}|x_{k,t})p(x_{k,t}|Y_{1:t-1}). \qquad (2.2.45)$$

With things standing as such, the computational complexity involved in the determining the posterior association probabilities β's is a deterrent to direct use of this technique. For an observer, to evaluate each of these posteriors, all possible permutations of data association must be considered. This number, for even a moderate number of targets, is a mighty one and when there are several sensors, things only detoriate. To make this computation manageable the JPDAF filter uses the concept of "Gating" [Bar-Shalom and Fortmann, 1988] - that is at each observer, reject all *infeasible* data associations by defining a *measurement validation region* for each target via computation of its predictive likelihood as discussed above. For a particular target only measurements falling in this region are considered feasible. This reduces the entire complexity by an order of magnitude. A more thorough discussion of the Gating concept can be accessed in [Vermaak et al., 2005]. The main drawback of the JPDAF [Bar-Shalom and Fortmann, 1988] is that at each instant it approximates the posterior filtering estimate of each target by a Gaussian distribution for convenience. This could prove harmful in the presence of clutter when data likelihoods are multi-model in nature. A Gaussian posterior would mean discarding several co-existent alternate hypotheses. Attempts to do away with such conveniences, by maintaining Gaussian mixture forms for the posteriors, have also been proposed [Pao, 1994]. A full SMC approach bearing the name MC-JPDAF has been proposed in [Vermaak et al., 2005] to propagate non-Gaussian and possibly multi-modal posterior distributions for the targets.

Continuing the discussions on the MC-JPDAF, [Vermaak et al., 2005] consider the general problem of estimating both the state of the targets and their association vectors together in an Sequential MonteCarlo framework. In formal terms, they propose to estimate the following distribution.

$$p(X_t, \hat{\Lambda}_t|Y_{1:t}) \propto p(\tilde{\Lambda}_t)p(Y_t|X_t, \tilde{\Lambda}_t) \int p(X_t|X_{t-1})p(X_{t-1}|Y_{1:t-1})dX_{t-1}, \qquad (2.2.46)$$

where $X_t = \{x_{1,t}, \ldots, x_{K,t}\}$ is the joint state of the targets and $\hat{\Lambda}_t = \{\lambda_{1,t}, \ldots, \lambda_{N,t}\}$ is the joint data association vector. A standard assumption of independence of measurements over time, conditional on the joint state and data association vectors is made (See [Doucet et al., 2001]). Within the SMC framework, [Vermaak et al., 2005] propose different forms for constructing proposal distributions for importance sampling.

Depending on the assumptions used for defining the dependecies between the elements of an association vector to arrive at the proposal for association vector hypotheses, they arrive at the Serial Sampling Particle Filter [Vermaak et al., 2005] for multi target tracking. The proposal distribution for an observers association vector is constructed to subsume the idea of gating by serially conditioning the elements of an association vector on its preceeding elements, thus pruning away infeasible hypotheses. In yet another filter, termed the Independent partition particle filter, they assume that at any observer several targets may be assigned to a single measurement and vice-versa. This eases the construction of the proposal distributions and effectively brings the problem of sampling the joint state and association vector space to simply sampling from individual target posteriors, although paving the way for a degraded performance under a large number of targets in high clutter. In conclusion, the authors remark the consistency of the MC-JPDAF over the other methods.
Of the other interesting approaches is the Markov Chain Markov Chain (MCMC) based particle filter for tracking multiple interacting targets of nearly identical nature [Khan et al., 2004]. In particular the authors tackle the problem of tracking a bunch of ants in a closed environment. They insist that traditional factored models for motion prior are inefficient when there are frequent interactions of identical targets, and demonstrate that a motion prior which includes a term describing the interactions of a target with other targets in its vicinity leads to robust tracking. The traditional factored motion models are replaced by an MRF based motion prior whose links (edges) are modified online depending on the possible interactions of each target. They also replace importance sampling by an MCMC sampler [Doucet et al., 2001], based on the Metrepolis-Hastings algorithm [Hastings, 1970], to draw true samples from the joint target state posterior.
BraMBLe is another Bayesian approach based on sequential MC for multiple object tracking for indoor scenes [Isard and MacCormick, 2001]. This method relies on blob representations of people and

2.2.5 Mode of tracking

Two modes of tracking stand out. First of which is the *automatic multi-cue mode* and the second is *interactive mode* or tracking with an *operator in the loop*. In principle, most tracking algorithms, especially multi-cue robust schemes, claim to be automatic in nature, delivering a "track" (or tracks in multiple target case) once they have been set to track a particular target(s) by another device (manually initialised) or algorithm (say, a face detector). But it is only automatic until the first failure point i.e when the method loses track, an occurence which is common in all tracking methods today which work in unconstrained environments. At each failure, the method has to be reinitialised by a supporting algorithm like an object detector or by manual intervention. Thus, one can argue that, there still is no true automatic tracking method which can do without any external aid. In practice though, one could loosely define automatic tracking algorithms as those "online" or "sequential" (meaning not batch processed) methods which require "minimal" external support for tracking in an unconstrained environment when the target displays complex motion, appearance changes and occlusions, for a duration in the order of a few hundred seconds. This definition itself implores further study of the general visual tracking problem.

Although almost all of today's tracking methods can be technically classified into interactive tracking methods and there is no fine line dividing automatic and interactive modes, it is generally understood that algorithms requiring considerable manually interaction or which are "offline batch based" can be classified as interactive schemes. Of immense value in several applications including high end post-production of cinema, motion capture, to name a few, interactive methods have recently gained in popularity and is currently indispensable when the tracked state is of a high dimension (contours and layouts). A brief summary of some algorithms in these two classes is made below.

1. **Automatic multi-cue mode**

   Since automatic tracking is synonymous with requirement of robust tracking, it relies on methods based on multiple cue fusion which stand out in this category. From collective empirical experience, it is common knowledge that, by somehow offsetting the weaknesses of any single cue by complementary cue(s) and introducing cooperative interaction between cues, leads to robust tracking.

   Among the recent examples in this category are hand tracking based on a bunch of KLT feature points [J.Shi and C.Tomasi, 1994] and object color distribution [Kolsch and Turk, 2004]. A set of KLT features or "good features to track" are chosen inside a detector provided area and several of them pruned away based on the learnt color distribution of the hand and a probability likelihood spatial belonging mask provided at the detection stage. The remaining features are tracked using the KLT point tracking approach and a subset of them removed, if considered to deviate from the "flock" distance-wise or matching correlation-wise. The removed features are replaced by new features (this time non KLT features) at

locations "close" to the flock and with a neighbourhood color content within a prescribed distance to the hand color distribution. This approach is seen as a two-cue fusion of *textured* feature points and *textureless* color distribution, where the strength of the color modality can be increased (or decreased) based on increased replacement (longer retention) of feature points at each instant.
The Co-inference learning approach is another example, where, in light of a variational approximation of a factorized HMM graphical model (both hidden states and observations are factored) for tracking, the authors seek introduce intercoupling of cues. The estimation of the variational parameters (which play the role of the data likelihoods in the original factored model) in the approximate model effectively couple the cues together. This fact is used as a basis for developing a SMC filtering based method which tracks a target based on shape (ellipse) cues and color cues. The essence of co-inference learning is captured via importance sampling: samples of shape (conics) are derived based on color cues and vice-versa. It is then hoped that such co-operation will maximise the data likelihood of both the cues.
[Triesch and von der Malsburg, 2000] propose *democratic integration* of multiple cues for robust tracking. They reemphasize that the cues must *self-organize* in absence of an external agent monitoring their reliabilities. In lieu of a democratic setup, information from cues (shape, color, intensity differences, motion continuity and contrast change) which are coherent or concordant are integrated automatically while suppressing the discordant ones. Each cue is associated with a reliability weight which follows a first order differential equation law parameterized by a variable time constant. Based on this law, the reliability of cue varies as the difference between its current *quality* or how well it participated in arriving at the current estimate of the target state and its current value. Computing the reliability based on this law essentially produces a prediction of the reliability for the next time instant. The time constant determines how quickly the reliability must be varied. Once the reliabilities are in hand the state of the target is a MAP estimate of a linear combination of saliency maps (created by comparision of the cue appearance model or prototype to each image location) of the cues. A similar differential law is also used for adapting the appearance models or prototypes for each cue. Apart from the preceeding methods there exist several other techniques, some of which have already been discussed earlier in

this review [Perez et al., 2004; Han et al., 2007] and some others [Veeraraghavan et al., 2006; Moreno-Noguer et al., 2008].

2. **Interactive mode**
Among the very recent approaches is the formulation of [Buchanan and Fitzgibbon, 2006] for interactive point tracking. Theirs is a "search and optimization" strategy to tracking. To start with, every pixel (actually a patch centered on it of the size of feature point template) in each frame of the video sequence is projected onto a basis and so "coded" automatically. For each frame, the codes of its pixels are arranged as a k-d tree for purposes of efficient search. This concludes the preprocessing stage. Next, in the interactive stage, the user views the sequence and designates key frames and the feature point location, template in each of these key frames. A search operation is then launched on the k-d trees to look for the top M locations in each tree which best matches these key frame templates. Finally, a table is filled in, with these matches on the ordinate and their "match errors" on the abscissa. A provision is made for occlusions by including a occlusion state with a fixed occlusion penalty as one of the candidates in the ordinate. Dynamic programming (DP) is carried out to compute the least cost path through the table and thus directly marking the trajectory of the feature point. In extension to the interactive feature point tracking of [Buchanan and Fitzgibbon, 2006], [Wei et al., 2007] propose interactive tracking of color objects. It is batch based with no preprocessing stage. The interactive contribution of the user is in providing key frames in which a bounding box over the target is marked. A non-parametric color distribution of the target is then constructed (histograms) and a target detector based on color similarity is used to select "close" candidates in each frame. The best M of these candidates in each frame is "picked out" by a forward-backward mean-shift tracking based trajectory growing within a small time window. From this point onwards the method aligns itself to the DP optimization based optimal trajectory selection. The above two methods are batch based methods from recent times. In contrast non-batch based or naive "step back, correct and proceed" techniques has been in existence for a long time. Their simplicity must not be misconstrued as their weakness and for a multitude of tasks which, for example, cannot afford enough computation for massive preprocessing as required in [Buchanan and Fitzgibbon, 2007; Wei et al., 2007] or which need tracking of complex contours or layouts

or which need quick-fix solutions, this is a very useful resort. Reiterating what was said earlier, most automatic methods can all be made interactive, but their usefulness in an interactive mode is directly determined by their inherent characteristics and amount of user support that is required. More on this issue can be found in Chapter 6.

2.2.6 Methods vs Systems

The discussion on automatic tracking methods highlighted their need for an external agent or method, like an object detector, to "correct" them. Differently, tracking algorithms can play a symbiotic role too, that is, they could support an external agent or method by reducing their characteristic burden. In such cases, where tracking is used as a functional component, it is defined as a tracking based system. It is emphasized here that this viewpoint as to what constitutes a tracking based system is a personal one of the author and may not necessarily find universal approval.

Tracking systems are very popular in applications such as people tracking for visual surveillance and activity monitoring [Ramanan and Forsyth, 2003; Siebel and Maybank, 2004; Wren et al., 1997] and building Human Computer Interfaces (HCI) [Wren et al., 1997]. Samples of such systems in literature include, people finding and tracking system of [Ramanan and Forsyth, 2003]. The idea is to describe the image of a person by a set of rectangular boxes placed on the torso, arms and legs and then generate a continuous track of these parts. To achieve this, the authors search for rectangle like segments, compute their color histogram models and cluster them in color feature space to produce likely segments of body parts. Clusters which disobey a motion constraint (people are assumed to move constantly) are pruned away. The remaining clusters then produce tracks of different body segments. The authors contrive a probabilistic graphical model, inference on which would amount to generating the required tracks. To infer the track of a person, a loopy undirected graphical model linking the appearances of various segments and their positions is constructed. Inference on this loopy graph is reduced to a series of inferences on different trees connecting some nodes of the loopy graph. For instance, first only the appearance of the torso and the hidden positions of the torso in the sequence are inferred. In parallel, the appearance of the arms and its positions are inferred. Thus the appearance models (in color histogram feature space) are leant for different body segments. It is argued that such inferences are equivalent to clustering in color histogram feature spaces to produce likely clusters of body segments.

Then on, the next tree connects the positions of the arms and torso with a kinematic prior. The positions of the arms and torso are inferred by message passing [Yedidia et al., 2001] on this tree and in doing so, imposing a kinematic constraint.

An example of how tracking is used in a system is in the person detection and localization system of [Song et al., 2000]. Given a frame in a video sequence, key point features are automatically detected. These key points are tracked to the next frame and the problem is to detect a person based on computing a ratio of likelihoods for labeling these points into labels belonging to foreground and background. If the likelihood of labelling the points as foreground body parts (arms, wrist, knee etc.) is greater than labeling them as background features, then a person is considered detected in the scene. Once detected, a person is localized by deriving the Maximum Likelihood (ML) estimate for the labels. The likelihood model is an MRF with Gaussians as the compatibility potentials whose parameters are learnt by extensive hand labeling of body features and their correspondences between frames. Provisions are also made for missing or occluded body parts.

In the context of visual surveillance, an example of a tracking based system is the Annotated Digital Video For Intelligent Surveillance and Optimized Retrieval or in short, ADVISOR system [Siebel and Maybank, 2004] for people tracking, crowd monitoring and behaviour analysis in railway stations. Tracking is done using data from multiple calibrated cameras for determining the position and size of moving objects. The results of tracking is also fed to a behavior analysis module for signaling rogue activity.

Tracking is also essential in several augmented reality applications, for instance, creating virtual spaces of people and other graphically generated objects undergoing interaction. More recently the concept of virtual reincarnations or Avatars has caught the fancy of the masses. These avatars are puppets or cartoonized alter egos of people and creating plausible avatars requires accurate tracking and analysis of the real person's motion. The *pfinder* detection and tracking system of [Wren et al., 1997] is used for creating such avatars, vision driven interfaces for video games and a preprocessor for gesture recognition to decipher American sign languages. It uses a "blob" representation of the human body following the work of [R.J. Kauth and Thomas, 1977] where each pixel on the body belongs to a cluster in a feature space of spatial location and color distribution. These blobs are constructed by automatic detection of a person inside a closed space and tracked then on for other motives. As an aside, blob representations have also been used by SMC filtering based trackers like the BraMBLe multi-blob tracker of [Isard and MacCormick, 2001]. Finally, simultaneous tracking and object recognition systems based on Nearest Neighbour (NN) search object localization has also been proposed [Sakagaito and Wada, 2007].

2.3 Summary

Even this non-exhaustive review gives a fair idea of the omnipresence of visual tracking methods in computational vision. Practically every video processing application beckons to them. These calls have been answered by numerous strategies and frameworks over the years, each of which can be viewed from one aspect or other. Such views directly reflect the diversity of the methods and beyond doubt establishes the difficulty of even the simplest visual tracking problem. This motivates the search for better and more advanced frameworks for visual tracking. Towards this goal, existing strategies may need to be revisited and made compliant with current day frameworks. It is also useful to bear in mind the application scenarios while designing a tracking method, as suitable prior knowledge or interaction can be exploited for maximum benefit. These are the key points which motivate the contributions of this thesis.

3

Probabilistic inference

Consider a random variable $x \in \mathbb{R}^n$ about which some *prior* knowledge is available with some degree of certainty. Let *measurement* or *evidential data* $y \epsilon \mathbb{R}^m$ be known which is *related* to x in some manner and so can say something about x with some degree of certainty. The *inference problem* is to combine this prior and evidential knowledge in a meaningful fashion to produce a statement or *belief* about x. Due to the inherent randomness involved in the description of the prior knowledge and the measurements, this problem is elegantly stated using the language of probability. In particular, the inference problem can be stated using the Bayes rule for conditional probabilities [Papoulis and Pillai, 2002]. By this rule,

$$p(x|y) = \frac{p(y|x)p(x)}{\int p(y|x)p(x)dx}. \tag{3.0.1}$$

In words, the above rule expresses the knowledge of x *conditional* on the given information y, known as the *filtering distribution*, using the prior (knowledge) distribution $p(x)$ and the relationship between y and x stated through the *likelihood function* $p(y|x)$. This rule is very intuitive: if the prior knowledge is very strong, which implies $p(x)$ is peaked, and the likelihood function is diffuse due to uncertain measurements, then the prior has a larger contribution to the filtering distribution and vice-versa. The denominator is a function of the measurement y and the parameters of the prior model, if any. This function acts to normalize the numerator so that the filtering distribution is a probability density function. As a note of interest, the Bayes rule employed above acts as an *inference rule*, that is a way to infer the distribution of the hidden variable under light of evidential data.

The components of Eqn. 3.0.1 will be frequently encountered throughout this manuscript and therefore is discussed in more detail below.

3.1 Prior

The prior incorporates all empirical or otherwise obtained knowledge about the *hidden* random variables of interest to the inference problem. It captures two main forms of knowledge, the *structure* of dependencies (or independencies) between the hidden random variables and the *functional form* or *parameterization* of these dependencies. Suppose the set of hidden variables of interest is denoted as $X_N = \{x_i, i = 0 : N\}$, then a standard way to express the prior knowledge is by supplying a joint distribution of these hidden variables, $p\big(X_n\big)$.
A structure can be imposed on this joint distribution to write this prior as a product of some simpler factors. The general form of this structure is as shown below.

$$p\big(X_N|\theta\big) = \frac{1}{Z\big(\theta\big)} \prod_{C\epsilon\Omega(X_N)} \psi_C\big(x_C|\theta\big), \tag{3.1.1}$$

where $C = \Omega(X_N)$ is a subset of the set X_N. Each factor or *compatibility potential* $\psi_C\big(x_C|\theta\big)$ is therefore a function of only a subset of the hidden variables, $x_C = \{x_i, i\epsilon C\}$. These factors define the dependency structure of the hidden variables.
The proportionality constant (independent of the hidden state) Z is a function of the parameters controlling the prior distribution and is known as the *partition function.* This function can be evaluated as follows.

$$Z\big(\theta\big) = \int \prod_{C\epsilon\Omega(X_N)} \psi_C\big(x_C|\theta\big) dX_N. \tag{3.1.2}$$

Other than in some simple cases, it is difficult to evaluate the above integral due to the dependency structure of the hidden variables and/or due to the functional form of the factors $\psi_C\big(x_C|\theta\big)$ themselves.
An example of a prior distribution is the multi-variate Gaussian prior defined below.

$$p\big(X_N|\theta = \{\mu_N, \Sigma_N\}\big) = \frac{1}{(2\pi)^{d/2}|\Sigma_N|^{d/2}} \exp -\frac{1}{2}\big(X_N - \mu_N\big)^T \Sigma_N^{-1}\big(X_N - \mu_N\big), \tag{3.1.3}$$

where $\theta = \{\mu_N, \Sigma_N\}$ are the parameters (mean and covariance matrices respectively) of the prior distribution and d is the dimensionality of X_N. If $x_i\epsilon\mathbb{R}^n$, then $d = nN$.

In this Gaussian case, the partition function is easily evaluated to be a function of the parameters as shown below.

$$Z(\theta) = \frac{1}{(2\pi)^{d/2}|\Sigma_N|^{d/2}}. \tag{3.1.4}$$

In the Gaussian case the dependency structure between the hidden variables $x_i, i = 1 : N$ is controlled by the inverse co-variance matrix Σ_N^{-1}. If this matrix is diagonal then the hidden variables are independent, each with an *exponential form*. In this case the structure is completely disconnected. On the other end, all the variables are fully connected if none of the elements in the inverse co-variance matrix are zero. Several other possibilities exist mid-way between the two. A convenient way to visualise this dependency structure is provided by what are called graphical models [Pearl, 1997]. Graphical models are a qualitative visual representation of a *dependency model* such as a probability distribution (the prior distribution) from which several conditional independence statements among the hidden variables can be asserted by inspection of the graphs without resorting to complex and cumbersome numerical operations. This representation is very useful to design engines for solving inference problems. Conversely, probability distributions, like the ones representing the prior, can be derived from graphical models which are constructed using a set of qualitative conditional independency statements such as, *a variable X is independent of variable Y given a third variable Z* between subsets of (hidden) variables from the set of variables of interest. These attributes of graphical models have found widespread use in modeling problems in such varied domains as computer vision (visual tracking and image segmentation), digital communications, signal processing, speech processing and model identification. A short note on the kind of graphical models and their representation capabilities is provided below, all along bearing in mind that these models are used to represent the structural dependencies of the prior.

Undirected graphs [Pearl, 1997]

An undirected graph $G = \big(V, E\big)$ contains a set of vertices or nodes V and a set of edges E between certain pairs of vertices or nodes in V. By an undirected graphical representation of a probability distribution over a set of random variables U, it is meant that a direct correspondence between V and elements of U (for example, X_N from earlier discussions) is established such that the topology of G reflects some conditional independency properties among the elements of U. Once this correspondence is established, then both the graph G and the probability distribution are seen to be equivalent, that is $G = \big(U, E\big)$. An example of an undirected graph is shown in Fig. 3.1.

Undirected graphs aim to represent conditional independency statements between

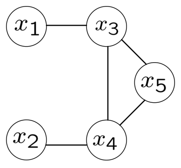


Figure 3.1: An undirected graph.

triplets of disjoint subsets of hidden variables using *vertex separation* on the graph. For instance, if a set of variables X is conditionally independent of another (disjoint) set of variables Y given a set of variables Z (disjoint from X and Y), then the vertices corresponding to the elements of Z intercept all paths going from an element of X to an element of Y.

Two varieties of maps can be obtained using undirected graph representations. The D-map or dependency map in which all dependencies between sets of disjoint variables asserted by the probability distribution are connected by edges on the graph, but one in which some dependencies may go unrepresented; this is because all forms of dependencies cannot be captured by vertex separation on undirected graphs. The I-map or independency map in which all conditional independency statements asserted by the dependency model is represented by vertex separation on the graph, but one in which some conditional independency statements may go unrepresented; having some superfluous edges on the graph.

If all the conditional independency statements asserted by the probability distribution can be faithfully represented by vertex separation on the undirected graph (that is the graph is both an D-map and an I-map), then the graph is called the *perfect map* of the dependency model.

For the example shown in Fig. 3.1, the following conditional independency statements can be read out by inspection (of vertex separation).

$$\begin{aligned} &x_1 \perp \{x_2, x_4, x_5\} | x_3, \quad \text{read } x_3 \text{ separates } x_1 \text{ from the rest of the variables,} \\ &x_2 \perp x_5 | x_4, \quad \text{read } x_4 \text{ separates } x_2 \text{ and } x_5, \end{aligned} \tag{3.1.5}$$

and so on. A convenient way to inspect conditional independency between a set of nodes X and another set Y given a set Z (all disjoint), is to remove from the graph Z

all the edges connected to it, and then see if there are other paths connecting X and Y. If there are, then they are not conditionally independent, else they are.
As mentioned earlier, all forms of dependencies cannot be faithfully represented by vertex separation, in particular, if induced, logical or functional dependencies connect the variables. For example, consider the three random variables x_1, x_2, x_3 related by $x_3 = x_1 x_2$. Let $p(x_1, x_2) = p(x_1)p(x_2)$ (mutually independent) by definition. The D-map for this relation is shown in Fig. 3.2 (by definition, all dependencies must be explicitly shown here, but some dependencies may escape representation). From the

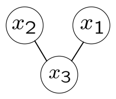


Figure 3.2: Shortcoming of an undirected graph representation.

graph in Fig. 3.2, $p(x_1, x_2|x_3) = p(x_1|x_3)p(x_2|x_3)$, but, given x_3 (an instantiation of it) the variables x_1, x_2 are dependent, a fact which is not represented by the graph. The graph is therefore not an I-map because, if were so, then separated vertices x_1 and x_2 would be conditionally independent. Joining the vertices x_1, x_2 does not help either as this implies x_1, x_2 are dependent even if nothing is known about x_3, contradicting their definition. Therefore (vertex separation on) undirected graphs can offer perfect representations only in the absence of induced, logical or functional dependencies between the variables.
For a probability distribution to have a perfect map requires it to satisfy certain axioms (see [Pearl, 1997] for axiomatic characterization of vertex separation) which would allow the conditional independencies asserted by it to be represented on an undirected graph on which vertex separation implies conditional independence. The presence of induced dependencies between the variables prevents the distribution from satisfying these axioms. Nevertheless, sub-optimal representations in the form of I-maps can still be constructed based on the axiomatic characterization of the notion of conditional independence itself (see [Pearl, 1997]). The most important of them is the *minimal* I-map or Markov network (or field) for a probability distribution (for instance, the prior) which can be constructed (if certain axioms of conditional independence are fulfilled by the probability distribution) in a rule based fashion (edge deletion procedure or Markov boundary approach).
Conversely, one may start from an undirected graphical model and derive a probability

distribution from it. For instance, a graphical structure may be imposed on the hidden state variables according to the designers mental understanding of the problem, say, and a corresponding probabilistic distribution (the prior) can be derived using a product of simpler compatibility potentials and a normalizing function (the partition function). The theorem of Hammersley and Clifford guarantees that the undirected graph will be a perfect map of the derived probability distribution [Besag, 1974].
The probability distribution for which the graph in Fig. 3.1 is a perfect map can be derived using the notion of clique potentials (see Hammersley and Clifford theorem, [Pearl, 1997]). Cliques are defined as maximal sub graphs with nodes which are *all* adjacent to each other. For example, the set $\{x_3, x_4, x_5\}$ is a clique. To each clique C_i a non-negative compatibility function $\psi_i(c_i)$ is assigned which describes the relative strength of each value assignment c_i or *configuration* to the variables in the clique. With this assignment, the probability distribution *quantifying* the conditional independency structure of the graph can be derived as follows (for the example case shown in Fig. 3.1).

$$p(X = \{x_1, \ldots, x_5\}) = \frac{1}{Z}\psi_1(c_1 = \{x_1, x_3\})\psi_2(c_2 = \{x_2, x_4\})\psi_3(c_3 = \{x_3, x_4, x_5\}), \tag{3.1.6}$$

where,

$$Z = \int \left[\prod_{i=1:3} \psi_i(c_i) \right] dX. \tag{3.1.7}$$

In general, it is difficult to define the compatibility functions $g_i(c_i)$ by empirical experience alone as the interaction between the variables in each clique is also dependent on other cliques, making it difficult to "book-keep" all the possible interactions and even if it is possible, evaluating Z can be computationally expensive or intractable. Fortunately, some graphical models like Hidden Markov Models (HMM) or Markov trees allow a natural decomposition of the probability distributions they represent so that there is a direct relation between marginal probabilities of the variables of a clique and the compatibility functions. Such models are widely employed. However such graphical models are based on a designers viewpoint, which is itself based on experiential learning, and may or may not reflect the true nature of the conditional independence structure among the variables. A simple graphical structure is usually justified in a computational complexity sense, the true structural dependencies may be more complex.
Next, the inability of undirected graphs to represent induced dependencies is rectified

using the richer representation of directed graphs.

Directed graphs [Pearl, 1997]

Consider the following directed acyclic graph (DAG), that is a graph with arrows on the edges and without having directed loops. In DAG's conditional independency state-

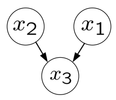


Figure 3.3: A directed graph for $x_3 = x_1 x_2$

ments can be read using the *d-separation* rule [Pearl, 1997]. This rule prescribes what connections on the graph are made (activated) or broken (blocked) if knowledge of a set of (conditioning) variables is given. In verbatim, recounting [Pearl, 1997],

If $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ are three disjoint subsets of nodes on a DAG D, then $\mathbf{Z}$ is said to *d-separate* $\mathbf{X}, \mathbf{Y}$, if along every chain of nodes between a node in $\mathbf{X}$ and a node in $\mathbf{Y}$ there is a node w satisfying one of the following two conditions: (1) w has converging arrows and none of w or its descendants are in $\mathbf{Z}$, or (2) w does not have converging arrows and w is in $\mathbf{Z}$.

For the graph in Fig. 3.3, this rule implies x_1, x_2 are *mutually independent* if nothing is known about x_3 or the path between them is *blocked* and this path is *activated* upon knowledge of x_3. This notion is exactly what undirected graphs could not capture.

A more complex DAG is shown in Fig. 3.4. The following statements can now be read out by inspection.

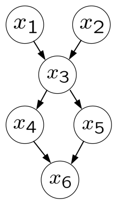


Figure 3.4: A directed graph for $x_3 = x_1 x_2$

1. x_1 is mutually independent of x_2.

2. x_4 may be dependent on x_5 (under no conditioning; "may be" because there is a chance that $p(x_4, x_5) = p(x_4)p(x_5)$ due to numerical compatibility, purely by coincidence).

3. Given both x_3 and x_6, x_4 may be once again be dependent on x_5 (two paths between x_4 and x_5).
4. Given x_3, x_6 is independent of x_1.
5. Given x_4, x_6 maybe dependent on x_1 (two paths between x_6 and x_1).

Similar to undirected graphs, minimal directed I-maps or Bayesian networks asserting *all conditional independencies* present in the probability distribution (for instance, the prior) can be constructed, if the probability distribution satisfies certain fundamental axioms (called semi-graphoid axioms) governing conditional independence [Pearl, 1997]. Conversely, a probability distribution can be derived from a DAG such that the DAG is a perfect map of the probability distribution, an attribute which is widely used. For example, for the tree structure shown in Fig. 3.4, the probability distribution can be developed as follows.

$$p(x_1,x_2,x_3,x_4,x_5,x_6) = p(x_6|x_4,x_5)p(x_4|x_3)p(x_5|x_3)p(x_3|x_2,x_1)p(x_1)p(x_2), \tag{3.1.8}$$

or, written more abstractly,

$$p(x_1,x_2,x_3,x_4,x_5,x_6) = \prod_{i=1:6} p(x_i|Pa_i), \tag{3.1.9}$$

where Pa_i represents the set of *parent nodes* of the node i. As another frequently encountered example, consider the first order HMM shown in Fig. 3.5, for which,

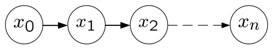


Figure 3.5: A first order Hidden Markov Model.

$$p(x_{0:n}) = \prod_{i=1:n} p(x_i|x_{i-1})p(x_0). \tag{3.1.10}$$

For Bayesian networks wherein there is a sequential chaining of the variables as in Fig. 3.5, the terms *process model* or *state evolution* model is sometimes used to describe the probability distribution (prior) derived from such models. As a note of interest, the process model could, for instance, be defined using a one step relationship as shown below.

$$x_n = x_{n-1} + \eta_n, \text{ where, } \eta_n \perp x_{0:n-1}. \tag{3.1.11}$$

where η_n is a i.i.d process noise model whose parameters need to be specified or determined by a learning step.

To summarise the discussions on the prior, the structural knowledge can be incorporated directly using Markov or Bayesian networks or even a combination of them depending on the nature of the dependencies between the hidden variables or the designers intent and experience. The compatibility potentials over the cliques on the graph model capture the functional form (strengths of the cliques) of the knowledge (potentials are usually drawn from the exponential family for ease of integration's sake). With the structural and functional form defined, the prior distribution is completely established.

3.2 Data likelihood

The relationship between the *observed* or *measured* data y and the hidden variable(s) of interest x is given by the (hidden) data likelihood or just likelihood function $p\big(y|x\big)$. The likelihood is sometimes explained as an *inverse probability*, that is, given the data y, the probability of some sample of x is represented by this likelihood function.
The terms observed data, measured data or simply data are used interchangeably, although in general, the observed data is taken to be the sequence of images and the measured data could be parts of an image(s), some feature(s) (say, an edge map, optic flow fields) or quantities (say, edge corners, motion model parameters) derived from it, see [Isard and Blake, 1996; Arnaud et al., 2004] for relevant examples. Whatever the nature of the data, the important fact is that it can be observed or measured by a device (a camera, for example) *independently* from the hidden variables and some inference about the hidden variables can be drawn from it through an *a priori* prescribed *observation process* or model. For instance, a simple linear observation process is shown below.

$$y = Hx + \zeta, \tag{3.2.1}$$

where H is a parameter and ζ is the observation noise. The parameters of the model and the observation noise must be established by empirical knowledge or in other words, these parameters must be *learnt* from examples. However, this is very difficult to do, for example, if the data is a sequence of images (usually of dimension in the order 10^6) and the hidden variable is, say, the position and some other attributes of an object, scale, contour (usually of a dimension $\leq 10^2$), then establishing a relationship between the data and the hidden variable is indeed a phenomenal task, even if it can be achieved. Further, the problem of establishing this relationship as a conditional probability distribution requires computation of some very high dimensional (in the order $\geq 10^6$) integrals over highly non-linear integrands, which in general is intractable. In an attempt to handle the dimensionality issue, the general practice in computer vision is to define *low dimensional feature matching distance* based likelihood functions, usually of the following form.

$$p(y|x) = \frac{1}{Z_x} \exp\left\{\lambda\big(f(y,x)\big)\right\}, \tag{3.2.2}$$

where λ is a control parameter, $f(y,x)$ is a matching distance, say the Bhattacharya coefficient for matching histograms [Perez et al., 2002] and Z_x is the normalizing factor given as follows.

$$Z_x = \int_y \exp\left\{\lambda\big(f(y,x)\big)\right\} dy, \tag{3.2.3}$$

where the integral is to be evaluated over the space of all possible data (imagine the space of all possible images) and not just the observed data. If this is not possible, then comparisons such as $p\big(y|x = x^i\big)$ and $p\big(y|x = x^j\big)$ for instantiations or samples x^i, x^j cannot be performed. If by some means it is possible to evaluate the above integral, the result (normalizing factor) still remains a function of x. The common but *flawed practice* is to ignore the dependence of Z_x on x (so treating it as a constant), which implicitly biases the likelihood towards samples with larger Z_x. These are undesired or trivial samples which usually provide high match scores (see [Minka, 2004] for a succint description).

The correct, albeit difficult, approach would be to evaluate the proper normalizing factors. A recent attempt to tackle this problem is the PDF projection theorem (for data classification problems) proposed by [Baggenstoss, 2003]. The idea is to develop the (normalized) likelihood function of the original high dimensional data from a normalized likelihood function over low dimensional features extracted from the *training* data and a nominal or base distribution of the original high dimensional data (the distribution of images in the training set). However, even arriving at a rough estimate of the base distribution can be a difficult task in practice, especially for filtering problems in high dimensional state spaces. Further research in this direction may reveal new methodologies to resolve this problem. Meanwhile an important note to bear in mind is that employing such *biased likelihood functions* can lead to less informative filtering distributions, meaning the uncertainty in the estimation of the state described by the filtering distribution may be unreliable or even erroneous. Some respite to this practical problem can be found by *averaging* over several measurements (likelihood functions), an aspect which motivates the contributions in Chapter 5.

3.3 Evidence of prior model parameter(s)

Consider the denominator in Eqn. 3.0.1 with the prior parameterized by θ as shown below.

$$p(y|\theta) = \int p(y|x,\theta)p(x|\theta)dx. \tag{3.3.1}$$

This is the probability distribution of the data which acts as the normalizing factor in the Bayes rule, Eqn. 3.0.1. At the same time, this may be viewed as the *evidence of the prior model parameter* θ given the data y or simply the likelihood function of θ given the data y. Note that the likelihood function $p(y|x)$ is sometimes also called the model evidence, especially in data classification literature (see [Bishop, 2006] for discussion on likelihood ratios or Bayes factors); this must not be confused with the evidence of the prior model parameter θ.

As an example, if the likelihood function and the prior on the hidden state variables are both Gaussian, then the distribution of the data is also a Gaussian as shown below.

$$p(y|\mu,\Sigma) = N(y;\mu = H\mu_x, \Sigma = H\Sigma_x H^T + \Sigma_y) = \int N(y;\mu_y = Hx, \Sigma_y)N(x;\mu_x,\Sigma_x)dx, \tag{3.3.2}$$

where the notation $N(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left\{\frac{-1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$ is used throughout this manuscript.

In most computer vision problems it is difficult to evaluate the model parameter evidence due to difficulties involved in either normalizing the likelihood function (model evidence) or computing the partition function of the prior on the hidden variables or in the worst case, both. Therefore other approximations of the prior model parameter evidence, using deterministic quadrature or Monte Carlo simulations need to be invoked.

3.4 Sequential filtering problem

The sequential filtering problem is to evaluate or equivalently, *propagate* the following *posterior* distribution.

$$p(x_{0:n}|y_{1:n}) = \frac{p(y_{1:n}|x_{0:n})p(x_{0:n})}{\int p(y_{1:n}|x_{0:n})p(x_{0:n})dx_{0:n}}. \tag{3.4.1}$$

On the right hand side of the above equation is a formidable inversion problem, that is, computing the likelihood of the sequence of hidden random variables $x_{0:n}$ given the *sequentially collected data* $y_{1:n}$. Secondly, the dimensionality of this filtering problem increases at each time step as more and more variables (both hidden and observed) are appended into the equation above. Thirdly, at each time step the memory required to store the sequential observed data increases. However, all these issues maybe dealt with depending on the structure of the prior over the hidden variables and the way the observed data is related to the hidden variables. Graphical models once again provide a convenient modeling and visualization tactic to make this sequential filtering problem tractable. The most popular model is the first order HMM (with data included in gray filled circles) Bayesian network shown in Fig. 3.6. For this directed graph, the following conditional independency statements can be asserted in keeping with the d-separation rule.

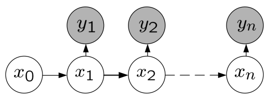


Figure 3.6: A model for sequential filtering

$$\begin{aligned} &p(y_n|x_{0:n}, y_{1:n-1}) = p(y_n|x_n), \\ &p(x_{n+1}, x_{n-1}|x_n) = p(x_{n+1}|x_n)p(x_{n-1}|x_n), \text{using d-separation rule.} \end{aligned} \tag{3.4.2}$$

With these statements the filtering problem can be simplified as shown below.

$$\begin{aligned} p(x_{0:n}|y_{1:n}) =& \frac{p(y_n|x_n)p(y_{1:n-1}|x_{1:n-1})p(x_n|x_{0:n-1})p(x_{0:n-1})}{p(y_{1:n})} \\ =& \frac{p(y_n|x_n)p(x_n|x_{n-1})p(x_{0:n-1}|y_{1:n-1})}{p(y_n|y_{1:n-1})} \end{aligned} \tag{3.4.3}$$

A recursive form arises from Eqn. 3.4.3, the posterior at instant n can be developed from the posterior at instant $n-1$, the likelihood function at n and the low order one step process model. The *filtering distribution* too can be obtained from Eqn.3.4.3 by appropriate marginalization as shown below.

$$\begin{aligned} p(x_n|y_{1:n}) &= \int p(x_{0:n}|y_{1:n})dx_{0:n-1} \\ &\propto p(y_n|x_n)\int p(x_n|x_{n-1})p(x_{0:n-1}|y_{1:n-1})dx_{0:n-1} \\ &= p(y_n|x_n)\int p(x_n|x_{n-1})p(x_{n-1}|y_{1:n-1})dx_{n-1}, \end{aligned} \tag{3.4.4}$$

which upon using the Chapman-Kolmogorov rule, yields,

$$p(x_n|y_{1:n}) \propto p(y_n|x_n)p(x_n|y_{1:n-1}). \tag{3.4.5}$$

The term $p(x_n|y_{1:n-1})$ is sometimes called the *effective prior*, which is a slight abuse of terminology as a prior should be independent of any observed or measured data. The proportionality factor in Eqn. 3.4.5 can be evaluated as follows.

$$p(y_n|y_{1:n-1}) = \int p(y_n|x_n)p(x_n|y_{1:n-1})dx_n. \tag{3.4.6}$$

This factor can be seen as a *measurement prediction* distribution (see the application of this density to *measurement gating* in multi-target tracking [Vermaak et al., 2005]). In another way this distribution is once again the likelihood or evidence of the parameters of the prior model as described in Sec. 3.3. Finally, depending on the nature of the application, either the posterior or the filtering distribution may need to be propagated. The propogation of the posterior or the filtering distributions is dependent on the functional form of the prior (the process model) and the likelihood function (the observation model). Only in a limited number of cases can posterior/filtering distributions be propagated by analytical evaluations. In all other cases, more general strategies must be resorted to. Both these cases are described below in order.

3.4.1 Analytical solutions

Consider the following state space model (the process + observation model) in $\Re^n$.

$$x_n = Ax_{n-1} + \eta_n, \text{ where, } \eta_n \perp x_{0:n-1}. \tag{3.4.7}$$

where A is a parameter and the process noise $\eta_n \sim \mathcal{N}(\eta_n; 0, P)$.

$$y_n = Hx_n + \zeta_n, \tag{3.4.8}$$

where H is a parameter and the observation noise $\zeta_n \sim \mathcal{N}(\zeta_n; 0, Q)$. With these definitions, the filtering distribution can be evaluated as follows.

$$p(x_n|y_{1:n}) \propto \mathcal{N}(y_n; Hx_n, Q) \int \mathcal{N}(x_n; Ax_{n-1}, P)\mathcal{N}(x_{n-1}; \mu_{n-1}, \Sigma_{n-1})dx_{n-1}, \tag{3.4.9}$$

where the filtering distribution at instant $n-1$ is represented as $\mathcal{N}(x_{n-1}; \mu_{n-1}, \Sigma_{n-1})$. The effective prior can be computed using standard rules for Gaussian distributions (see [Bishop, 2006]) as shown below.

$$p(x_n|y_{1:n-1}) = \mathcal{N}(x_n; \mu_{eff} = A\mu_{n-1}, \Sigma_{eff} = A\Sigma_{n-1}A^T + P) \tag{3.4.10}$$

Upon meshing the effective prior with the likelihood function the following Gaussian form of the filtering distribution is obtained,

$$p\big(x_n|y_{1:n}\big) = \mathcal{N}\big(x_n; \big(H^T Q^{-1} H + \Sigma_{eff}^{-1}\big)^{-1}\big(H^T Q^{-1} y_n + \Sigma_{eff}^{-1}\mu_{eff}\big), \big(H^T Q^{-1} H + \Sigma_{eff}^{-1}\big)^{-1}\big). \tag{3.4.11}$$

This filtering technique is the celebrated Kalman filter which propagates Gaussian distributions. After some manipulations, this result is usually expressed in another form using the Kalman gain factor $K = \Sigma_{eff} H^T \big(Q + H\Sigma_{eff} H^T\big)^{-1}$, through which,

$$p\big(x_n|y_{1:n}\big) = \mathcal{N}\big(x_n; \mu_{eff} + K\big(y_n - H\mu_{eff}\big), \big(I - KH\big)\Sigma_{eff}\big), \tag{3.4.12}$$

where the expression $\big(y_n - H\mu_{eff}\big)$ is known as the *innovation* term. This term is the difference between the actual measurement at n and the predicted measurement using the effective prior estimate μ_{eff}. The Kalman gain then balances the contributions of the innovation and μ_{eff}. If the gain K is large then the the measurement variance is small compared to the effective prior variance and so contributes more to the filtered estimate than the effective prior, and vice-versa. The parameters of the state space model A, P, H, Q can all be time variant (if they are updated online) without any change to the filtering equation.

In general, analytical recursive (to avoid storage problems) solutions (see mean and covariance terms in Eqn. 3.4.12) can be obtained for any member of the exponential family of distributions (Gaussian, Gamma, Poisson, Beta distributions, see [Minka, 1999]) as these are closed under multiplication (filtering is in effect multiplication of the prior and likelihood function) and can be described by a finite number of sufficient statistics. Recursions for non-Gaussian distributions in the exponential family will however be different from the Kalman filter recursions.

The Kalman filter is also employed for linearlized formalisms of non-linear state space models. The general non-linear state space model is given as follows.

$$x_n = f_{n-1}\big(x_{n-1}, \eta_{n-1}\big), \tag{3.4.13}$$

which upon linearization using the Taylor's series expansion,

$$x_n = F_{n-1} x_{n-1} + b_{n-1} + \eta^*_{n-1} + \eta_{n-1}, \tag{3.4.14}$$

where, $F_{n-1} = \frac{\partial f_{n-1}}{\partial x_{n-1}}\Big|_{\hat{x}_{n-1}}$, $b_{n-1} = f_{n-1}\big(\hat{x}_{n-1},\big) - F_{n-1}\hat{x}_{n-1}$, η^*_{n-1} represents the *linearization error* (or residue) with co-variance P^*_{n-1} (notice the time index here). A similar linearization of a (possibly) non-linear observation model can be made as shown below.

$$y_n = h_n\big(x_n, \zeta_n\big), \tag{3.4.15}$$

which upon linearization using the Taylor's series expansion,

$$y_n = H_n x_n + d_n + \zeta_n^* + \zeta_n, \tag{3.4.16}$$

where, $H_n = \left.\frac{\partial h_n}{\partial x_n}\right|_{\tilde{x}_n}$, $\tilde{x}_n$ is the predicted state estimate derived from the process model, $d_n = h_n\big(\tilde{x}_n,\big) - H_n\tilde{x}_n$, η_n^* represents the *linearization error* (or residue) with co-variance Q_n^*.

The different approaches to filtering with non-linear state space models through the Kalman filter differ in the way they deal with the linearization and the linearization error. For instance, the Extended Kalman Filter (EKF) linearizes the process (and/or observation) model around the past (and/or predicted) estimate and ignores the resulting linearization error(s). The Linear Regression Kalman Filter (LRKF) linearizes the process (and/or observation) model by linear regression over a set of sampled points around the past (and/or predicted) estimate and the (non-linear) function values obtained by the respective process (and/or observation) model(s). The sample points themselves are so selected that the sample statistics matches the statistics of the past estimate. The LRKF also takes into account the linearization error in the filtering process, thus reducing bias in the filtered estimates. It is interesting to note that the LRKF is a general formulation of the well known Unscented Kalman Filter (UKF). For a more detailed comparative review see [Schutter, 2001].

In the foregoing state space models it was assumed that the functional form of the non-linearities (the relationship between the noise terms, hidden state terms and the observed data) are known and the statistics of the noise terms are supplied. Further, it was implicitly assumed that the non-linearities were tractable to some method of linearization. In a more general case, such linearizations may be intractable, especially in the observation model, due to the unknown statistics of the noise and the form of the non-linear relationship between the high dimensional observed data , the lower dimensional hidden state and the corrupting noise. Things turn more complex if the observed data itself is a function of the hidden state (see the discussions in Sec. 3.2). Therefore, it is difficult in such cases to supply even a general non-linear observation model of the form in Eqn. 3.4.15. In all, only an evidence or likelihood for a hypothesis of the hidden state may be computable. Other issues may arise on the side of the process model too, making it difficult to write down the low order conditional probabilities such as $p(x_n|x_{n-1})$ and instead only an *unnormalized compatibility* function (see Sec. 3.1) between the hidden state variables may be accessible. Another way to summarise these difficulties is to view the filtering problem as one of propagating arbitrary (out of the exponential family) distributions. Approximate techniques to solve the filtering problem with these restrictions have been developed. These are considered below.

3.4.2 Simulation methods

Monte Carlo simulation methods are used to tackle two problems:
1. The sampling problem, which is to draw independent samples from a probability distribution $p(x)$.
2. The approximation problem, which is to evaluate expectations of the form $\hat{f} = \int f(x)p(x)dx$, where $f(x)$ is some function of the random variable x.
If the first task is solved then the second follows immediately from the property of the law of large numbers (LLN) [Papoulis and Pillai, 2002];

$$\hat{f} \approx \frac{1}{M} \sum_{i=1:M} f(x^i),\, x^i \sim p(x), i = 1 : M. \tag{3.4.17}$$

However, in several cases, the probability distribution may only be available upto a multiplicative constant, that is one may be able to evaluate a function $p^*(x)$ such that $p(x) \propto \frac{1}{Z^*} p^*(x)$. The computation of the normalization constant Z^* may require an enormous amount of effort if the dimension of the state space (dimension of x) is large. Even if x were discrete in dimension $N = 1000$ and its elements were binary, it would require 2^{1000} additions to compute Z^*. If x took continuous values, then the $Z^* = \int p^*(x)dx$ could be analytically intractable in all but a few cases.
An alternative way to compute expectations of the form shown in Eqn. 3.4.17 is to evaluate the integral at uniformly drawn samples in the state space, by which,

$$\hat{f} \approx \sum_{i=1:M} f(x^i) \frac{p^*(x^i)}{\sum_{j=1:M} p^*(x^j)}. \tag{3.4.18}$$

The success of the above technique depends on how many samples lie within the *typical set* of $p(x)$. The typical set, defined as $|T| \approx 2^{H(x)}$, where $H(x)$ is the Shannon-Gibbs entropy of $p(x)$, is a small region in the state space where the majority of the probability mass of the distribution is concentrated (especially in higher dimensions). Therefore, if sufficient samples do not "hit" the typical set (typically requires $2^{N-H(x)}$

samples, a very large number indeed for high dimensions), then the estimate may be under valued. Other alternatives have been developed to handle such problems. Of these the technique of importance sampling is widely used (including this thesis) and is described in detail below. Subsequently other relevant Monte Carlo techniques are briefly mentioned.

1. **Importance sampling (IS)**

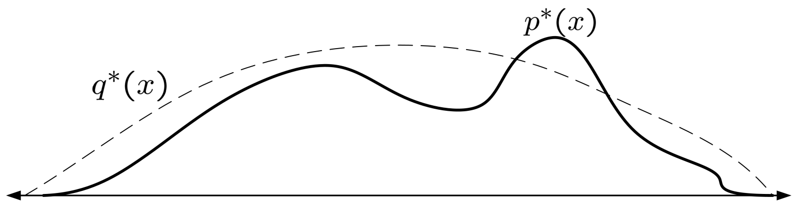


Figure 3.7: Importance sampling - an illustration

This technique is used to evaluate expectations of the form shown in Eqn. 3.4.17. The idea is as follows. Select a proposal or importance sampling density $q(x) = \frac{1}{Z_q} q^*(x)$ which is easy to sample from (Z_q is known) and also evaluate (see Fig. 3.7). With this, consider the following manipulation.

$$\begin{aligned}\hat{f} &= \int f(x) \frac{p^*(x)}{\int p^*(x) dx} dx, \\ &= \int f(x) \frac{p^*(x)}{q(x)} \frac{1}{\int \frac{p^*(x)}{q(x)} q(x) dx} q(x) dx.\end{aligned} \tag{3.4.19}$$

Draw independent samples $x^i \sim q(x), i = 1 : M$. Appealing to the law of large numbers,

$$\hat{f} \approx \sum_{i=1:M} f(x^i) \frac{p^*(x^i)}{q(x^i)} \frac{1}{\sum_{i=1:M} \frac{p^*(x^i)}{q(x^i)}}. \tag{3.4.20}$$

Terming $w^i \equiv \frac{p^*(x^i)}{q(x^i)}$ as the *unnormalized* importance weight, the expectation is approximated using *normalized* importance weights $\sum_{i=1:M} \widetilde{w}^i = 1$ as shown below.

$$\hat{f} \approx \sum_{i=1:M} \widetilde{w}^i f(x^i). \tag{3.4.21}$$

It is informative to observe that the intractable normalizing constant is obtained by consequence of the above approximation as given below.

$$\int p^*(x)dx \approx \sum_{i=1:M} w^i. \tag{3.4.22}$$

The (normalized) importance weights can be seen as a *corrective weight* being attached to each sample from $q(x)$ as it is not being sampled from the true distribution $p(x)$. At samples where $q(x) < p(x)$ the samples are assigned a greater weight to offset their under representation and vice-versa.

The practical difficulty with importance sampling lies in the fact that the variance of the estimator $\hat{f}$ cannot be reliably indicated through the empirical variance of $\widetilde{w}^i f(x^i)$. If the *value* of $\left|f(x)p^*(x)\right|$ is large where $q(x)$ is small, then even after a large number of samples have been drawn, the empirical estimate of the expectation can be drastically wrong with no indication in the empirical variance of the estimate that the true variance of the estimate is large. This is the weight variance problem in importance sampling (see [Arulampalam et al., 2002]). Due to this reason importance sampling densities with *heavy tails* is advocated in practice.

In higher dimensions importance sampling suffers from two problems. First the sampling density $q(.)$ must have a typical set similar to that of $p(.)$ so that a large number of samples come from this set, which then requires $q(.)$ to be a good approximation of $p(.)$ and second, even if the samples lie in the typical set their importance weights vary by a large order of magnitude, thereby increasing the variance of the estimate. Some solutions have been proposed to address this weight variance problem (see importance resampling (IR) technique in [Arulampalam et al., 2002]).

The importance sampling technique can also be manipulated to approximate probability distributions using the following tautology.

$$\begin{aligned} p(x) &= \int \delta(x-\tau)p(\tau)d\tau, \\ &= \int \delta(x-\tau)\frac{p^*(\tau)}{q(\tau)}\frac{1}{\int \frac{p^*(\tau)}{q(\tau)}q(\tau)d\tau}q(\tau)d\tau, \end{aligned} \tag{3.4.23}$$

which upon drawing samples $\tau^i \sim q(\tau)$, can be approximated using the *normalized sample-set* $\{\tau^i, \widetilde{w}^i\}_{i=1:M}$ as shown below.

$$p(x) \approx \sum_{i=1:M} \widetilde{w}^i \delta(x - \tau^i). \tag{3.4.24}$$

This approximation of distributions is what is used in making sample-set approximations of the posterior distribution of Eqn. 3.4.3. The following approach is commonly used.
Sample $x^i_{0:n} \sim q(x_{0:n}|y_{1:n}), i = 1 : M$ and evaluate unnormalized importance weights as shown below.

$$w^i_n = \frac{p(y_n|x^i_n)p(x^i_n|x^i_{n-1})p(x^i_{0:n-1}|y_{1:n-1})}{q(x^i_{0:n}|y_{1:n})}, i = 1 : M. \tag{3.4.25}$$

Normalizing these weights appropriately gives,

$$p(x^i_{0:n}|y_{1:n}) \approx \sum_{i=1:M} \widetilde{w}^i_n \delta(x_{0:n} - x^i_{0:n}). \tag{3.4.26}$$

This weight computation can be made *recursive* by virtue of the following factorisation.

$$\begin{aligned} w^i_n =& \frac{p(y_n|x^i_n)p(x^i_n|x^i_{n-1})p(x^i_{0:n-1}|y_{1:n-1})}{q(x^i_n|x^i_{0:n-1}, y_{1:n})q(x^i_{0:n-1}|y_{1:n-1})}, i = 1 : M \\ =& \widetilde{w}^i_{n-1} \frac{p(y_n|x^i_n)p(x^i_n|x^i_{n-1})}{q(x^i_n|x^i_{0:n-1}, y_{1:n})}, i = 1 : M. \end{aligned} \tag{3.4.27}$$

Notice that the weight recursion proceeds from the *normalized weights* at instant $n-1$ to *unnormalized weights* at instant n. One issue remains, that is importance sampling of *sample trajectories* $x^i_{0:n}, i = 1 : M$ which increases in dimension at each instant (if $x \in \mathbb{R}^N$, then $\mathcal{O}(n\mathbb{R}^N)$). As discussed earlier, sampling is very high dimensions is inefficient and leads to biased estimates. However, if the proposal density can be factorised into low order conditional probability distributions then this Markovian nature of the proposal can be leveraged to sample in low dimensions. For example, if the proposal density is (sub-optimally) chosen to be a first-order Markovian process prior $p(x_{0:n}) = p(x_0) \prod_{k=1:n} p(x_k|x_{k-1})$, then at each stage of importance sampling the trajectory samples can be obtained via *sample augmentation* as $x^i_{0:n} = \{x^i_n, x^i_{0:n-1}\}_{i=1:M}$, where $x^i_n \sim p(x_n|x^i_{n-1})$. In this case the weight recursion largely simplifies as shown below.

$$w^i_n = \widetilde{w}^i_{n-1} p(y_n|x^i_n), i = 1 : M. \tag{3.4.28}$$

The use of the prior as the proposal density is widespread but nevertheless sub-optimal. The optimal proposal density can be derived with a view to minimize the empirical variance of the importance weights, thereby hoping to minimize the empirical variance of the posterior. To this end, upon computing the variance of the importance weight $w_n^i(x_n)$ conditional upon the trajectory $x_{0:n-1}^i$ and observed data $y_{1:n}$,

$$var_q\left[w_n^i(x_n)|x_{0:n-1}^i, y_{1:n}\right] = \left(\widetilde{w}_{n-1}^i\right)^2 \left[\int \frac{\left(p(y_n|x_n)p(x_n|x_{n-1}^i)\right)^2}{q(x_n|x_{0:n-1}^i, y_{1:n})} dx_n - p(y_n|x_{n-1}^i)^2\right]. \tag{3.4.29}$$

The above variance is zero if, $q(x_n|x_{0:n-1}^i, y_{1:n}) = p(x_n|x_{n-1}^i, y_n)$. To verify this it is only necessary to note that this form of proposal can be rewritten as follows.

$$p(x_n|x_{n-1}^i, y_n) = \frac{p(y_n|x_n)p(x_n|x_{n-1}^i)}{p(y_n|x_{n-1}^i)}. \tag{3.4.30}$$

This optimal proposal is rarely used in practice as it requires sampling from the distribution $p(x_n|x_{n-1}^i, y_n)$ which involves the current observed data and secondly it requires evaluation of the density $p(y_n|x_{n-1}^i) = \int(y_n|x_n)(x_n|x_{n-1}^i dx_n)$ so as to compute the importance weights as follows.

$$w_n^i = \widetilde{w}_{n-1}^i p(y_n|x_{n-1}^i), i = 1:M. \tag{3.4.31}$$

The above evaluations are only possible for a very restrictive case of state space models (see [Doucet et al., 2000] for some examples).

2. **Rejection sampling**

The technique of rejection sampling is used to draw samples from distribution $p(x) \propto p^*(x)$ which can only be evaluated upto a multiplicative constant, as before. To this end a simpler proposal density $q(x) = \dfrac{q^*(x)}{Z_q}$ is chosen for sampling with a requirement that for some positive constant c, $cq^*(x) > p^*(x)$. Once c is found, a sample $x' \sim q(x)$ is drawn and another sample u is drawn from an uniform distribution in the range $\left[0, cq^*(x')\right]$. If, $u > p^*(x)$, then x' is rejected otherwise it is accepted as a sample from $p(x)$.

The intuition behind rejection sampling is simple: the probability $p(u \leqslant p^*(x')) = \dfrac{p^*(x')}{cq^*(x')} \propto p^*(x')$, therefore it must be that the probability of the sample $p(x') \propto p^*(x')$ as desired.

Like importance sampling, rejection sampling too is inefficient in high dimensions, in general $c \propto \exp(N)$, where N is the dimension and the *acceptance rate* for samples goes as $\frac{1}{c}$, which decreases as the dimension grows. In general it is even difficult to evaluate c as the modes of $p^*(x)$ are difficult to compute in high dimensional spaces or due to its complex nature. Therefore, alternative techniques are employed for sampling in high dimensions.

3. **The Metropolis method**

   This is an iterative method for drawing independent samples from a distribution $p(x) \propto p^*(x)$ which can only be evaluated upto a multiplicative constant, as before. The idea is to draw a sample $x' \sim q(x; x_t)$ where the sampling density is a function of the current state x_t (there is no restriction on the form of this density). With this, define,

$$a = \frac{p^*(x')}{p^*(x_t)} \frac{q(x_t; x')}{q(x'; x^t)}. \tag{3.4.32}$$

   If $a \geq 1$, then the sample x' is accepted, else it is accepted with probability a. If the step is accepted then $x_{t+1} = x'$ or if the step is rejected, then $x_{t+1} = x_t$. it is proven that following this procedure $x_t \sim p(x)$ as $t \to \infty$ (see [Mackay, 1998]). This method is an example of a Markov chain Monte Carlo (MCMC) method as the probability distribution of each state x_{t+1} depends on the previous state x_t. Since *the states are correlated, it is usually necessary to run several iterations to obtain independent samples* from $p(x)$ (because as $t \to \infty$, $p(x)$ tends towards the *invariant distribution* of the Markov chain $q(x; x_t)$, see [Mackay, 1998] for more details).

   The advantage of the Metropolis (also called Metropolis-Hastings) algorithm is that it is independent of the dimension of the state (the sampling density need not satisfy any specific properties on its form, unlike importance and rejection sampling), but it is controlled by the size of the Markov chain transitions in the state space. Small transitions are useful in exploring high dimensional spaces, but take a lengthy duration to generate an independent sample. On the other hand large transitions may fall outside the typical set of $p(x)$. However there have been some successful attempts at improving the convergence speed of MCMC schemes (hybrid Monte Carlo, see [Mackay, 1998]), although, it remains difficult to analyse the convergence of MCMC schemes.

4. **Gibbs sampling**

   Also known as the *heat bath* method, this sampling technique is used to sample from distributions over several variables (minimum two). This is also a Metropolis method where the proposal densities are the conditional distributions of the variables which can be easily sampled from. This is a special case of the Metropolis sampling scheme where every new sample is always accepted.

   As an example, to draw a sample from the distribution over two variables $p\big(X = x_1, x_2\big)$, a sample $x_1^{t+1} \sim p\big(x_1|x_2^t\big)$ and then a sample $x_2^{t+1} \sim p\big(x_2|x_1^{t+1}\big)$ is drawn to obtain X^{t+1}. This iterative procedure is followed to convergence. Since the procedure is a Metropolis method the probability distribution of X^t converges to $p\big(X\big)$as $t \rightarrow \infty$.

   The advantage of Gibbs sampling is that there are no adjustable parameters like the other methods. The disadvantages of the Metropolis class of methods however remains due to the random walk nature of exploration of the state space. For a more elaborate description of all the aforementioned techniques, the reader is referred to [Mackay, 1998].

3.5 Inference on graphs - message passing schemes

This section describes a general technique to compute beliefs or posterior probability distributions of hidden variables in Bayesian and Markov networks under light of observed data or evidence. This foundation of this technique lies on passing *messages*, which are functions of random variables. If the random variables are discrete, then messages can be represented by vectors or else if they are continuous, then they can be represented as continuous functions. In either case, if the messages are normalized to sum to unity, then they act as probability distributions of the corresponding random variables.

The description of the technique follows largely from the one found in [Pearl, 1997]. For the sake of generality, it is assumed that the variables are *continuous* in nature (replace all integrals by sums for discrete variables). Bayesian networks are considered first.

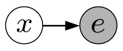


Figure 3.8: A simple directed chain.

3.5.1 Bayesian networks

1. **One link chain**

$$b(x) \equiv p(x|e) = \beta\lambda(x)\pi(x), \tag{3.5.1}$$

where, e is the observed data or evidence. It is emphasized that evidence or observed data includes any *external knowledge* input to the network (usually represented by a shaded node) and/or all instantiations of hidden variables in the network. $\lambda(x) = p(e|x)$ is the likelihood, also called the *diagnostic message*, and $\pi(x) = p(x)$ is called the causal or *predictive message* (the prior or predictive prior), see Fig. 3.8. $\beta = \dfrac{1}{p(e)}$ is a normalization constant to ensure the belief sums to unity. In order to keep the notations clear, in the remaining discussions the constant β is to be understood as an *appropriate* normalization factor, wherever it appears.

2. **Causal chain**

Consider the belief of node x on the chain (see Fig. 3.9).

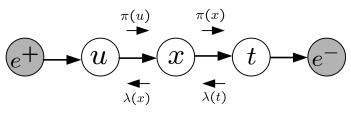


Figure 3.9: A causal chain - evidence on both sides.

$$b(x) \equiv p(x|e = \{e^-, e^+\}) = \beta\lambda(x)\pi(x), \tag{3.5.2}$$

where,

$$\lambda(x) \equiv p(e^-|x), \tag{3.5.3}$$

and,

$$\pi(x) \equiv p(x|e^+), \tag{3.5.4}$$

in which e^- denotes all observed data or evidence at the tail end of the chain and similarly, e^+ denotes all observed data or evidence at the head end of the chain.

$$\lambda(x) \equiv p(e^-|x) = \int p(e^-|t)p(t|x)dt = \int \lambda(t)p(t|x)dt. \tag{3.5.5}$$

The λ messages propagate backwards along the chain using the above rule.

$$\pi(x) \equiv p(x|e^+) = \beta \int p(x|u)p(u|e^+)dt = \int p(x|u)\pi(u)du. \tag{3.5.6}$$

The π messages propagate forwards along the chain using the above rule. At each node in the chain (except the evidence nodes) the belief can be computed as a product of its λ and π messages, which in turn are computed recursively, and $\beta = p(e^-|e^+)^{-1}$.

3. **Causal tree**

Consider the belief of node x on the tree shown in Fig. 3.10.

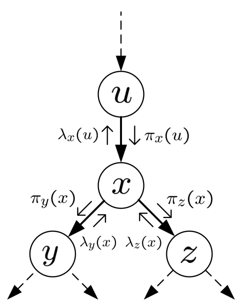


Figure 3.10: A causal tree - one parent per node; message directions are indicated.

$$b(x) \equiv p(x|\{e_x^-, e_x^+\}) = \beta\lambda(x)\pi(x), \tag{3.5.7}$$

where,

$$\lambda(x) \equiv p(e_x^-|x), \tag{3.5.8}$$

and,

$$\pi(x) \equiv p(x|e_x^+), \tag{3.5.9}$$

in which e_x^- denotes all observed data or evidence available in the sub-tree rooted at x, that is all evidence contained in its descendants (may include an instantiation of x itself, if necessary) and e_x^+ denotes all observed data or evidence in the rest of the network (in its ancestors, cousins).

It is sufficient to describe how each variable updates its belief from the messages it receives and the manner in which it generates messages for its descendants and its parent. Through such local computations alone (due to the d-separation criterion for Bayesian nets) at each variable, all the variables on the graph can correctly update their belief states in a time proportional to the diameter of the network. The two local steps are described below for the tree in this example.

Incoming messages to variable x - belief update:

$$p(e_x^-|x) = p(e_y^- \cup e_z^-|x) = p(e_y^-|x)p(e_z^-|x), \quad \text{(by d-separation)}. \tag{3.5.10}$$

Denote, $\lambda_y(x) \equiv p(e_y^-|x)$, the message sent from child y to parent x and with a similar connotation, $\lambda_z(x) \equiv p(e_z^-|x)$.

$$p(x|e_x^+) = \int p(x|u)p(u|e_x^+)du, \quad \text{(by d-separation)}. \tag{3.5.11}$$

Denote, $\pi_x(u) = p(u|e_x^+)$, the message sent from parent u to child x. From Eqns. 3.5.10,3.5.11 the belief of variable x can be updated as shown below.

$$b(x) = \beta p(e_x^-|x)p(x|e_x^+). \tag{3.5.12}$$

Outgoing messages from variable x - propagation in trees:

Consider,

$$\lambda_x(u) \equiv p(e_x^-|u) = \int p(e_x^-|x)p(x|u)du = \int \lambda(x)p(x|u)du. \tag{3.5.13}$$

The above rule prescribes the recursive computation (based on the message node x receives from its children) required to compute the message variable x sends to its parent u. Consider,

$$\pi_y(x) \equiv p(x|e_y^+) = \beta p(e_y^+ = \{e_x^+, e_z^-\}, x) = \beta p(e_z^-|x)p(x|e_x^+) = \beta\lambda_z(x)\pi(x). \tag{3.5.14}$$

The above rule prescribes the recursive computation (based on the message x receives from its other child z and its parent u) required to compute the message variable x sends to its child y. Now the method is complete. The Eqns. 3.5.10,3.5.11,3.5.13,3.5.14,3.5.12 enable all hidden variables in the network to update their beliefs in light of the evidence provided to the network.

The key point to observe here is that the computation of the messages $\lambda_y(x)$ and $\pi_y(x)$ are unaffected by each other. This ensures that the belief update of x does not enter a circular loop (this is due to the d-separation criterion). Hence all the belief states can be stably updated in one single pass (bottom up and top down) through the network.

As a note of interest, consider the HMM shown in Fig. 3.6. Each hidden variable has one parent and two children (one of them, the observed data, is instantiated). The belief of x_n can be computed using the message passing equations as follows.

$$b(x_n) \equiv p(x_n|\{e^-_{x_n}, e^+_{x_n}\}) = \beta\lambda(x_n)\pi(x_n) = \beta p(e^-_{x_n}|x)p(x_n|e^+_{x_n}), \qquad (3.5.15)$$

where,

$$\lambda(x_n) = p(y_n|x_n), \quad \text{(leaf node)}. \qquad (3.5.16)$$

$$\pi(x_n) = \int p(x_n|x_{n-1})\pi_{x_n}(x_{n-1})dx_{n-1}, \qquad (3.5.17)$$

where,

$$\pi_{x_n}(x_{n-1}) \equiv p(x_{n-1}|e^+_{x_n}) = p(x_{n-1}|e^+_{x_n} = \{y_{1:n-1}\}). \qquad (3.5.18)$$

Combining the above equations,

$$b(x_n) = \beta p(y_n|x_n) \int p(x_n|x_{n-1})p(x_{n-1}|y_{1:n-1})dx_{n-1}. \qquad (3.5.19)$$

This is the standard filtering equation, see Eqn. 3.4.5.

Next consider the task of evaluating the belief for the hidden variable x_n in the

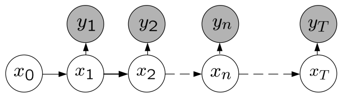


Figure 3.11: Smoothing on a HMM.

Bayesian network shown in Fig. 3.11. The incoming λ and π messages for x_n can be evaluated as follows.

$$\lambda(x_n) = p(y_n|x_n)\lambda_{x_{n+1}}(x_n) = p(y_n|x_n) \int \lambda(x_{n+1})p(x_{n+1}|x_n)dx_{n+1}. \qquad (3.5.20)$$

$$\pi(x_n) = p(x_n|e^+_{x_n}) = \int p(x_n|x_{n-1})\pi_{x_n}(x_{n-1})dx_{n-1}, \qquad (3.5.21)$$

where,

$$\pi_{x_n}(x_{n-1}) \equiv p(x_{n-1}|e^+_{x_n} = \{y_{1:n-1}\}). \qquad (3.5.22)$$

Combining the above equations,

$$b(x_n) = \beta p(y_n|x_n) \int \lambda(x_{n+1})p(x_{n+1}|x_n)dx_{n+1} \int p(x_n|x_{n-1})p(x_{n-1}|y_{1:n-1})dx_{n-1}. \qquad (3.5.23)$$

The above rule is the smoothing equation. When seen on a time axis, the belief of x_n is computed with past, present and future evidences taken into account. If the $\lambda_{y_n}(x_n) = p(y_n|x_n), \forall n = 1:T$ messages have a Gaussian form and the link probability distributions $p(x_n|x_{n-1}), \forall n = 1:T$ are also Gaussian in nature, then this belief update recursion is exactly the Rauch-Tung-Streibel (RTS) smoothing equation, also known as the Kalman smoother (see [Minka, 1999]).

4. **Causal polytrees**

In a polytree a variable has one or more parents and several children. An

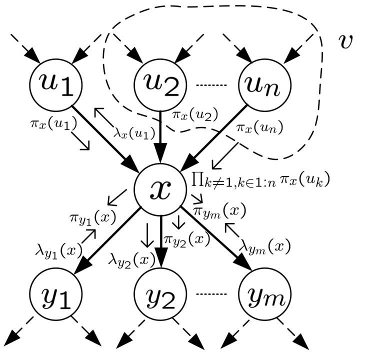


Figure 3.12: A causal polytree - several parent per node; example message directions are indicated.

illustrative example is shown in Fig. 3.12. Consider the belief of variable x on this graph.

$$b(x) \equiv p(x|e_x^-, e_x^+) = \beta p(e_x^-|x)p(x|e_x^+), \tag{3.5.24}$$

where, $e_x^+ = \{e_{xu_1}^+, \dots, e_{xu_n}^+\}$, is the collection of evidences contained in sub networks on the parent side of x and $e_x^- = \{e_{xy_1}^-, \dots, e_{xy_m}^-\}$ is the collection of evidences contained in the sub-network on the descendants side of x.

Incoming messages to variable x - belief update:

$$\lambda(x) \equiv p(e_x^-|x) = \prod_{i=1:m} p(e_{xy_i}^-|x) = \prod_{i=1:m} \lambda_{y_i}(x), \quad \text{(d-separation)}. \tag{3.5.25}$$

$$\pi(x) \equiv p(x|e_x^+) = \int p(x|u_{1,\dots,n}) \prod_{i=1:n} p(u_i|e_{xu_i}^+) du_{1,\dots,n}. \tag{3.5.26}$$

Denote $\pi_x(u_i) = p(u_i|e_{xu_i}^+)$, the message sent from parent u_i to x.

Outgoing messages from variable x - propagation in trees:

The message x sends to its parent u_i is computed as follows. Let $v = \{u_k\}_{k\neq i, k=1:n}$.

$$\lambda_x(u_i) \equiv p(e_{xv}^+, e_x^-|u_i), \tag{3.5.27}$$

where e^+_{xv} is the evidence available in the v subnetwork and e^-_x is the evidence available in the network rooted at x.

$$\begin{aligned}
\lambda_x(u_i) =& \beta \int p(e^+_{xv}, e^-_x, u_i, x) dx, \\
=& \beta \int p(e^-_x|x) p(e^+_{xv}, u_i, x) dx, \quad \text{(d-separation)}, \\
=& \beta \int \lambda(x) \int p(e^+_{xv}, u_i, v, x) dv dx, \\
=& \beta \int \lambda(x) \int p(e^+_{xv}|v) p(x, v|u_i) dv dx, \\
=& \beta \int \lambda(x) \int \frac{p(v|e^+_{xv})}{p(v)} p(x|v, u_i) p(v|u_i) dv dx, \\
=& \beta \int \lambda(x) \int p(v|e^+_{xv}) p(x|v, u_i) dv dx, \\
=& \beta \int \lambda(x) \int \prod_{j \neq i, j=1:n} \pi_x(u_j|e^+_{xu_j}) p(x|v, u_i) dv dx. \qquad (3.5.28)
\end{aligned}$$

The message x sends to the child y_i can be computed as follows.

$$\begin{aligned}
\pi_{y_i}(x) \equiv p(x|e^+_{xy_i}) =& \beta p(x, \left\{e^+_{xu_j}\right\}_{j=1:n}, \left\{e^-_{xy_j}\right\}_{j=1:m, j \neq i}), \\
=& \beta \prod_{j \neq i, j=1:m} p(e^-_{xy_j}|x) \pi(x), \\
=& \beta \prod_{j \neq i, j=1:m} \lambda_{y_j}(x) \pi(x). \qquad (3.5.29)
\end{aligned}$$

These equations are sufficient to update the beliefs of the hidden variables in the network.

5. **Loopy Bayesian networks - multiply connected networks**

Loopy graphs are multiply connected Bayesian networks that contain *undirected loops* (directed loops are disallowed in Bayesian networks). The message passing scheme described so far may not converge to a stable equilibrium as the d-separation rule on which it is based does not hold true (a parent and a child of a variable may be connected via another path through the network). An illustrative example is shown on the left part of Fig.3.13.

There are three different approaches to deal with loops. First, is the *clustering* method where variables are lumped together to form a compound variable and so converting the loop to a tree on which standard message passing can be used. See Fig. 3.13 for an illustration. This is reasonable only when the loops

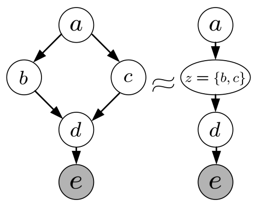


Figure 3.13: Reducing a loopy graph to a tree - the clustering approach.

are small and the dimension of the variables are low, otherwise the *re-evaluation* of the probability distributions of the links (potentials) would be difficult due to the compounding of high dimensional variables (with exponentially increasing cardinality). Things are more complicated if the variables are continuous. Another difficulty is that the lack of structure within the compound variable makes it difficult to explain the belief updates of its component variables.

The second approach is the *conditioning* approach, where a variable in the loop

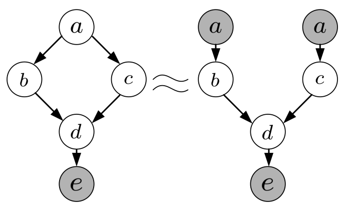


Figure 3.14: Reducing a loopy graph to a tree - the conditioning approach.

is instantiated such that the loop is converted to a singly connected network and then the beliefs of the non-instantiated variables are averaged over the posterior distribution of the instantiated variable. For the graph shown on the left part of Fig.3.14, if the variable a is instantiated, then the belief of variable b can be computed using the Chapman Kolmogorov rule as shown below.

$$p\big(b|e\big) = \int p\big(b|a, e\big)p\big(a|e\big)da. \tag{3.5.30}$$

The belief of a itself is computed as follows.

$$p\big(a|e\big) = \beta p\big(e|a\big)p\big(a\big), \tag{3.5.31}$$

which requires knowledge of the prior distribution of a and the ability to evaluate $p\big(e|a\big)$ (normally evaluated using message passing on a singly connected network between e and a). This method again suffers when the network is highly connected and the number of variables required to be instantiated to render a singly connected network is large.

The third approach is the *Gibbs sampling* approach (see Sec.3.4.2) to drawing samples from the posterior distribution over all the hidden variables in the network. The beliefs for *each state* of a hidden variables is then computed by counting the frequency with which that sample is drawn. An implicit assumption in this approach is that a sample of a hidden variable can be drawn from the distribution of the variable conditioned on its Markov blanket (see [Pearl, 1997]). This is usually difficult in general cases where the likelihood functions (λ messages) are difficult to sample from due to normalization problems (see Chapter 6 for such a case).

3.5.2 Markov networks

1. **Markov trees**

   Consider the Markov network shown in Fig. 3.15. The belief of variable x can be computed using a strategy similar to the one described for Bayesian networks with a few modifications (there is no notion of a parent and child in these networks).

   Belief update:

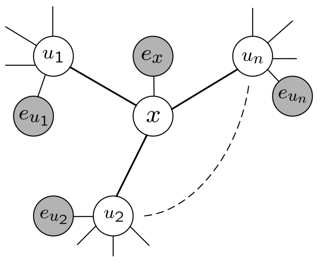


Figure 3.15: General Markov tree.

$$b(x) \equiv p(x|e) = \beta p(e_x|x)p(x|e_{M(x)}), \tag{3.5.32}$$

where e_x is the evidence directly connected to x, $e_{M(x)}$ is the evidence available in the Markov boundary of x (the Markov boundary is the minimum set of neighbours of x which vertex separates x from the rest of the network, in Fig. 3.15, $M(x) = \{u_1, \ldots, u_n\}$) and β is an appropriate normalizing constant to ensure the belief sums to unity. For comparison's sake, $\lambda(x) \equiv p(e_x|x)$ and $\pi(x) \equiv p(x|e_{M(x)})$.

$$\begin{aligned} p(x|e_{M(x)}) =& \beta \int p(x, M(x), e_{M(x)}) dM(x), \\ =& \beta \int p(e_{M(x)}|M(x)) \phi(x, M(x)) dM(x), \end{aligned} \tag{3.5.33}$$

where $\phi(x, M(x)) = \prod_{i \in 1 \ldots n} \phi_i(x, u_i)$ is a compatibility function over the cliques (here, pair-wise cliques) of the network. Now,

$$\begin{aligned} p(x|e_{M(x)}) =& \beta \int \prod_{i \epsilon M(x)} p(e_{M(u_i)/x} \cup e_{u_i}|u_i) \phi_i(x, u_i) dM(x), \quad \text{(By conditional independency)}, \\ =& \beta \prod_{i \epsilon M(x)} m_{u_i \to x}(x), \end{aligned} \tag{3.5.34}$$

where $m_{u_i \to x}(x) = \int p(e_{M(u_i)/x} \cup e_{u_i}|u_i) \phi_i(x, u_i) du_i$ is the message sent from u_i to x. With these messages the belief of x can be computed as shown below.

$$b(x) \equiv p(x|e) = \beta p(e_x|x) \prod_{i \epsilon M(x)} m_{u_i \to x}(x). \tag{3.5.35}$$

Message propagation:
The message x sends to its neighbour u_i can be computed recursively as shown below.

$$\begin{aligned} m_{x \to u_i}(u_i) \equiv& \beta \int p(e_{M(x)/u_i} \cup e_x|x) \phi_i(x, u_i) dx, \\ =& \beta \int p(e_x|x) \phi_i(x, u_i) \prod_{k \epsilon M(x)/u_i} m_{u_k \to x}(x) dx. \end{aligned} \tag{3.5.36}$$

These recursions are sufficient to propagate the effects of the observing new data throughout the network.

2. **Loopy Markov networks - multiply connected networks**
As for loopy Bayesian networks, only approximate belief propagation strategies exist for multiply connected Markov networks. To see why, consider the network shown in Fig. 3.16, the elements in the Markov boundary of x_1 are *not separated* by x_1 due to the other pathway $x_2 - x_3 - x_4$. But this is the assumption under which the recursive message propagation equations 3.5.36 were derived. Therefore, exact beliefs cannot be computed on such loopy graphs.

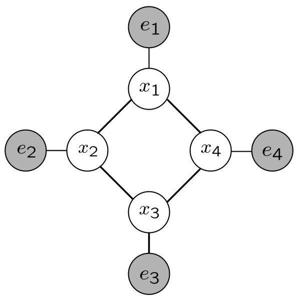


Figure 3.16: A loopy Markov network.

The general strategy to deal with loops in Markov networks is to perform belief updates and message propagation as described for the case of Markov trees. Theoretical convergence to the exact beliefs is not guaranteed, nevertheless the method has found wide spread use due to its empirical success. There have also been some attempts to explain its success by viewing loopy belief propagation as an energy minimization strategy (see [Yedidia et al., 2001]). In formal terms, the message propogation from variable x_1 to its neighbour x_2, at iteration t, in the loopy graph of Fig. 3.16 is made as follows.

$$m^t_{x_1 \to x_2}(x_2) \equiv \beta \int p(e_1|x_1)\phi_{1,2}(x_1, x_2) \prod_{k \epsilon M(x_1)/x_2} m^{t-1}_{x_k \to x_1}(x_1) dx_1, \qquad (3.5.37)$$

The above form of message update is used for all message computations. The belief update of variable x_2 at iteration t is now made as follows.

$$b(x_2) \equiv p^t(x_2|e = \{e_1, \ldots, e_4\}) = \beta p(e_{x_2}|x_2) \prod_{i \epsilon M(x_2)} m^t_{x_i \to x_2}(x_2). \qquad (3.5.38)$$

The above form of belief update is used for hidden variables. Note that these iterative updates are exactly the same as the message propagation and belief update equations given in Eqn. 3.5.35 and Eqn. 3.5.36 (with added time index for more clarity).

A note on message computation:
Although all the messages necessary for a belief update can be computed in an unnormalized form and their product normalized to render the normalized belief, it is generally advised to normalize the messages individually (that is, compute the β constants for each message) before the belief update step to avoid numerical overflow problems.

4

Probabilistic fusion of point trackers

4.1 Introduction

Tracking feature points is ubiquitous and its relevance cannot be over stated. On its own, it is a vital preprocessing step for applications such as, digital cinema post-production [Buchanan and Fitzgibbon, 2007], motion estimation [Sand and Teller, 2006] and people finding [Ramanan and Forsyth, 2003]. This approach has also established its place as a component of several recent tracking methods including, the "flock of KLT features" approach of [Kolsch and Turk, 2004; Tomasi and Kanade, 1991] for hand tracking, "learning features" approach of [Grabner et al., 2007] for tracking rigid objects, "conditional filters" for point tracking of [Arnaud et al., 2004], simultaneous localization and mapping (SLAM) algorithms [B. Williams and Reid, 2007] and "randomized features" as a cue within a multi-cue object tracking [Badrinarayanan et al., 2007b] method.

The strength of point tracking is apparent when tracking distinct feature points, like SIFT based features [Lowe, 2004] or corner points [Harris and Stephens, 1988], lying on objects displaying smooth and nearly planar motion. The resulting trajectory is smooth and can be used, for instance, to estimate object motion models or camera pose parameters. If additional care is taken [J.P.Lewis, 1995], then points can be tracked even under drastic changes in illumination or shadow effects visible on the object. However, weaknesses begin to appear in presence of non-planar motions, clutter, camera jerks or occlusions. Here the trajectory goes awry and the tracking process needs to be restarted. This is because point trackers implicitly assume the posterior distribution of the tracked point is uni-modal and the object is at a distance from the camera for which its motion can be assumed planar. All these are very restrictive assumptions and more often than not unrealistic. Attempts have been made recently to overcome some of these shorcomings by driving the "search" for feature points based on the predictions of the global motion of the object [Buchanan and Fitzgibbon, 2007]. In any case, point trackers are still useful during tracking over short time-spans and when several of them are made to work in a globally consistent fashion [Buchanan and Fitzgibbon, 2007; Badrinarayanan et al., 2007b].

Two fundamental differences exist between discriminative approaches like point tracking and generative approaches like Bayesian filtering. First is in the relationship between the observed data (image or collection of some of its parts) and the hidden state variable. In point tracking, true to other discriminative approaches, the observed data is used to directly predict the position of the point (the hidden state). In this sense there is a deterministic relationship established between the observed data and the hidden state. On the other hand, generative models in a Bayesian framework introduce a data likelihood which assigns a measure to the hidden variable. This measure quantifies how likely the hidden variable is to have generated the observed data. Therefore, there is a probabilistic relationship established between the observed data and the hidden state. The second difference lies in the way the state space is explored by these approaches. In point tracking methods, such as correlation based tracking [J.P.Lewis, 1995] , the exploration is regular image grid search, usually centered on the estimated position of the point at the previous instant. In contrast, several Bayesian methods employ random walks or other stochastic diffusion equations to explore the state space. These two characteristic differences are an asset and can be exploited advantageously within a framework in which these two approaches can operate together symbiotically while essentially retaining their characteristic qualities. Probabilistic graphical models provide a mechanism to model these ideas.

The contents of this chapter first present a probabilistic graphical model to view point tracking as a Monte Carlo (MC) simulation filter in Sec. 4.2. Then on, in Sec. 4.3 a graphical model is presented for measurement fusion via a "probabilistic message combination" technique. Based on a simplified switching version of this model, a useful tracking scheme termed *randomized feature point tracking* is constructed in Sec. 4.4. Some experiments are then setup in Sec. 4.5 to test this tracker. Results of tracking with this scheme are presented and the qualities of this tracker discussed in Sec. 4.6. Conclusions are drawn in Sec. 4.7.

4.2 Point tracking from a Bayesian perspective

In feature point tracking, a small patch of image around a chosen feature point, also called the point *image appearance template* (or interchangeably termed template), is tracked. At the arrival of new data (image) this template is matched with test templates extracted around grid points (samples) in a search space S. In practice, S is centered on the estimated location of the feature point at the previous instant. The matching involves computing the sum of square distance (SSD) or normalized cross-correlation (NCC) between the appearance template and the test template.The samples in S and their corresponding match weights together compose a *similarity or match* surface. The coordinates corresponding to the modal point of this surface is taken to be the estimated location of the feature point.

Emulation of these forms of point tracking within a stochastic filtering framework requires the specification of an appropriate process model for the hidden state, herein the position of the feature point, and an observation model describing the relation between the sequential data (images) and the hidden state. The process model facilitates the prediction of the state of the point based on its previous states. To specify such a model, it is informative to observe that given the image at instant n, the grid search space in which the new location of the feature point is sought is a deterministic function of the match surface at the instant $n-1$ (in general practice, search space S is centered on the coordinates of the modal point of this surface). This search is equivalent to a kinematic prediction. Therefore, this prediction of the hidden state is parameterized by the match surface or some statistic derived from it.

The observation model describes the relation between the image data (or a collection of its parts) and the hidden state. This relationship is in general described through a non-linear function between the image data and the hidden state; the matching process in the case of point tracking. Based on these guidelines, the graphical model in Fig. 4.1 is put forth for a stochastic description of point tracking.

In the graphical model shown in Fig. 4.1, where x_n denotes the hidden state at instant n (the location of the feature point at n), $y_{1:n}$ denotes the sequence of measurements associated to the sequence of hidden states $x_{1:n}$ and $q_{0:n-1}$ are parameters controlling the process model. From the model, the measurement process is independent when conditioned on the hidden states. Upon noting this and the form of the graphical model in Fig. 4.1, the filtering distribution can be expressed as shown below.

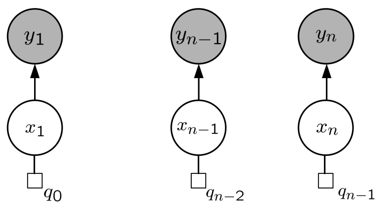


Figure 4.1: Disconnected graphical model for point tracking. Deterministic parameters are marked by small rectangles.

$$p\big(x_n|y_{1:n};\vartheta,q_{n-1}\big) = p\big(x_n|y_n;\vartheta,q_{n-1}\big) \propto l\big(y_n|x_n,\vartheta\big)p\big(x_n|q_{n-1}\big), \tag{4.2.1}$$

where ϑ is a parameter of the data likelihood $l(.)$, which, for instance, could be the *image appearance template* of a feature point.

In this model, the matching process between the image appearance model and test templates at samples of x_n can be subsumed in the functional definition of the data likelihood $l\big(y_n|x_n;\vartheta\big)$ as demonstrated below.

$$l\big(y_n|x_n;\vartheta\big) \triangleq \|\mathcal{T}\left(y_n,x_n\right) - \vartheta\|, \tag{4.2.2}$$

where the function $\mathcal{T}\left(y_n,x_n\right)$ depends on the *observation* (test templates) and $\|.\|$ is a chosen distance measure (SSD, NCC). Here, ϑ is the reference image appearance template for a point tracker against which the test templates are compared.

Approximating the filtering distributions:

Say, at instant $n-1$, the following *sample set* $\left\{x^i_{n-1},\widetilde{w}^i_{n-1}\right\}^M_{i=1}$ approximation of the filtering distribution is available.

$$p\big(x_{n-1}|y_{n-1};\vartheta,q_{n-2}\big) \approx \sum_{i=1}^{M} \widetilde{w}^i_{n-1}\delta_{x^i_{n-1}}\big(x_{n-1}\big). \tag{4.2.3}$$

From this approximation, the parameter q_{n-1} is updated as shown below.

$$q_{n-1} \approx \underset{x_{n-1}}{\operatorname{argmax}} \sum_{i=1}^{M} \widetilde{w}^i_{n-1}\delta_{x^i_{n-1}}\big(x_{n-1}\big). \tag{4.2.4}$$

Given this estimate of q_{n-1}, the filtering distribution at instant n can be approximated by aid of *importance sampling* [Arulampalam et al., 2002], with the prior $p\big(x_n|q_{n-1}\big)$

acting as the importance or proposal distribution. Samples $\{x_n^i \sim p(x_n|q_{n-1})\}_{i=0}^{M}$ are drawn and associated with an *unnormalized* importance weight as indicated below.

$$w_n^i = l(y_n|x_n^i, \vartheta),\ i = 1 \dots M. \tag{4.2.5}$$

In accordance with the principle of importance sampling, normalizing these importance weights delivers the required approximation of the filtering distribution:

$$p(x_n|y_n; \vartheta, q_{n-1}) \approx \sum_{i=1}^{M} \widetilde{w}_n^i \delta_{x_n^i}(x_n). \tag{4.2.6}$$

These steps are repeated at each new filtering instant.

The proposal density

The following proposal density is designed with a view to imitate regular grid search used in point tracking.

$$p(x_n|q_{n-1}) = \mathcal{U}_S(x_n; \mu = q_{n-1}), \tag{4.2.7}$$

where $\mathcal{U}_S(x_n; \mu = q_{n-1})$ is a uniform distribution in x_n, with mean μ and support S. If sufficient non-repetitive samples of x_n are drawn from this distribution, then *regular grid search* inside the search space S can be *mimicked*. As a note of interest here, the definition of the prior in Eqn. 4.2.7 is only one among several ways in which the parameter q_{n-1} could be used to imitate point tracking. Other novel ways to utilize such parametrization could indeed lead to various other discriminative/deterministic tracking schemes.

Two points are noteworthy in this filtering by simulation representation of point tracking. First is the fact that any arbitrarily shaped filtering distribution can be propagated over time, true to a sequential Monte Carlo approach and second is that filtering at each time instant can be made *iterative*. The approximated filtering distribution at the end of an iteration can parameterize the filtering distribution at the following iteration. This brings it closer in spirit to iterative optimization schemes such as in the deterministic tracker of [Comaniciu and Meer, 1999], [Tomasi and Kanade, 1991], wherein the sampling of the search space is in the direction of the steepest gradient. Further work is necessary in this direction to establish this proposition. Finally, the appendix at the end of this chapter describes another Bayesian viewpoint of point tracking based on the concept of Assumed density filtering (ADF) [Boyen and Koller, 1998].

If the problem is to track the center of an object, then tracking a single feature point on the object (excluding the center itself) consistently would be sufficient if the spatial relationship between the feature point and the center remains constant while the object is in motion. But such assumptions are easily violated in practice. Instead, if several points on the object are tracked independently and their estimates of the center are consistently fused, then a meaningful estimate with high certainty can be obtained when a majority of the points are tracked correctly. The next section paves the way for modeling such an idea borrowing from the concept presented so far.

4.3 Graphical model for tracking with a set of point trackers

4.3.1 Static model

Consider the following joint model corresponding to the graph shown in Fig. 4.2.

$$p(x_0, x_1, x_2|y_1, y_2; \theta) \propto l_1(y_1|x_1)l_2(y_2|x_2)p(x_0, x_1, x_2|\theta), \tag{4.3.1}$$

where, x_0 is the hidden "fused state", x_1, x_2 represent hidden states with corresponding measurements (data) y_1, y_2. l_1 and l_2 define the likelihood functions (no observation is associated with x_0), whose partition functions are difficult to compute analytically. x_0 is related to the other hidden states by the *prior* defined below.

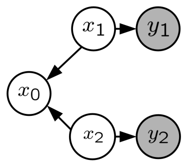


Figure 4.2: Graphical model for message combination in the static case with 3 hidden variables.

$$p(x_0, x_1, x_2|\theta) \triangleq \theta g(x_0|x_1)p(x_1)p(x_2) + [1-\theta]h(x_0|x_2)p(x_2)p(x_1); \tag{4.3.2}$$

where, *parameter* $\theta \in [0, 1]$ and $g(x_0|x_1) \neq p(x_0|x_1)$, $h(x_0|x_2) \neq p(x_0|x_2)$ are two instrumental conditional distributions. With this definition, the posterior distribution of x_0 is obtained by marginalisation of the joint law in Eqn. 4.3.1:

$$p(x_0|y_1, y_2; \theta) = \theta\frac{\int l_1(y_1|x_1)g(x_0|x_1)p(x_1)dx_1}{\int l_1(y_1|x_1)p(x_1)dx_1} + [1-\theta]\frac{\int l_2(y_2|x_2)h(x_0|x_2)p(x_2)dx_2}{\int l_2(y_2|x_2)p(x_2)dx_2}.$$

Analogous to belief propagation terminology [Yedidia et al., 2001; Sudderth et al., 2003], *messages* $m_{1\to0}, m_{2\to0}$ are defined and used as follows.

$$\begin{aligned} m_{1\to0}(x_0) &= \frac{\int l_1(y_1|x_1)g(x_0|x_1)p(x_1)dx_1}{\int l_1(y_1|x_1)p(x_1)dx_1}, \\ m_{2\to0}(x_0) &= \frac{\int l_2(y_2|x_2)h(x_0|x_2)p(x_2)dx_2}{\int l_2(y_2|x_2)p(x_2)dx_2}, \\ p(x_0|y_1,y_2;\theta) &\propto \theta m_{1\to0}(x_0) + [1-\theta]m_{2\to0}(x_0). \end{aligned} \tag{4.3.3}$$

The *message switching* or *linear weighted message combination* attribute is apparent from the above equation. For instance, if θ is a binary variable taking value 1, then the posterior of x_0 is influenced only by $m_{1\to0}(x_0)$. On the other hand, if θ takes on real values in the range $[0,1]$, then the posterior is composed by a linear combination of messages. Also, note that the messages are normalized to sum to unity.
It is also informative to observe the posterior distribution of x_1:

$$\begin{aligned} p(x_1|y_1,y_2;\theta) = \theta \int l_1(y_1|x_1)g(x_0|x_1)p(x_1)\int l_2(y_2|x_2)p(x_2)dx_0dx_2 + \\ [1-\theta]\int l_2(y_2|x_2)h(x_0|x_2)p(x_2)\int l_1(y_1|x_1)p(x_1)dx_0dx_2, \end{aligned} \tag{4.3.4}$$

which reduces to;

$$p(x_1|y_1,y_2) \equiv p(x_1|y_1) \propto l_1(y_1|x_1)p(x_1). \tag{4.3.5}$$

This posterior is not controlled by the parameter θ and the same holds true for state x_2 too. Similar results emerge for the sequential case too, as discussed below.

4.3.2 Sequential model

Consider the following model corresponding to the Dynamic Bayesian Network (DBN) shown in Fig. 4.3.

$$p(X_n|Y_{1:n};\theta_n) \propto l(Y_n|X_n)p(X_n|Y_{1:n-1};\theta_n), \tag{4.3.6}$$

where, $X_n = \{x_n^0, x_n^1, x_n^2\}$, $Y_{1:n} = \{y_{1:n}^1, y_{1:n}^2\}$. The first order Markovian transition distribution (or joint process model) is defined below.

$$p(X_n|X_{n-1};\theta_n) \triangleq \Big[\theta_n g(x_n^0|x_n^1) + [1-\theta_n]h(x_n^0|x_n^2)\Big]p(x_n^1|x_{n-1}^1)p(x_n^2|x_{n-1}^2), \tag{4.3.7}$$

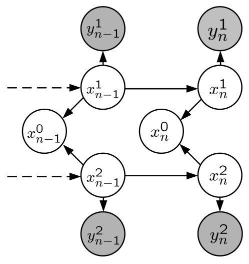


Figure 4.3: Graphical model for message combination in the sequential case with 3 hidden variables.

with parameter $\theta_n \in (0,1)$ and instrumental distributions, $g\big(x_n^0|x_n^1\big), h\big(x_n^0|x_n^2\big)$. The filtering distribution of x_n^0 can then be evaluated through the following steps.

$$p\big(x_n^0|Y_{1:n};\theta_n\big) \propto \int \Big[\theta_n g\big(x_n^0|x_n^1\big) + \big[1-\theta_n\big] h\big(x_n^0|x_n^2\big)\Big] l_1\big(y_n^1|x_n^1\big) l_2\big(y_n^2|x_n^2\big) \times \\ p\big(x_n^1|x_{n-1}^1\big) p\big(x_n^2|x_{n-1}^2\big) p\big(X_{n-1}|Y_{1:n-1};\theta_{n-1}\big) dx_n^1 dx_n^2 dX_{n-1}. \quad (4.3.8)$$

Tackling temporal loops

The variables x_n^1, x_n^2 are intricately connected in the above integral via the term $p\big(X_{n-1}|Y_{1:n-1};\theta_{n-1}\big)$. In general, it is difficult to achieve the goal of expressing the above integral as a linear combination of messages like in Eqn. 4.3.3. However, the integral is indeed simplified under the special case of filtering by simulation model of point tracking (see Eqn. 4.2.1 in Sec. 4.2). When this special case is invoked the model

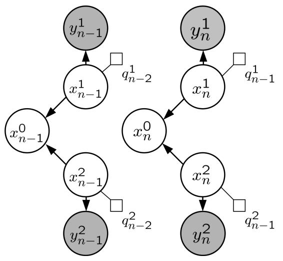


Figure 4.4: Simplified graphical model for message combination in the sequential case with 3 hidden variables.

simplifies to the form shown in Fig. 4.4. It is brought to the notice of the reader here that such model simplifications will be used frequently in the forthcoming discussions. From the reduced model the following equations can be written down.

$$p\big(x_n^0|Y_n;q_{n-1}^1,q_{n-1}^2,\theta_n\big) = \theta_n m_{1\to 0}\big(x_n^0|q_{n-1}^1\big) + \big[1-\theta_n\big] m_{2\to 0}\big(x_n^0|q_{n-1}^2\big), \quad (4.3.9)$$

where,

$$m_{1\to 0}\big(x_n^0|q_{n-1}^1\big) = \frac{\int l_1\big(y_n^1|x_n^1\big)p\big(x_n^1|q_{n-1}^1\big)g\big(x_n^0|x_n^1\big)dx_n^1}{\int l_1\big(y_n^1|x_n^1\big)p\big(x_n^1|q_{n-1}^1\big)dx_n^1}, \tag{4.3.10a}$$

$$m_{2\to 0}\big(x_n^0|q_{n-1}^2\big) = \frac{\int l_2\big(y_n^2|x_n^2\big)p\big(x_n^2|q_{n-1}^2\big)h\big(x_n^0|x_n^2\big)dx_n^2}{\int l_2\big(y_n^2|x_n^2\big)p\big(x_n^2|q_{n-1}^2\big)dx_n^2}, \tag{4.3.10b}$$

and q_{n-1}^1, q_{n-1}^2 are parameters defined as in Eqn. 4.2.4.Proceeding in the same lines as the preceding demonstration, the filtering distribution of say, x_n^1, is as shown below.

$$p\big(x_n^1|y_n^1;q_{n-1}^1\big) \propto l_1\big(y_n^1|x_n^1\big)p\big(x_n^1|q_{n-1}^1\big). \tag{4.3.11}$$

Once again, as in the static case, the prior is so arranged that the filtering distributions of variables x_n^1, x_n^2 are not influenced by the parameter θ_n. This is an important point and proves useful in the update of θ_n.

4.3.3 Parameter update

The remaining issue in computing the filtering distribution of x_n^0 is the parameter update. In convenient situations, where the joint state posterior can be computed analytically, the Maximum Likelihood Estimate (MLE) or Maximum a Posteriori (MAP) estimate of θ_n can be obtained. However such closed form computations generally involve intractable integrations. Instead, the Iterative Conditional Estimate (ICE) technique of [Salzenstein and Pieczynski, 1995] proposes the following iterative method for parameter update.

$$\theta_n = \mathbf{E}_{p\left(X_n|Y_n;\theta_{n-1}\right)}\Theta\big(X_n, Y_n\big). \tag{4.3.12}$$

The statistical estimator $\Theta\big(.\big)$ could in an analytically convenient case be a MLE or MAP estimator. In more difficult cases, a choice based on empirical experience needs to be made. One such example, where the statistical estimate is *binary*, is given below.

$$\Theta\big(X_n, Y_n\big) = \begin{cases} 1, \text{if, } \frac{C_1\left[p\left(x_n^1|Y_n;q_{n-1}^1\right)\right]}{C_2\left[p\left(x_n^2|Y_n;q_{n-1}^2\right)\right]} \leq 1 \\ 0, \text{otherwise,} \end{cases} \tag{4.3.13}$$

where C_1, C_2 are some *scalar* measures of certainty or *figures of merit*, for instance, determinants of covariance matrices. The filtering distributions for hidden states x_n^1, x_n^2 can themselves be approximated independent of θ_n (see Eqn. 4.3.11). As such, other verification measures to determine inconsistency like the ones suggested in Hua et al [Hua et al., 2006] can also be configured to deliver statistical estimates resembling the one shown above.

To put all these ideas in context, a motivational example of sequential state estimation based on the sequential message switching model is presented below. Two hypothetical point trackers are employed for demonstration. The measurement models are assumed non-linear and priors are assume to be non-Gaussian, thus invoking *importance sampling* for approximations of filtering distributions [Arulampalam et al., 2002].

4.3.4 Switching two point trackers

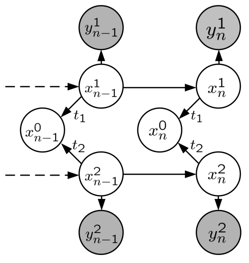


Figure 4.5: Graphical model for switching two point trackers. Parameters controlling the links are denoted.

Problem: Given two distinct point trackers on an object and their prior relationship with the position of the centre of the object (x^0_n), estimate sequentially the position of centre of the object by tracking these two points (see Fig. 4.5.
With reference to Fig. 4.5, the filtering model for the joint state, with explicit parameterization, is shown below.

$$p\big(X_n|Y_{1:n};\theta_n,\mathbf{t}_1,\mathbf{t}_2,\vartheta_1,\vartheta_2\big) \propto l\big(Y_n|X_n;\vartheta_1,\vartheta_2\big)p\big(X_n|Y_{1:n-1};\theta_n,\mathbf{t}_1,\mathbf{t}_2\big), \qquad (4.3.14)$$

where the notations for the state and measurements remain the same as before. $\mathbf{t}_1,\mathbf{t}_2$ parameterize the prior relationship between x^0_n, x^1_n and x^0_n, x^2_n respectively. ϑ_1,ϑ_2 are the image appearance templates for the point trackers.
Once again, to overcome the intractable inference issue associated with temporal loops, the model is simplified using the strategy presented earlier for the sequential case (see Sec. 4.2 and Fig. 4.4); whereupon, the filtering distribution of the hidden "fused state"

x_n^0 is given as follows.

$$p\big(x_n^0|Y_n;\theta_n,q_{n-1}^1,q_{n-1}^2,\mathbf{t}_1,\mathbf{t}_2,\vartheta_1,\vartheta_2\big)=\theta_n m_{1\to 0}\big(x_n^0|q_{n-1}^1,\mathbf{t}_1,\vartheta_1\big)+ \\ \big[1-\theta_n\big]m_{2\to 0}\big(x_n^0|q_{n-1}^2,\mathbf{t}_2,\vartheta_2\big), \tag{4.3.15}$$

where,

$$m_{1\to 0}\big(x_n^0|q_{n-1}^1,\mathbf{t}_1,\vartheta_1\big)=\frac{\int l_1\big(y_n^1|x_n^1;\vartheta_1\big)p\big(x_n^1|q_{n-1}^1\big)g\big(x_n^0|x_n^1;\mathbf{t}_1\big)dx_n^1}{\int l_1\big(y_n^1|x_n^1;\vartheta_1\big)p\big(x_n^1|q_{n-1}^1\big)dx_n^1}, \tag{4.3.16a}$$

$$m_{2\to 0}\big(x_n^0|q_{n-1}^2,\mathbf{t}_2,\vartheta_2\big)=\frac{\int l_2\big(y_n^2|x_n^2;\vartheta_2\big)p\big(x_n^2|q_{n-1}^2\big)h\big(x_n^0|x_n^2;\mathbf{t}_2\big)dx_n^2}{\int l_2\big(y_n^2|x_n^2;\vartheta_2\big)p\big(x_n^2|q_{n-1}^2\big)dx_n^2}, \tag{4.3.16b}$$

and, q_{n-1}^1, q_{n-1}^2 are parameters defined as in Eqn. 4.2.4. Let,

$$g\big(x_n^0|x_n^1;\mathbf{t}_1\big)=\delta_{x_n^1+\mathbf{t}_1}\big(x_n^0\big),\, h\big(x_n^0|x_n^2;\mathbf{t}_2\big)=\delta_{x_n^2+\mathbf{t}_2}\big(x_n^0\big), \tag{4.3.17}$$

and

$$p\big(x_n^1|q_{n-1}^1\big)\triangleq\mathcal{U}_S\big(x_n^1;\mu_1=q_{n-1}^1\big),\; p\big(x_n^2|q_{n-1}^2\big)\triangleq\mathcal{U}_S\big(x_n^2;\mu_2=q_{n-1}^2\big); \tag{4.3.18}$$

with the above densities being interpreted as in Eqn. 4.2.7.

Sampled based approximation of the messages

The representative example of $m_{1\to 0}\big(x_n^0|q_{n-1}^1,\mathbf{t}_1,\vartheta_1\big)$ is considered here. On drawing samples $x_n^{1,i}\sim p\big(x_n^1|q_{n-1}^1\big), i\in 1\ldots N$, this message can be approximated as shown below.

$$m_{1\to 0}\big(x_n^0|q_{n-1}^1,\mathbf{t}_1,\vartheta_1\big)\approx\frac{\sum_{i\in 1\ldots N} l_1\big(y_n^1|x_n^{1,i};\vartheta_1\big)\delta_{x_n^{1,i}+\mathbf{t}_1}\big(x_n^0\big)}{\sum_{i\in 1\ldots N} l_1\big(y_n^1|x_n^{1,i};\vartheta_1\big)}. \tag{4.3.19}$$

The other message can be approximated with a similar strategy.
Now, the filtering distribution of x_n^0 can be written down using a sample based approximation as follows.

$$p\big(x_n^0|Y_n;\theta_n,q_{n-1}^1,q_{n-1}^2,\mathbf{t}_1,\mathbf{t}_2,\vartheta_1,\vartheta_2\big)=\theta_n\frac{\sum_{i\in 1\ldots N} l_1\big(y_n^1|x_n^{1,i};\vartheta_1\big)\delta_{x_n^{1,i}+\mathbf{t}_1}\big(x_n^0\big)}{\sum_{i\in 1\ldots N} l_1\big(y_n^1|x_n^{1,i};\vartheta_1\big)}+ \\ \big[1-\theta_n\big]\frac{\sum_{i\in 1\ldots N} l_2\big(y_n^2|x_n^{2,i};\vartheta_2\big)\delta_{x_n^{2,i}+\mathbf{t}_2}\big(x_n^0\big)}{\sum_{i\in 1\ldots N} l_2\big(y_n^2|x_n^{2,i};\vartheta_2\big)}. \tag{4.3.20}$$

As gathered from Eqn. 4.3.11, the filtering distributions of x_n^1 and x_n^2 are independent from θ_n. Therefore, their filtering distributions can be approximated with sample sets

using importance sampling approximations of the form presented in Section. 4.2 and this subsequently is used to determine the state of θ_n. If $\theta_n = 1$ the posterior of x_n^0 is composed by the approximation of $m_{1\to 0}\big(x_n^0|q_{n-1}^1, \mathbf{t}_1, \vartheta_1\big)$ and vice-versa. Notice that the approximations of the messages are already normalized to sum to unity.
Extending the discussions so far, a practical object tracking filter, henceforth termed the randomized feature point tracker, is developed in the following section.

4.4 Randomized feature point tracker: RFT-filter

Based on the switching model presented in the previous section, the idea here is to construct an object position tracker with a set of randomly selected feature points (equivalently their image templates) on the object. Hence the name "randomized feature point tracker". In this tracker, the set of feature point templates captures the appearance of the object. While conforming to the graphical model the tracker is given the ability to replace a subset of the point trackers on the fly. The key strength of this tracker then lies in its ability to "adapt" the appearance model of the object itself via online point tracker replacement. The method proposes two intuitive and practical ways to replace outlier point trackers by sampling from the set of available posterior distributions. Each sampled point is then associated with an image appearance template to obtain a new point tracker. Finally, experiments on real data highlight the qualities of this tracker.
It is pointed out here that the idea of tracking with a set of point trackers is not entirely new. A close attempt which uses a cluster of detected corner points for tracking is described in the work by [Reid and Murray, 1996]. Their method is proposed in the context of active vision wherein a bunch of corner points are tracked to fixate a head/eye system onto a particular point in the scene. In their process, unreliable corner points are removed and, if available, new ones are included. Such replacement is also the essence of the feature learning technique proposed by [Grabner et al., 2007]. [Kolsch and Turk, 2004] employ a flock of features and feature replacement within a deterministic method. [Arnaud et al., 2004] use a stochastic filtering model for point tracking with global guidance from optic flow computations. This work is somewhat close in spirit to the stochastic simulation model for point tracking presented here. The difference with their work is that their focus is on developing new methodologies for point tracking itself.
Apart from the obvious stochastic nature of "picking up" point trackers online, what

is indeed different from all these methods is the probabilistic fusion idea on which the randomized feature tracker is based. This tracker is a stochastic filter propagating arbitrary distributions. The result of tracking at each instant is a distribution, not just a point estimate as in most of these other methods. This fact will later facilitate integration of discriminative point trackers with other stochastic filters, such as particle filters, for robust tracking (See chapter 5).

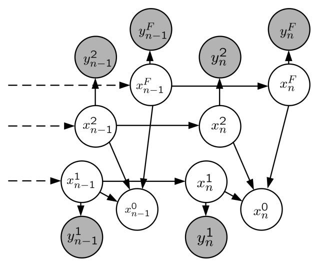


Figure 4.6: Graphical model for tracking with a set of point trackers

The graphical model underlying this tracker is shown in Fig. 4.6. At instant n, the hidden state representing the position of the object is x^0_n, while $\left\{x^f_n, y^f_n, f = 1 : F\right\}$ represent the hidden states and associated measurements of the point trackers. Let $X_n = \left\{x^0_n, x^1_n, \ldots, x^F_n\right\}$ denote the joint hidden state and $Y_n = \left\{y^1_n, \ldots, y^F_n\right\}$ the collection of measurements at instant n. Notice that there is no measurement associated with the hidden state x^0_n itself. The filtering distribution of the joint hidden state is factorised as shown below.

$$p\big(X_n|Y_{1:n}; \Phi_n, \mathcal{T}\big) \propto l\big(Y_n|X_n\big) \int \Psi\big(X_n|X_{n-1}; \Phi_n, \mathcal{T}\big) p\big(X_{n-1}|Y_{1:n-1}; \Phi_{n-1}, \mathcal{T}\big) dX_{n-1}, \tag{4.4.1}$$

where parameter $\Phi_n = \left\{\alpha^f_n, f = 1 : F\right\}$ and α^f_n are binary in nature. The elements of Φ_n will now play the role of θ_n from the preceeding discussions (See section 4.3.2). These parameters control the switching or combination of messages to compose the filtering distribution of the hidden state x^1_n. $l\big(Y_n|X_n\big)$ represents the likelihood function whose partition function is difficult to compute analytically. $\Psi\,(.)$ shown below is a *compatibility function* (unnormalized) between the hidden variables in the model and is parameterized by $\mathcal{T}$.

$$\Psi\big(X_n|X_{n-1}; \Phi_n, \mathcal{T}\big) \triangleq \sum_{f=1}^{F} \alpha^f_n \Psi_{0,f}\big(x^0_n|x^f_n; \mathcal{T}\big) p\big(x^f_n|x^f_{n-1}\big) \prod_{j=1, j\neq f}^{F} p\big(x^j_n|x^j_{n-1}\big). \tag{4.4.2}$$

This compatibility function describes the unnormalized relationship (in the probability distribution sense) between the hidden state variables. The above form is somewhat reminiscent of Eqns. 4.3.2, 4.3.7. The key difference is that $\sum_{f \in 1:F} \alpha_n^f \neq 1$. The idea behind this form of compatibility is fairly obvious, it is so chosen to enable the filtering distribution of x_n^0 to be expressed as a linear combination of messages, as in Eqn. 4.3.9 . The product term $\prod_{j=1, j \neq f}^{F} p(x_n^j | x_{n-1}^j)$ is strictly not necessary, it is inserted only as an adjustment factor to arrive at an elegant form of message combination. More discussions on the compatibility function will follow shortly. For now, the filtering distribution of the "fused" hidden state x_n^0 can be written down as follows.

$$p(x_n^0 | Y_{1:n}; \Phi_n, \mathcal{T}) \propto \sum_{f=1}^{F} \alpha_n^f \int l_f(y_n^f | x_n^f) \Psi_{0,f}(x_n^0 | x_n^f; \mathcal{T}) p(x_n^f | x_{n-1}^f) \times \\ \int \prod_{j=1, j \neq f}^{F} l_j(y_n^j | x_n^j) p(x_n^j | x_{n-1}^j) p(X_{n-1} | Y_{1:n-1}; \Phi_{n-1}, \mathcal{T}) dX_{n-1} dx_n^{1,\dots,F}. \tag{4.4.3}$$

Upon simplifying the (loopy) graphical model for tractable inference (see Sec. 4.2 and Fig. 4.4), the above filtering distribution can be succintly written using message notation as follows.

$$p\left(x_n^0 | Y_n; \Phi_n, \mathcal{T}, \left\{q_{n-1}^f\right\}_{f=1:F}\right) \propto \sum_{f=1}^{F} \alpha_n^f m_{f \to 0}(x_n^0 | q_{n-1}^f), \tag{4.4.4}$$

with,

$$m_{f \to 0}(x_n^0 | q_{n-1}^f) = \frac{\int l_f(y_n^f | x_n^f) \Psi_{0,f}(x_n^0 | x_n^f; \mathcal{T}) p(x_n^f | q_{n-1}^f) dx_n^f}{\int l_f(y_n^f | x_n^f) p(x_n^f | q_{n-1}^f) dx_n^f}, f = 1 : F, \tag{4.4.5}$$

where q_{n-1}^1, q_{n-1}^2 are parameters defined as in Eqn. 4.2.4. As may be verified, the filtering distribution of the hidden states representing the position of the tracked feature points is as given below.

$$p(x_n^f | y_n^f; q_{n-1}^f) \propto l_f(y_n^f | x_n^f) p(x_n^f | q_{n-1}^f), \ f = 1 : F. \tag{4.4.6}$$

It is to be noted that the above filtering distributions are not dependent on the parameter Φ_n. For the present, a few words about the compatibility function are in order.

4.4.1 The elements of the compatibility function

The compatibility function is used to inject all prior knowledge regarding the hidden states into the graphical model *per se.* Since this function reflects the designers empirical knowledge about the relationships between the variables it can be so composed

that further computations (after observations are available) can be made conveniently. For instance, the form of the compatibility function chosen in Eqn. 4.4.2 results in the composition of the filtering distribution of x_n^0 by a *linear combination of messages* from the point trackers. Therefore, the message from each tracker can be computed independently (and sequentially) of the others.
The elements of the compatibility function, $\Psi_{0,f}\big(x_n^0|x_n^f;\mathcal{T}\big), f = 1:F$, describe the relation between the hidden state of the object and the hidden states of the point trackers. An example where the relation is *linear* and deterministic is shown below.

$$\Psi_{0,f}\big(x_n^0|x_n^f;\mathcal{T}\big) = \delta_{x_n^f+\mathbf{t}_f}\big(x_n^0\big),\ f = 1:F, \tag{4.4.7}$$

where δ is the Dirac delta and parameter $\mathcal{T} = \mathbf{t}_f, f = 1:F$. Other functions describing affine or higher order relationships can also be used if deemed necessary. Inserting Eqn. 4.4.7 into Eqn. 4.4.3, and using the sample based approximation of messages demonstrated in Sec. 4.3.4, results in the following convenient approximation.

$$p\left(x_n^0|Y_n;\Phi_n,\mathcal{T},\left\{q_{n-1}^f\right\}_{f=1:F}\right) \propto \sum_{f=1}^{F}\alpha_n^f\frac{\sum_{i=1:N} l_f\big(y_n^f|x_n^{f,i}\big)\delta_{x_n^{f,i}+\mathbf{t}_f}\big(x_n^0\big)}{\sum_{i=1:N} l_1\big(y_n^f|x_n^{f,i}\big)} \tag{4.4.8}$$

In this case of delta compatibilities, the form of the messages is compliant with standard normalized cross correlation (NCC) point tracking. Each of the message approximations in practice is equivalent to computing the NCC surface of the corresponding point tracker and normalizing it sum to unity. Finally, the filtering distribution in Eqn. 4.4.8 is obtained by summing these normalized surfaces and renormalizing the result appropriately.

4.4.2 Parameter update

For further consideration, assume the filtering distributions of $x_n^f, f = 1:F$ are approximated by particle-sets $\left\{x_n^{f,j},\pi_n^{f,j}\right\}_{j=1}^{j=K}, f = 1:F$ respectively.

$\alpha_n^f,\ f = 1:F$ **update:**
As the discussions of this section describe a switching method these parameters are assumed to be *binary* in nature. Therefore, they determine whether measurements associated with a point tracker need to be considered or *discarded* at any instant. Hence a robust and possibly inexpensive method must be supplied to determine this fact. Note that as the parameters are not part of the graphical model any meaningful external

method or tool can be used for their updates. One such method based on motion clustering is described below.
Between instants $n-1$ to n, let the *translation vector* of each tracked point be computed, say as a difference of the MAP estimates of their filtering distributions at these instants (See Eqn. 4.4.6). Let the collection of pairs of hidden states and their corresponding translation vectors be denoted as $\{x^f, v^f\}_{f=1}^{F}$ (the time index n is dropped for brevity). See the illustration in Fig. 4.7.

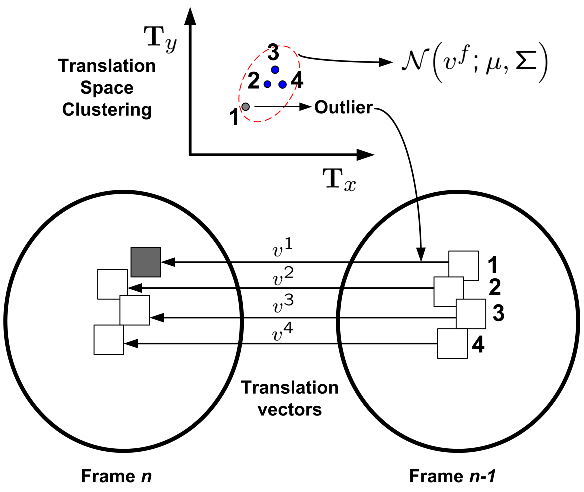


Figure 4.7: An illustration of clustering based outlier determination. The grey filled box represents the feature point being signalled as an outlier.

Probabilistic rejection of point trackers:

1. *Clustering*:
   Consider the augmented set $\left\{x^f, v^f, B^f\right\}_{f=1}^{F}$, where B^f is the bin count for the vector v^f. The bin count is initially set to 1 for all the point trackers.
   Each motion vector in the set $\left\{v^f\right\}_{f=1}^{F}$ is compared with the remaining vectors using an Euclidean distance measure and its bin-count is incremented for every vector that lies within a small predefined clustering radius r of this vector.
   A subset $\left\{v^f, f \in I, |I| \leq F\right\}$ of the set $\left\{v^f, f = 1 : F\right\}$ is formed by selecting all the motion vectors with associated bin-count $B^f \geq \frac{F}{2}$. If F is odd, then the relation $B^f \geq \frac{F+1}{2}$ is used instead.

2. *Rejection Control*:
   The two-dimensional mean of the subset $\left\{v^f, f \in I, |I| \leq F\right\}$ and the resulting covariance matrix, denoted as $\{\mu, \Sigma\}$ are computed. For the sake of simplicity, the cross variance terms are assumed zero.
   Let $\mathcal{N}\left(v^f; \mu, \Sigma\right)$ denote a Gaussian distribution in variable v^f with mean parameter μ and variance parameter Σ. A weight $w^f = \mathcal{N}\left(v^f; \mu, \Sigma\right)$ is assigned to each motion vector in the set $\left\{v^f, f = 1 : F\right\}$. The set $\left\{x^f, w^f\right\}_{f=1}^{F}$ is formed. The weights in the set are sorted in descending order and the weight of the median element is denoted as w_{med}. Each point tracker is then accepted with a probability,
   $$p^f = \min\left\{1.0, \frac{w^f}{w_{med}}\right\}, \; f = 1 : F. \tag{4.4.9}$$
   Following this,
   $$\alpha_n^f = \begin{cases} 1, \text{if } p^f \geq \rho \\ 0, \text{if } p^f < \rho, \end{cases} \tag{4.4.10}$$
   where ρ is a predefined threshold.

Once the parameters have been updated, Eqn. 4.4.3 can be used to compute the filtering distribution of x_n^0. The next objective is to replace all point trackers which have been signalled as inconsistent during parameter update. A procedure for replacement is described below.

4.4.3 Replacing point trackers

The new point trackers which would substitute the discarded ones are drawn from the posterior distribution of the hidden state of the object Eqn. 4.4.8.

Sampling the filtering distribution of the fused hidden state

Draw a true sample from the posterior for the case indicated below.

$$s^f \sim p\left(x_n^0 | Y_n; \Phi_n, \mathcal{T}, \left\{q_{n-1}^f\right\}_{f=1:F}\right), \text{ if } \alpha_n^f = 0. \tag{4.4.11}$$

These samples represent the locations of the new feature points.
An important consideration here is the effect of this replacement on the graphical model. It is argued that a feature point may be replaced without breaking and replacing links in the original graphical structure. Referring to Eqn. 4.2.7, replacing a feature

point implies an update of the external parameter which controls the prior. Doing so adjusts the search space for the newly sampled point tracker. Similarly, associating an appearance template to the newly sampled point tracker is an update of parameter ϑ (See Eqn. 4.2.1). Therefore, new point trackers may replace existing point trackers without interfering with the structure of the graphical model.

Algorithm 1: Randomized feature tracking (RFT) filter

```
input    : Video sequence with $L$ frames and target bounding box in first frame;
Initialization;
  $x_0^0$ = Center of bounding box;
  Draw $F$ point trackers uniformly in the bounding box and set point tracking parameters;
  Set compatibility function parameters $\mathbf{t}_f, f = 1 : F$;
  for $n \leftarrow 2$ to $L$ do
      $\left\{m_{f\to 0}\big(x_n^0|q_{n-1}^f\big), f = 1 : F\right\} \leftarrow$ ComputeMessages $\left(\left\{q_{n-1}^f, \mathbf{t}_f, f = 1 : F\right\}\right)$;
      $\{v^f, f \in I, |I| \le F\} \leftarrow$ Clustering $\left(\{p(x_n^f|y_n^f; \Phi_n), f = 1 : F\}\right)$;
      $\{\alpha_n^f, f = 1 : F\} \leftarrow$ RejectionControl $\left(\mathcal{N}\big(v^f; \mu, \Sigma\big)\right)$; /*See sec. 4.4.2*/
      $p\big(x_n^0|Y_n; \Phi_n\big) \leftarrow$ ComposePosterior $\left(\left\{m_{f\to 0}\big(x_n^0|q_{n-1}^f\big)\forall f \ni \alpha_n^f = 1\right\}\right)$; /*See sec.
      4.4*/
      if $p\big(x_n^0|Y_n; \Phi_n\big)! = 0$ then
          output : Bounding box centered on $\mathbf{E}\left[p\big(x_n^0|Y_n; \Phi_n\big)\right]$;
          ReplaceOutlierPointTrackers $\left(p\big(x_n^0|Y_n; \Phi_n\big)\right)$; /*See Eqn. 4.5.1*/
      end
      else
          output : Tracking failure and exit;
      end
  end
```

4.4.4 Dealing with occlusions

It is a hard problem to detect occlusions, especially when the occluding object presents an appearance similar to the tracked object. Even so, some mechanism is needed to signal a possible occlusion event so that some corrective action can be taken. In the randomized feature tracking scheme, an occlusion event is signalled if outlier feature points (See Eqn. 4.4.10) are a majority, that is, greater than $\frac{F}{2}$ (F even) or $\frac{F+1}{2}$ (F odd). In such a case, any replacement of outliers is arrested temporarily to avoid *drifting*. The hope is that once the occluded object reappears the point trackers may refind their correct tracks. However such a fortunate event is driven more often than not by chance factors and thus cannot be generally relied upon. The algorithm so standing is then for all practical purposes considered vulnerable to occlusions. Aside from this drawback, the randomized feature point tracker possesses some useful capabilities as gathered from the following experiments. A pseudo-code to aid implementation of the randomized feature tracking filter (RFT-filter) is given in Algorithm 1.

4.5 Experimental setup

The randomized feature point tracker is put to two tests, the first with a goal to track the position of a human head (its center) in a video sequence and the second with a goal to track a moving vehicle from an aerial video sequence. These sequences are chosen with a view to qualitatively assess the performance of the tracker under varying target appearances, environmental variations like lighting, shadow effects and occlusions.
In all experiments, the tracked targets were manually initialised via a target bounding box. The positions of the feature points in the initialisation frame are chosen by drawing samples uniformly within the provided bounding box. Their number, dimensions of their image appearance templates and search spaces are hand tuned to comply with the expected complexity and magnitude of motion of the target in the video sequence. At each frame following the initial one the bounding box in the sample results (for instance in Fig. 4.8) is centered around the *expected value* of the computed posterior in Eqn. 4.4.8.
For each newly sampled feature point tracker x_n^f (at the initialisation stage or replacement stage), the corresponding parameter $\mathbf{t}_f$ of the compatibility function $\delta_{x_n^f}\left(x_n^0 - \mathbf{t}_f\right)$ is updated in the following manner.

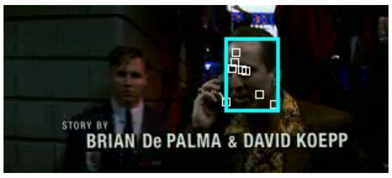
$$\mathbf{t}_f = \mathbf{E}\left[p\left(x_n^0|Y_n;\Phi_n,\mathcal{T},\left\{q_{n-1}^f\right\}_{f=1:F}\right)\right] - s^f, f = 1:F. \tag{4.5.1}$$

4.6 Results and discussions

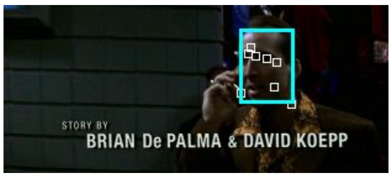
The results of the two tests are discussed in detail below.

Head tracking

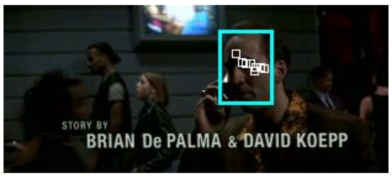
A series of snapshots of head tracking results for the Snake-eyes test sequence are presented in Figs. 4.8, 4.9. The total track length until failure is about 180 frames and therefore only a small informative selection among these frames are displayed. In each frame, the feature points (equivalently the centers of their image apperance templates) are marked by small squares.
Two variants of results of head tracking are discussed below.



(a). Frame 1

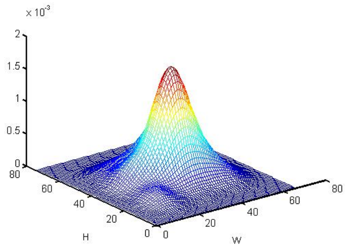


(b). Frame 26

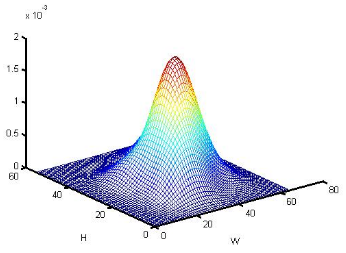


(c). Frame 70

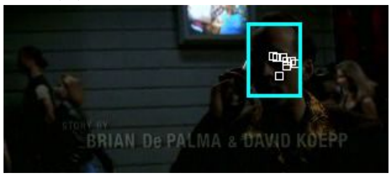
(d). Posterior at frame 2



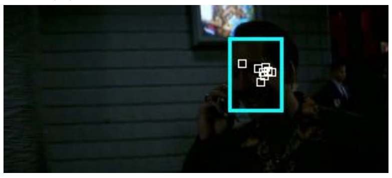
(e). Posterior at frame 26

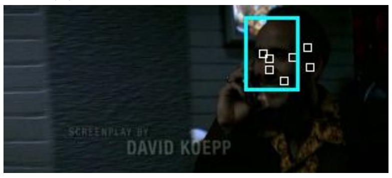


(f). Posterior at frame 70

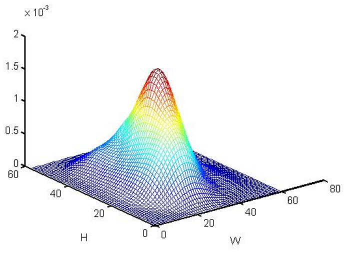


(g). Frame 86

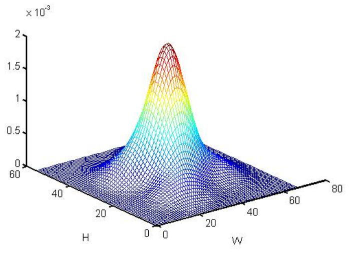

(h). Frame 98



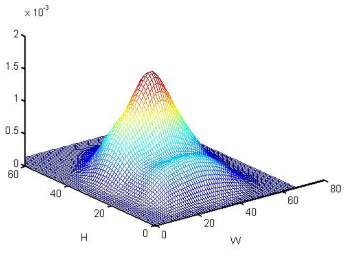
(i). Frame 108



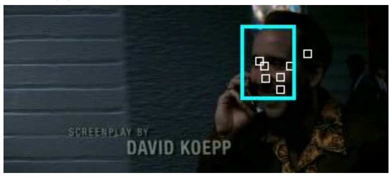
(j). Posterior at frame 86



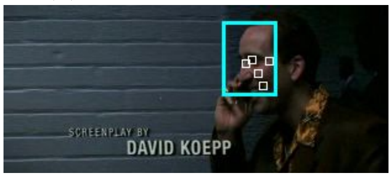
(k). Posterior at frame 98



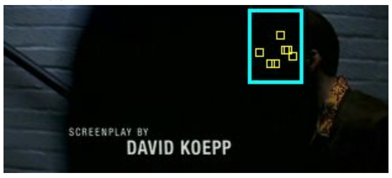
(l). Posterior at frame 108



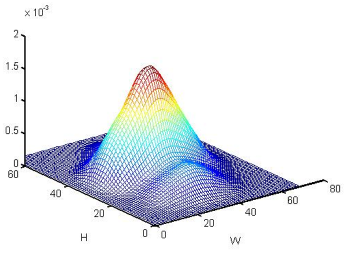
(m). Frame 113



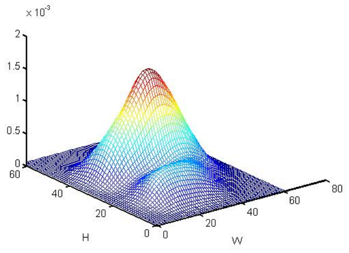
(n). Frame 120



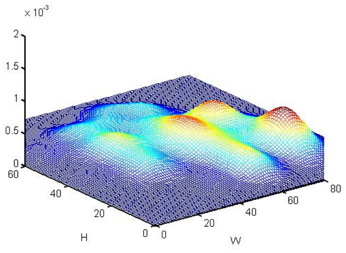
(o). Frame 173



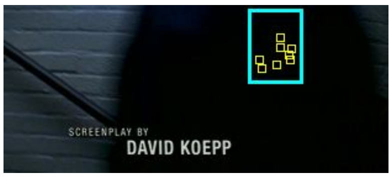
(p). Posterior at frame 113



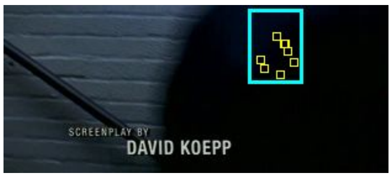
(q). Posterior at frame 120



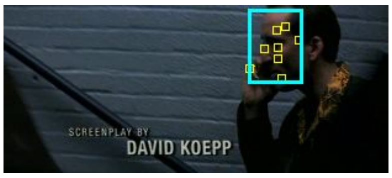
(r). Posterior at frame 173



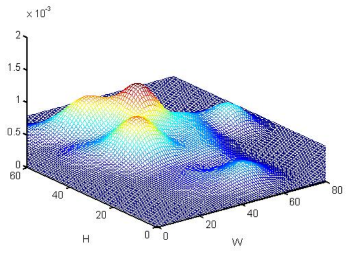
(s). Frame 174



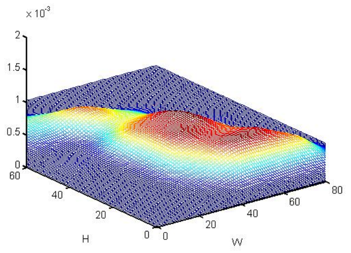
(t). Frame 175



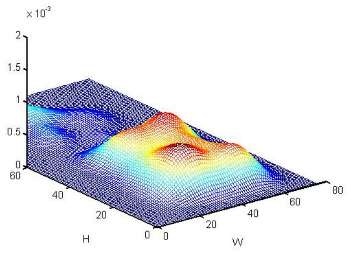
(u). Frame 178



(v). Posterior at frame 174



(w). Posterior at frame 175



(x). Posterior at frame 178

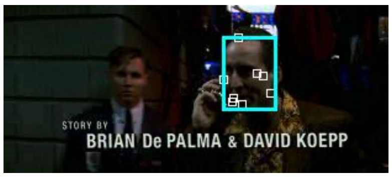
Figure 4.8: Randomized feature point tracking on the Snake-eyes sequence - replacement from posterior of target. Template size = 21×21, $S = 61 \times 61$, $r = 5.0$, $\rho = 0.8$.

Replacing point trackers by draws from the filtering distribution of the target position:

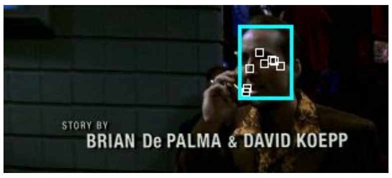
In this experiment the outlier point trackers are replaced using samples from the target posterior in accordance with Eqn. 4.4.11. Under each selected frame in Fig. 4.8 is a graphic display of the target posterior seeking to provide some insight into its evolution over time.

The ability to track through shadow effects, Fig. 4.8(b), and to overcome *illumination changes*, Figs. 4.8(c,g,h,i,m), can be seen from these result samples. The distinct unimodality of the posterior under these conditions can be observed from Figs. 4.8(e,f,j,k,l). Between pairs of posteriors in Figs. 4.8(f,j) and Figs. 4.8(k,l), one can observe an increase in the spread of the posterior due to changes in illumination. The tracker also remains stable under small *pose changes*, see Figs. 4.8(m,n), due to online point tracker replacement. The increased spread of the posterior under pose change is however observable in Figs. 4.8(p,q).

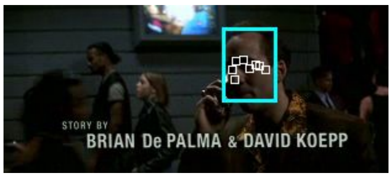
Following these events is an *occlusion* of the target by an object of dissimilar appearance, Figs. 4.8(o,s,t). The color of the feature point squares are changed to yellow to indicate the absence of any inliers. From Eqn. 4.4.8 it is seen that the posterior is degenerate. However, for the sake of a comparative study, all the feature points are considered inliers temporarily and a posterior is derived as usual. Reasonably, the posteriors are almost degenerate in Figs. 4.8(r,v,w) during the period of the occlusion. When the tracked object emerges out of the occlusion, it appears from Fig. 4.8(u) that



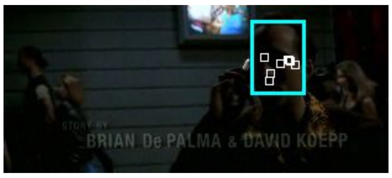
(a). Frame 1



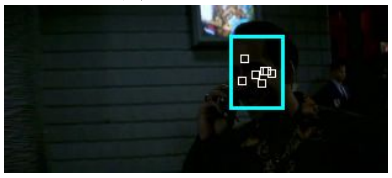
(b). Frame 26

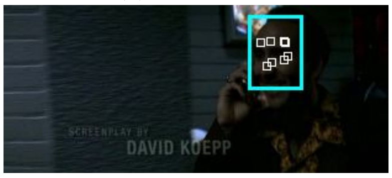


(c). Frame 70

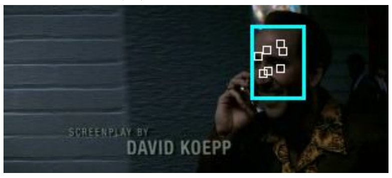


(d). Frame 86

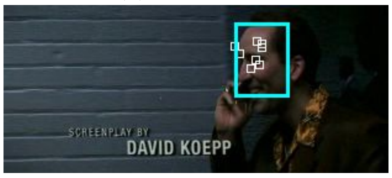

(e). Frame 98



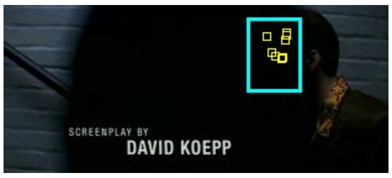
(f). Frame 108



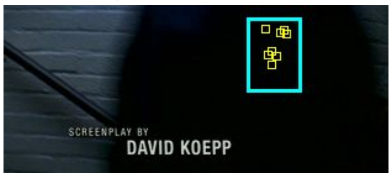
(g). Frame 113



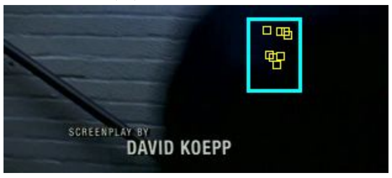
(n). Frame 120



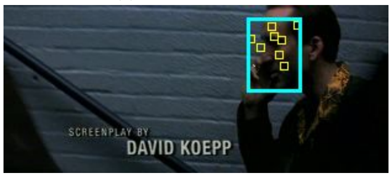
(i). Frame 173



(j). Frame 174



(k). Frame 175



(l). Frame 178

Figure 4.9: Randomized feature point tracking on the Snake-eyes sequence - replacement from point tracker posterior. Template size $= 21 \times 21$, $S = 61 \times 61$, $r = 5.0$,$\rho = 0.8$.

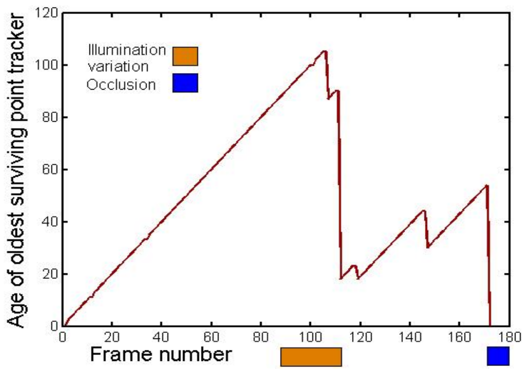


Figure 4.10: A plot of the age of the oldest active point tracker versus the frame number for the Snake-eyes sequence.

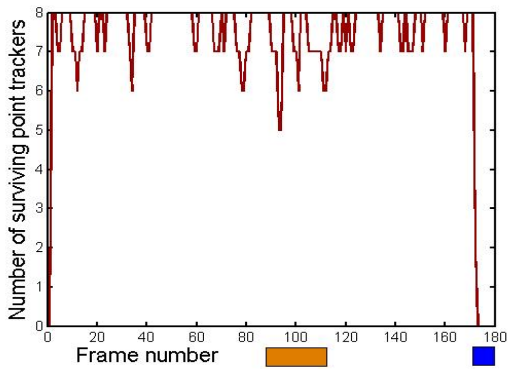


Figure 4.11: A plot of the number of point trackers declared inliers at each frame versus the frame number.

the tracker has refound the correct track. This is only a chance effect since the camera happens to follow the tracked object in this sequence. Strictly speaking the tracker has failed to overcome the occlusion. This fact is corroborated by the form of the posterior in Fig. 4.8(x).

Replacing point trackers by draws from the filtering distribution of a point tracker: The following sampling technique may indeed be used for replacement of outlier point trackers without compromising the robustness of the tracker.

$$s^f \sim p\Big(x_n^{\hat{f}} | y_n^{\hat{f}}; q_{n-1}^{\hat{f}}\Big), \tag{4.6.1}$$

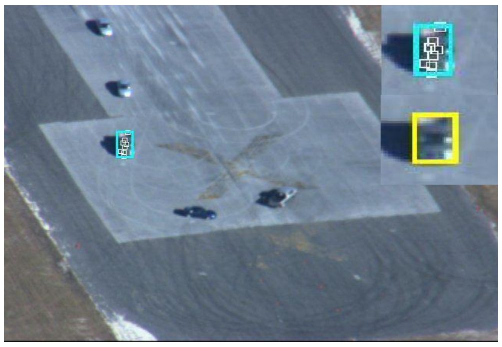
where, if the point trackers are sorted in a descending order of their track lengths, then $\hat{f}$ is the index of the *median length* point tracker. Intuitively, this advocates sampling of a new point tracker from the posterior of a point tracker which lies somewhere in between the most stable and the most recent ones, thereby taking a balanced risk. Fig.4.9 presents the results of head tracking in the Snake-eyes sequence using this form of replacement. Except during pose changes, the behaviour of the tracker is not significantly different as compared to Fig.4.8.

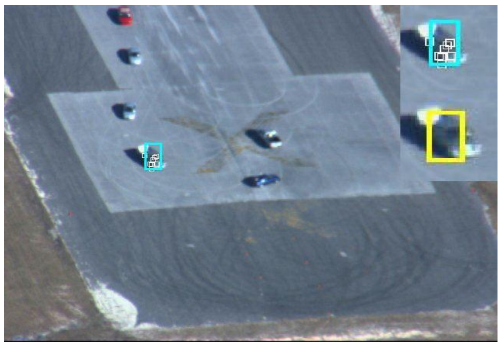
To round off the discussions on the results of head tracking, two more relevant analyses need to be made. First, to observe how many point trackers are replaced at each tracking instant and second, to observe the resilience of the *sampled* point trackers. These observations provide an insight into the stability of tracking the object. The graphical plot of the *age* in frames of the oldest active point tracker during head tracking in Fig. 4.9 is shown in Fig. 4.10. Interesting events in the sequence are labelled in these plots. As could be expected, the replacement activity is slightly hightened during illumination changes, thus reducing the age of the oldest tracker. Overall, the average age of point trackers is a healthy $40-60$ frames, an indicator that a bunch of "randomly" picked point trackers can provide stable tracking. The occlusion brings the ages of point trackers to zero.

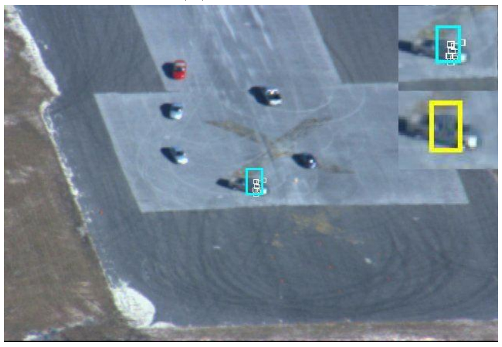
The graphical plot in Fig. 4.11 shows the number of point trackers declared as inliers at each tracking instant, for the sequence in Fig. 4.9. It can be seen that the occlusion event eliminates most of the point trackers as expected, otherwise only a minority are eliminated even under extreme illumination changes.

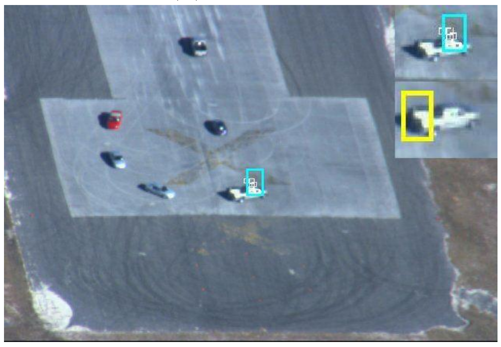
Vehicle tracking in an aerial video sequence

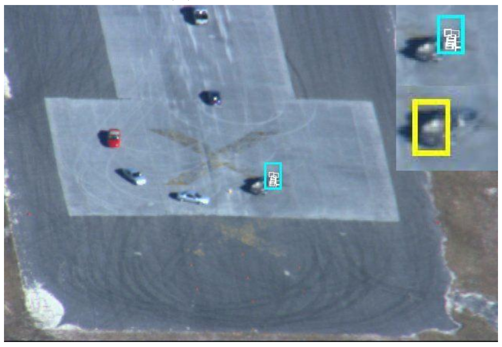
The idea behind this experiment is to test the resilience of the randomized point tracker

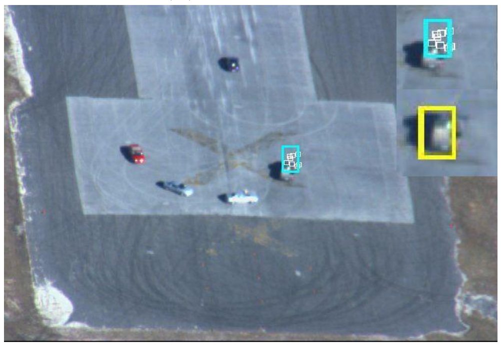

(a). Frame 1

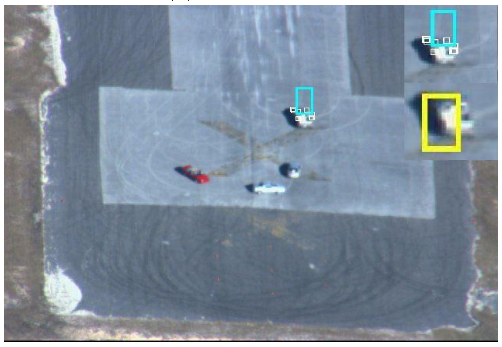

(b). Frame 60


(c). Frame 120


(d). Frame 180


(e). Frame 210


(f). Frame 240


(g). Frame 300

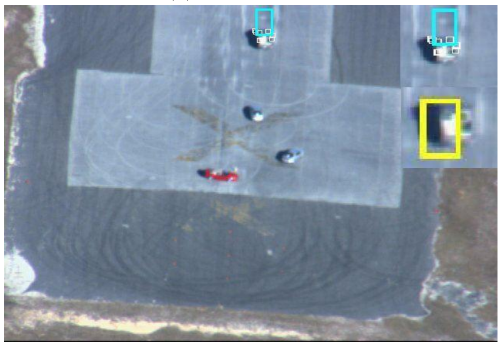

(h). Frame 360

Figure 4.12: Aerial sequence tracking using randomized feature tracker. Template size = 21×21, $S = 41 \times 41$, $r = 5.0$, $\rho = 0.8$. Overlaid, with yellow bounding boxes, is result of color based particle filtering. Video courtesy - CMU VIVID database.

to large pose changes, in spite of the fact that there is currently no device in it to accomodate such events. In lieu of the small size of the target, this experiment utilises replacement of point trackers by draws from the target posterior, thus encouraging dense clusters of points. The target area in the result samples are enlarged and overlaid on the images for visual assessment.
The drift in the *estimate* of the position of the target is apparent from the result samples in Figs. 4.12(c,d,e,f,g,h). Nevertheless the point trackers remain attached to the target throughout the length of the sequence, aided by the fact that the motion of the target is smooth. For a qualitative comparison, the results of tracking the same target with a color based particle filter [Perez et al., 2002] are overlaid. It can be seen that this filter too suffers from tracking drift due to illumination changes.

The issue of tracking drift
When dealing with point trackers, the issue of drift in the estimation of the target position arises, especially when the target undergoes large pose changes. As it stands, the randomized point tracker is not particularly designed for tracking under large pose changes. However, from the perspective of the graphical model, the ability to tackle tracking drift is directly related to the ability to update the compatibility function in Eqn. 4.4.2. For instance, by updating parameters $\mathbf{t}_f, f = 1 : F$ online by some mechanism or aid of another cue, the compatibility function can be updated appropriately. Thus the graphical model is general enough to accomodate such updates.

4.7 Conclusion and prospects

The principal contribution of this chapter is the probabilistic graphical model for measurement fusion using linear message combinations. This model has the flexibility to introduce empirical knowledge about the task in hand into the filtering with a goal to perform consistent fusion. It is hoped that such models will be of use in other vision problems which demand a provision be made for online fusion of only valid or consistent measurements.
In the simplest case, this message combination model can be reduced to a message switching model. Based on a probabilistic graphical model representation of point tracking, a novel randomized feature point tracker is derived from this switching model.

This tracker is seen to be useful when tracking a target under strong illumination variations and minor pose changes. Currently this tracker has no capability to handle large pose changes or occlusions. However these drawbacks are not a weakness of the graphical model itself. Arguably, the graphical model is general enough to accomodate corrective measures to alleviate such ailments.
The randomized feature point tracker has a very useful characteristic, an inherent ability to adapt to appearance changes of the target by replacement of point trackers. Combining this characteristic with its probabilistic nature results in a very useful filter which is now ready to be integrated with other generative model based stochastic filters. This is the very motivation and substance of the following chapter on multi-cue fusion. To recapitulate, beginning from a stochastic interpretation of point tracking and progressing through graphical message combination models, the contributions of this chapter have paved a way for the interplay of discriminative tracking approaches like point tracking and generative model based approaches towards robust fusion.

Appendix: Assumed density filtering viewpoint for point tracking

Assumed density filtering (ADF)

Consider the following first order Hidden Markov model (HMM).

$$p\big(x_n|y_{1:n}\big) \propto l\big(y_n|x_n\big) \int p\big(x_n|x_{n-1}\big)p\big(x_{n-1}|y_{1:n-1}\big)dx_{n-1}. \tag{4.7.1}$$

In step one, say an approximation q_{n-1} is substituted instead of the filtering distribution at $n-1$ in the above equation ($q_0 \equiv p\big(x_0\big)$). The following approximate filtering distribution can then be obtained.

$$\hat{p}\big(x_n|y_{1:n}\big) \propto l\big(y_n|x_n\big) \int p\big(x_n|x_{n-1}\big)q\big(x_{n-1}\big)dx_{n-1}. \tag{4.7.2}$$

Now, in step 2, the approximate filtering distribution can be re-approximated by another parametric distribution $q\big(x_n\big)$ by minimizing the KL-divergence between them. If the distributions q come from an exponential family, such a minimization is equivalent to matching moments of the approximate filtering distribution to the ones of q. These deliver the moments of q (its sufficient statistics). The basic idea behind ADF is to repeat these two steps at each filtering instant, thereby propagating the moments of the filtering distributions. The ADF concept is useful to emulate point tracking techniques such as normalized cross correlation (NCC) or sum of squared distances (SSD) based point tracking. Two forms of ADF viewpoints of point tracking are described below.

Propagating delta approximations

Consider the following standard model of a first order Hidden Markov Model (HMM).

$$p\big(x_n|y_{1:n},\vartheta\big) \propto l\big(y_n|x_n,\vartheta\big)\int p\big(x_n|x_{n-1}\big)p\big(x_{n-1}|y_{1:n-1},\vartheta\big)dx_{n-1}, \tag{4.7.3}$$

where the notations follow from Sec.4.2. Let density $q_{n-1}\big(x_{n-1}|\zeta\big)$ (ζ is a parameter) be an approximation to the filtering distribution $\hat{p}\big(x_{n-1}|y_{1:n-1},\vartheta\big)$. With this approximation, consider the exact posterior shown below.

$$\hat{p}\big(x_n|y_{1:n},\vartheta,\zeta\big) \propto l\big(y_n|x_n,\vartheta\big)\int p\big(x_n|x_{n-1}\big)q_{n-1}\big(x_{n-1}|\zeta\big)dx_{n-1}. \tag{4.7.4}$$

Now assume the following special form for the density q.

$$q\big(x_{n-1}|\zeta\big) = \delta_\zeta\big(x_{n-1}\big). \tag{4.7.5}$$

Using this form, the exact posterior in Eqn. 4.7.4 reduces to the following.

$$\hat{p}\big(x_n|y_{1:n},\vartheta,\zeta\big) \propto l\big(y_n|x_n,\vartheta\big)p\big(x_n|x_{n-1}=\zeta\big). \tag{4.7.6}$$

This is in a form suitable for point tracking as the prior $p\big(x_n|\zeta\big)$ can be conveniently defined to suit grid search as in Sec. 4.2. Finally, minimizing the KL-divergence $D\Big(\hat{p}_n\|q\big(x_n|\tilde{\zeta}\big)\Big)$ under the constraint that $E_{q\left(x_n|\tilde{\zeta}\right)}\left[x_n\right] = E_{\hat{p}_n}\left[x_n\right]$ simply results in $\tilde{\zeta} = E_{\hat{p}_n}\left[x_n\right]$. With $\zeta \leftarrow \tilde{\zeta}$,

$$\hat{p}\big(x_n|y_{1:n},\vartheta,\zeta\big) \approx q_n\big(x_n|\zeta = E_{\hat{p}}\left[x_n\right]\big). \tag{4.7.7}$$

Therefore, the filtering distributions are approximated by *point masses* around their expected values. If it is assumed that these filtering distributions are unimodal (which is usually implicit for point tracking), then this assumed density filtering describes discriminative point tracking in practice.

Propagating Gaussian approximations

Assume $p\big(x_n|x_{n-1}\big) = \delta_{x_{n-1}}\big(x_n\big)$ and an assumed density of the form indicated below.

$$q_{n-1}\big(x_{n-1}\big) = \mathcal{N}\big(x_{n-1};\mu_{n-1},\Sigma_{n-1}\big), \tag{4.7.8}$$

where $\mathcal{N}\big(x_{n-1}; \mu_{n-1}, \Sigma_{n-1}\big)$ is a Gaussian distribution in x_{n-1} with mean an variance parameters μ_{n-1}, Σ_{n-1} respectively. Now from Eqn. 4.7.3 the exact posterior is as follows.

$$\hat{p}\big(x_n|y_{1:n}, \vartheta\big) \propto l\big(y_n|x_n, \vartheta\big)\mathcal{N}\big(x_n; \mu_{n-1}, \Sigma_{n-1}\big), \tag{4.7.9}$$

which, under the assumption that the posteriors are unimodal, implies the prior is centered on the estimated location of the point tracker at $n-1$. Now once again if sufficient samples are drawn from this prior then this brings the computation of the posterior closer in spirit to regular grid searches. Finally, the exact posterior is replaced by a Gaussian with matching moments. As compared to the viewpoint prescribed in Sec. 4.2, these ADF viewpoints are restrictive due to the specific class of distributions they propagate.

5

Bayesian multi-cue fusion

5.1 Introduction

Visual tracking literature is abundant with schemes that discuss, recommend and propose tracking algorithms based on measurements from multiple complementary cues, such as color, motion or shape in order to mitigate the drawbacks of single cue tracking, as in [Perez et al., 2004; Badrinarayanan et al., 2007a; Wu and Huang, 2004; Toyama and Horvitz, 2000; Han et al., 2007; Triesch and von der Malsburg, 2000; Wang et al., 2004]. Such multi-cue trackers need to deal with problems of measurement fusion. Numerous approaches to measurement fusion have been proposed over the recent years. Before embarking on the literature review the motivational ideas for multiple cue tracking is presented below.

The case for multi-cue tracking techniques

To understand why multi-cue tracking is necessary, it is useful to answer the converse question: why single-cue tracking is insufficient in practice. In principle single-cue tracking would be sufficient if the cue with which the target is described has enough power to discriminate the target from any other image data and the approximate posterior distribution of the tracked state accurately describes the uncertainty in the tracked state. Both of these requirements are difficult to satisfy. Even if a cue were to be found which has sufficient discriminative power, it is generally difficult to specify the likelihood function associated with it due to dimensionality and normalization problems (see Chapter 3). Secondly an extremely large computational power would be necessary to describe and maintain the multi-modality, if any, of the posterior distribution. Due to these reasons the computational approximation of the posterior distributions do not accurately represent the uncertainty of the tracked state. Therefore, an alternative to improve the estimation of the tracked state is to somehow average over multiple measurements (equivalently cues) and decide on the estimate of the tracked state by consensus arguments. By doing so the discriminative power of the target reference model is improved and associating each kind of measurement to different filters with varying kinematic models allows the state space to be explored in multiple ways. These aspects motivate the use of multiple cues and filters for tracking an object. A review of some key techniques for multi cue tracking are now presented below.

Techniques for multi-cue fusion tracking

A list of multi-cue fusion techniques in a somewhat increasing order of complexity is presented below. Where relevant, a noteworthy example from literature is discussed to aid the commentary. This review is geared towards a conceptual classification of several techniques and is by no means an exhaustive survey.

1. **Likelihood Factorisation Based Approaches**

   In these approaches the filtering law is factorised as shown below.

   $$p\big(x_n|\underline{Y}_n\big) \propto \prod_{j=1}^{M} p\big(y_n^j|x_n^j\big)p\big(x_n|\underline{Y}_{n-1}\big); \tag{5.1.1}$$

   where measurement $\underline{Y}_n = \{y_{1:n}^1, y_{1:n}^2, \ldots, y_{1:n}^M\}$. Notable among the techniques relying on this assumption is the BlackBox technique proposed by [Leichter et al., 2006], wherein the *combined* filtering distribution is shown to be:

   $$p\big(x_n|\underline{Y}_n\big) \propto \frac{\prod_{j=1}^{M} p\big(x_n|y_n^j, \underline{Y}_{n-1}\big)}{\big[p\big(x_n|\underline{Y}_{n-1}\big)\big]^{M-1}}. \tag{5.1.2}$$

   Their technique aims to combine several filtering distributions using Eqn. 5.1.2, thereby pretending to fuse only the outputs of filters while remaining oblivious to the internal characteristics of the filters. This BlackBox treatement holds only in as far as the numerator in Eqn. 5.1.2 is concerned as the evaluation of the denominator requires the specification of a kinematic prior. In order to preserve the BlackBox nature of this technique the following assumption is made.

   $$p\big(x_n|\underline{Y}_{n-1}\big) = p(x_n) \Longrightarrow p\big(x_n|x_{n-1}\big) = p(x_n). \tag{5.1.3}$$

   This is untenable since the kinematic prior is necessary to generate the output filtering distribution of each black box.

Apart from the treatment found in [Leichter et al., 2006], the techniques based on likelihood factorisation have the virtue of mathematical simplicity and ease of implementation, but this assumption does not comply with reality. Measurements based on cues such as color and texture are in general dependent, therefore failing this assumption. This assumption is to be further discouraged in the presence of clutter or occlusions where there is a possibility of having non-overlapping filtering distributions.

2. **Factored State Based Approaches**

   The broad idea behind these approaches is the factorisation of the tracked state into several sub-states which are linked via observation model(s), effectively disengaging from likelihood factorisation. Exact probabilistic inference to compute the marginal posteriors (or beliefs) of these sub-states on such models is in general intractable. Therefore these models are approximated by simpler graphical models on which exact probabilistic inference is tractable.

   The co-inference technique of [Wu et al., 2003] for multi-cue fusion is a factored state approach based on the variational approximation technique [Ghahramani and Jordan, 1995] for simplifying inference. In the approximate graph, the observation probabilities in the original graph are replaced by *variational parameters*, leading to uncoupling of the sub-state Markov chains. These variational parameters are estimated using iterative optimization to arrive at the sought approximation. In particular, minimization of the KL-Divergence [Cover and Thomas, 1991] between the original and the approximate models result in fixed-point equations for these variational parameters. The form of these equations reveals an interesting relationship between the hidden sub-states. *The variational parameters of a sub-state are a functional of the belief of every other sub-state except itself.* These parameters effectively recouple the hidden sub-states by so called *co-inference* and this forms the basis of their proposed tracker.

   The authors do not attempt to provide a computational analysis of the co-inference concept adapted to tracking and instead derive a multi-cue tracker inspired by it. This tracker is implemented using an iterative sequential MC simulation based approach. At each instant, a color importance prior is somehow used to derive shape samples and these are weighed by a shape likelihood. Then, symmetrically, a shape importance prior is somehow used to derive color samples and these are weighed by color likelihoods. This process is iterated. The idea behind these iterations is that participation of both likelihoods will help propagate the best shape and color samples to the next instant. The details of this technique can be found in [Wu et al., 2003].

A shortcoming of this iterative importance sampling is that no importance samples are drawn from the priors defined for the individual hidden sub-states themselves. Therefore, there is an untenable assumption that measurements based on one of the cues are always consistent. This makes it difficult to uncouple the beneficial cue from the inconsistent cue. Further a wrong ordering of cues in the sampling iterations can mitigate the beneficial effects of both cues. Finally, it is unclear from the description in [Wu et al., 2003] as to how to translate between the shape and color spaces.

3. **Kernel Based Approaches**

   The fundamental tenet of these approaches is to model all the relevant distributions in Bayesian filtering as a mixture of Gaussians, thereby enabling the propagation of multi-modal analytic filtering distributions [Han et al., 2005; 2007]. One such scheme attempting sensor (read cue) fusion in a Gaussian Kernel based Bayesian filtering framework is by Han et al [Han et al., 2007]. The key idea in this scheme is to partition a fixed number of importance samples amongst the constituent sensors based on each sensor's future potential of high observation likelihood; thereby devoting a greater number of particles to it. To do so, importance sampling is broken down into two stages. In the first stage the importance samples for all the sensors are drawn from a common proposal (the predictive prior) and the individual sensor measurements evaluated. These measurements are then re-fed in a linear combination with the common proposal to generate the final (common) proposal. A sample from this proposal is *assigned* to one of the constituent sensors upon evaluation of the combined *weight* of the sample according to the sensors predictive prior and measurement likelihood. The principal argument of this Kernel based approach is that Kernel based modeling of these distributions aids in easy evaluation of these weights.

   Although such a scheme is appealing, the model is not general enough to handle a network of hidden states and the issue of balancing the predictive prior and the likelihood, in course of developing the final common proposal, still remains unresolved. In [Han et al., 2007] this balance is maintained constant, which is not very convincing since the very objective of designing this *hindsight* proposal is to adaptively weigh the likelihood and the predictive prior. This approach also bears striking resemblance to the more general non-parametric scheme proposed by [Vermaak et al., 2003], the only difference being that the number of particles per mixture component is held constant in [Vermaak et al., 2003]. Finally, the mixture weights balancing the filtering distributions from various sensors are seen to be proportional to each sensors model evidence, similar to the likelihood combination approach described earlier.

4. **Dependent Multi-Cue Fusion**
These approaches attempt to condition measurements with all other preceeding measurements at the cost of additional complexity in simulation. A recent example is [Moreno-Noguer et al., 2008]. The key feature of such techniques is illustrated in the following example involving two hidden states belonging to different state spaces, each with a different observation model.
Define the following *static* state space model where all x_1, x_2 represent hidden states associated with corresponding measurements y_1, y_2.

$$x_1 \sim p(x_1). \tag{5.1.4}$$

$$y_1 = g_1(x_1) + \zeta_1. \tag{5.1.5}$$

$$x_2 = f_2(x_1, y_1) + \eta_2. \tag{5.1.6}$$

$$y_2 = g_2(x_1, x_2, y_1) + \zeta_2. \tag{5.1.7}$$

In the above model, it is assumed that the statistics of the noise random variable η_2 is known and statistics of the noise random variable ζ_1, ζ_2 are unknown. In addition it is taken that functions g_1, g_2 are non-linear functions of their arguments while f_2 is a function such that the conditional density $p(x_2|x_1, y_1)$ can be easily sampled from. With the foregoing definitions and assumptions, the joint posterior of the hidden variables can be factorised as follows.

$$p(x_1, x_2|y_1, y_2) \propto p(y_2|y_1, x_1, x_2)p(x_2|x_1, y_1)p(x_1|y_1). \tag{5.1.8}$$

Assume a sample based approximation of the posterior $p(x_1|y_1)$ is available. Then construct a sub-optimal importance sampling proposal as shown below.

$$q(x_1, x_2|y_1) = p(x_2|x_1, y_1)p(x_1|y_1). \tag{5.1.9}$$

Draw samples of the joint state $x_1^i, x_2^i \sim q(.), i \in 1 \dots N$. Then an approximation of the joint posterior can be obtained as follows.

$$p(x_1, x_2|y_1, y_2) \propto \sum_{i \in 1 \dots N} p(y_2|y_1, x_1^i, x_2^i)\delta_{x_1^i, x_2^i}(x_1, x_2). \tag{5.1.10}$$

One of the issues with such an approach is the requirement of densities such as $p(x_2|x_1, y_1)$ which are data dependent. Such densities need to be learnt by a strategy of some sort. Further, an extension to sequential filtering runs into difficulties as discussed below.

Define the following state space model in accordance with the deliberations in [Moreno-Noguer et al., 2008].

$$x_t^1 = x_{t-1}^1 + \eta_t^1. \tag{5.1.11}$$

$$y_t^1 = g_1(x_t^1) + \zeta_t^1. \tag{5.1.12}$$

$$x_t^2 = f_2(x_{t-1}^2, x_t^1, y_t^1) + \eta_t^2. \tag{5.1.13}$$

$$y_t^2 = g_2(x_t^2, x_t^1, y_t^1) + \zeta_t^2. \tag{5.1.14}$$

In the above model, the statistics of the noise processes η_t^1, η_t^2 are assumed known while the statistics of the noise processes afflicting the measurement model are unknown. The measured variables are considered non-linear functions of their respective arguments as before. After some straightforward manipulations, the joint state filtering distribution can be factorised as shown below.

$$p(x_t^1, x_t^2|y_{1:t}^1, y_{1:t}^2) \propto p(y_t^2|y_t^1, x_t^1, x_t^2)p(x_t^2|x_t^1, y_t^1)p(x_t^1|y_{1:t}^1, y_{1:t-1}^2). \tag{5.1.15}$$

Now, it is not easy to approximate the last factor in the above proportionality as dependencies on the data $y_{1:t-1}^2$ are introduced due to *temporal loops* in the state space model. For the same state space model, [Moreno-Noguer et al., 2008] propose to evaluate distributions of the following form (notice the conditioning only on the instantaneous data):

$$p^t(x_t^1, x_t^2|y_t^1, y_t^2) \propto p_t^2(x_t^2|y_t^{1,2}, x_t^1; p_{t-1}^2)p_t^1(x_t^1|y_t^1; p_{t-1}^1). \tag{5.1.16}$$

Although seemingly convenient there remain doubts as to how to arrive at such a factorisation while remaining consistent with the state space model. An alternative factorisation, consistent with the state space model, would be as shown below.

$$p^t(x_t^1, x_t^2|y_t^1, y_t^2) \propto p(y_t^2|y_t^1, x_t^1, x_t^2)p(x_t^2|x_t^1, y_t^1)p(x_t^1|y_t^1). \tag{5.1.17}$$

To proceed from this stage first requires an approximation of the last factor in the above proportionality, for which an instanteneous prior $p(x_t)$ is necessary. Meaningful priors of this form are difficult to define. Apart from these issues it is difficult to define likelihoods which are conditioned on other data. Finally, imposing a *static ordering* of the state variables with no possibility to instantaneously change their dependency structure is definitely restrictive.

5. **Integration Avoiding Measurement Inconsistency**
When the objective is to fuse several measurements underlying a network of trackers, it is prudent to verify the mutual compatibility of the measurements to avoid errors. An approach advocating this verification is proposed by [Hua et al., 2006]. This scheme is derived for a pairwise Markov Random Field (MRF) with Gaussian priors and isotropic Gaussian likelihood densities. The idea is to iteratively compute the MAP estimate of the hidden states and the ML estimate of the variance of the pairwise priors using EM iterations and observe the convergent value of the variances. If, for a pair of hidden states and their associated measurements, the convergent value of the estimated variance for their pairwise prior is away from zero, then the pair of associated measurements are said to be incompatible with each other. The intuition here is that, if the measurements are consistent then a state can predict its neighbour's state accurately, thereby attaining a delta compatibility (variance goes to zero).
Their proposition is to discard all measurements which have a majority disagreement with its neighours. Once this done, statistical inference is performed by including only the consistent measurements. The usefulness in discarding inconsistent measurements is demonstrated on multi-part tracking with a color based particle filter and flow based Lucas-Kanade tracker. This scheme is shown to work with missing measurements, inconsistent measurements and consistent measurements. However this scheme has some drawbacks. The ideas are demonstrated on a static Markov network with offline learnt compatibilities. The form of the compatibilities are Gaussian and so is the form of the likelihood functions. As such, there is no trivial extension to a non-linear sequential filtering problem. The compatibilities are assumed fixed, while in a more general multi-part tracking scenario the compatibilities undergo changes (see Chapter 6). Nevertheless, the spirit of their work is retained in the forthcoming propositions, where another

mechanism is developed to *pick out* and fuse (consistent) measurements in more general scenarios.

The review so far summarised some of the pertinent and representative techniques of multi-cue fusion, primarily in a Bayesian framework. Their advantages and disadvantages were duly brought to light. This chapter presents a probabilistic model which encompasses the advantageous ideas from the foregoing techniques is presented. This model is based on the message combination concepts introduced in Chapter 4. The principal advantage of this model is that it allows the posterior of the hidden state of the target to be composed by *picking* or *mixing linearly* one or more *messages* from various other hidden states. It is as if, at each instant, the model automatically chooses messages to traverse through some particular paths to the hidden state of the target. By doing so, the model provides the flexibility to choose, leave out or balance a measurement with others, to perform fusion.
While a convenient model is imperative for cue fusion, integrating multiple cues alone does not guarantee robust tracking. The choice of cue(s) associated to each hidden state and the extent to which it complements other cues dictates the robustness of the tracker. A cue, in general, describes an apperance model of the target, a common example is a color histogram of the target [Comaniciu et al., 2000; Perez et al., 2002; K.Nummiaro et al., 2003; Vermaak et al., 2002]. Thus each of the hidden states can be associated to different appearance models of the target. This collection of appearance models can be seen as a *discrete set appearance model* of the target. This set viewpoint becomes interesting when first, the elements of this set are replaced online and second, when appearance model adaptation involves cue interaction; such a setup is a key ingredient for robust tracking. This forms another important subject matter of this chapter.
The contents of this chapter are arranged in the following manner. A probabilistic graphical model for multi-cue fusion is presented in Sec.5.2. Based on this model a novel multi-cue tracker is derived in Sec.5.3. The experimental setup to test this tracker is described in Sec.5.4. Both qualitative and quantitative results of the tests are discussed in Sec.5.5. Another setting of the parameters of the graphical model is used to derive a novel multi-cue combination tracker in Sec. 5.6. The nuances of this tracker and the advantages it brings over the multi-cue switch tracker are discussed with sample results. Drawbacks of these trackers alongwith some pointers to possible improvements are summarised in Sec.5.7. Conclusions are drawn in Sec.5.8.

5.2 Probabilistic graphical model for multi-cue fusion

The model for multi-cue fusion is an extension of the graphical model presented in Sec.4.3 of Chapter 4 for tracking with a set of point trackers. This model is now extended to include a color based particle filter via an addition of a new link in the graph, see the bottom part of Fig. 5.1. Here, variable λ_n is the hidden "fused" state and is the variable of primary interest in this model. x_n^0 is an intermediate hidden state tracked by the set of point trackers which are represented by hidden variables $x_n^f, f = 1 : F$ and ξ_n is the hidden state tracked by a color based particle filter of [Perez et al., 2002].

$X_n = \left\{\lambda_n, x_n^0, x_n^1, \ldots, x_n^F, \xi_n\right\}$ is the joint hidden state and $Y_{1:n} = \left\{y_{1:n}^1, \ldots, y_{1:n}^F, y_{1:n}^\xi\right\}$

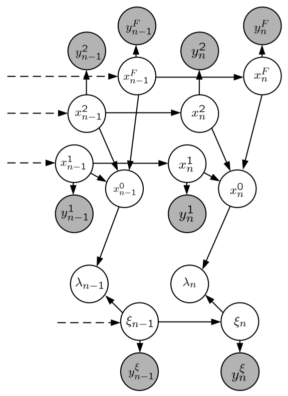


Figure 5.1: Graphical model for multi-cue fusion. λ_n is the *fused* hidden state.

is the collection of associated sequential measurements (data). The filtering distribution of the joint hidden state is factorised in the standard manner as shown below.

$$p(X_n|Y_{1:n};\Phi_n) \propto l(Y_n|X_n) \int \Psi(X_n|X_{n-1};\Phi_n) p(X_{n-1}|Y_{1:n-1};\Phi_{n-1}) dX_{n-1}, \quad (5.2.1)$$

with external parameter $\Phi_n = \{\alpha_n^0, \alpha_n^1, \ldots, \alpha_n^F, \alpha_n^\xi\}$. These parameters control message switching or message combinations to determine the filtering distribution of λ_n. $\Psi(.)$ is a *compatibility function* defined as follows.

$$\Psi(X_n|X_{n-1};\Phi_n) \triangleq \left[\alpha_n^0 \Psi_{\lambda,0}(\lambda_n|x_n^0) + \alpha_n^\xi \Psi_{\lambda,\xi}(\lambda_n|\xi_n)\right] \times \left[\sum_{f=1}^{F} \alpha_n^f \Psi_{0,f}(x_n^0|x_n^f) p(x_n^f|x_{n-1}^f) \prod_{j=1, j\neq f}^{F} p(x_n^j|x_{n-1}^j)\right] p(\xi_n|\xi_{n-1}). \quad (5.2.2)$$

Upon recalling the fact that each point tracker can be treated as a Bayesian filter (see Sec.4.2), the posterior of the "fused" hidden state λ_n can be written using message notation as follows.

$$p\left(\lambda_n|Y_n, y^{\xi}_{1:n-1}; \Phi_n, \left\{q^f_{n-1}\right\}_{f=1:F}\right) \propto \alpha^0_n \left[\sum_{f=1}^{F} \alpha^f_n m_{f\to\lambda}(\lambda_n|q^f_{n-1})\right] + \alpha^{\xi}_n m_{\xi\to\lambda}(\lambda_n), \tag{5.2.3}$$

where,

$$m_{f\to\lambda}(\lambda_n|q^f_{n-1}) = \frac{\int \Psi_{\lambda,0}(\lambda_n|x^0_n)\Psi_{0,f}(x^0_n|x^f_n)l_f(y^f_n|x^f_n)p(x^f_n|q^f_{n-1})dx^0_n dx^f_n}{\int l_f(y^f_n|x^f_n)p(x^f_n|q^f_{n-1})dx^f_n}, f = 1:F, \tag{5.2.4}$$

where $q^f_{n-1}, f = 1:F$ are parameters defined as in Eqn. 4.2.4, and,

$$m_{\xi\to\lambda}(\lambda_n) = \frac{\int \Psi_{\lambda,\xi}(\lambda_n|\xi_n)l_{\xi}(y^{\xi}_n|x^{\xi}_n)p(\xi_n|\xi_{n-1})p(\xi_{n-1}|y^{\xi}_{1:n-1})d\xi_{n-1}d\xi_n}{\int l_{\xi}(y^{\xi}_n|x^{\xi}_n)p(\xi_n|\xi_{n-1})p(\xi_{n-1}|y^{\xi}_{1:n-1})d\xi_{n-1}d\xi_n}. \tag{5.2.5}$$

If the components of the compatibility function are delta compatibilities as shown below,

$$\begin{aligned} \Psi_{\lambda,0}(\lambda_n|x^0_n) &= \delta_{x^0_n}(\lambda_n), \\ \Psi_{\lambda,\xi}(\lambda_n|\xi_n) &= \delta_{\xi_n}(\lambda_n), \\ \Psi_{0,f}(x^0_n|x^f_n; \mathbf{t}_f) &= \delta_{x^f_n+\mathbf{t}_f}(x^0_n),\ f = 1:F, \end{aligned} \tag{5.2.6}$$

where δ is the Dirac delta and $\mathbf{t}_f, f = 1:F$ parameterize the compatibilities between the relevant hidden states, then the posterior in Eqn. 5.2.3 can be approximated via sample based approximations of the above messages as shown below.

$$\begin{aligned} p\left(\lambda_n|Y_n, y^{\xi}_{1:n-1}; \Phi_n, \left\{q^f_{n-1}\right\}_{f=1:F}\right) \propto& \alpha^0_n \sum_{f=1}^{F} \alpha^f_n \frac{\sum_{i\in 1\ldots N} l_f(y^f_n|x^{f,i}_n)\delta_{x^{f,i}_n+\mathbf{t}_f}(\lambda_n)}{\sum_{i\in 1\ldots N} l_f(y^f_n|x^{f,i}_n)} + \\ & \alpha^{\xi}_n \frac{\sum_{j\in 1\ldots M} l_{\xi}(y^{\xi}_n|\xi^j_n)\delta_{\xi^j_n}(\lambda_n)}{\sum_{j\in 1\ldots M} l_{\xi}(y^{\xi}_n|\xi^j_n)}. \end{aligned} \tag{5.2.7}$$

The remaining filtering distributions take the following familiar forms which facilitate their sample-set approximations.

$$p(x^f_n|y^f_n; q^f_{n-1}) \propto l_f(y^f_n|x^f_n)p(x^f_n|q^f_{n-1}),\ f = 1:F. \tag{5.2.8}$$

$$p\left(x^0_n|Y_n; \Phi_n, \left\{q^f_{n-1}\right\}_{f=1:F}\right) \propto \sum_{f=1}^{F} \alpha^f_n \frac{\int \Psi_{0,f}(x^0_n|x^f_n)l_f(y^f_n|x^f_n)p(x^f_n|q^f_{n-1})dx^f_n}{\int l_f(y^f_n|x^f_n)p(x^f_n|q^f_{n-1})dx^f_n}, \tag{5.2.9}$$

and,

$$p(\xi_n|y^\xi_{1:n}) \propto l_\xi(y^\xi_n|\xi_n)p(\xi_n|y^\xi_{1:n-1}). \tag{5.2.10}$$

The key note of interest in this model is this, *the multi-cue fusion model integrates two stochastic filters itself and not just their measurements*: a tracking filter based on a set of point trackers can be "tapped off" at hidden state x^0_n and another tracking filter based on color measurements can be got at hidden state ξ_n.
For the specific case where the parameters α^0_n, α^ξ_n are *binary* the filtering distribution of λ_n is composed from one of the two stochastic filters (see Eqn. 5.2.3). This filtering distribution is identical to either the filtering distribution of x^0_n (which is $\propto \sum_{f=1}^F \alpha^f_n m_{f\to\lambda}(x^0_n|q^f_{n-1})$) or the one of ξ_n (which is $\propto m_{\xi\to\lambda}(\lambda_n)$). This "switching" idea is used to construct a novel multi-cue tracker.

5.3 Multi-cue switch tracker

The preceeding section showed how to compose the posterior of the hidden state of interest from messages. The control of which messages to include and which others to discard lies with the external parameters. So the identity of the multi-cue tracker is defined by the manner in which these parameters are updated at each tracking instant. The following steps describe a method for their update.

5.3.1 Parameter update

For further consideration assume the filtering distributions in Eqns. 5.2.8, 5.2.10 are approximated by sample-sets $\left\{x^{f,j}_n, \pi^{f,j}_n\right\}_{j=0}^{j=K}, f = 1 : F$ and $\left\{\xi^j_n, \pi^{\xi,j}_n\right\}_{j=0}^{j=K}$ respectively. The update of binary parameters $\left\{\alpha^f_n,\ f = 1 : F\right\}$ follows from the discussions in Chapter 4 and so will not be repeated here.

α^0_n, α^ξ_n update:
These parameters control switching between color based measurements and measurements from the bunch of point trackers. At any instant both these measurements may be valid, one of the two may be valid or both could be inconsistent. Since this is a switching method one of the two measurements must be preferred or both rejected. One way of doing this is by assigning a *priority* to these measurements and developing a method by which this priority logic is reflected in the update of these parameters. Such a concept is elucidated in the following discussions.

If the number of point trackers is F and the number of inlier point trackers among them at instant n is $\mathcal{I}_n$ (see Sec. 4.4.2 for a method to determine inliers), then define,

$$b_n^0 \triangleq \begin{cases} 1, \text{if } \mathcal{I}_n \geq \begin{cases} \frac{F}{2}, \text{if } F \text{ is even.} \\ \frac{F+1}{2}, \text{if } F \text{ is odd.} \end{cases} \\ 0, \text{otherwise.} \end{cases} \quad . \tag{5.3.1}$$

Probabilistic rejection of color cue:
Let C_n^π denote the covariance of the sample-set $\left\{\xi_n^j, \pi_n^{\xi,j}\right\}_{j=0}^{j=K}$ and let C_n^s denote the covariance of $\left\{\xi_n^j, \frac{1}{K}\right\}_{j=0}^{j=K}$. Compute,

$$p_n^r = \min\left\{1.0, \frac{\det[\mathrm{C_n^\pi}]}{\det[\mathrm{C_n^s}]}\right\}. \tag{5.3.2}$$

p_n^r tends to 1 as the posterior becomes more diffuse and tends towards 0 as the posterior becomes more peaked. Seen another way, this ratio provides a scalar indicator of the uniformity of the filtering distribution of the color based particle filter.
With this, define,

$$b_n^\xi \triangleq \begin{cases} 1, \text{if } p_n^r \leq \tau \\ 0, \text{if } p_n^r > \tau, \end{cases} \tag{5.3.3}$$

where τ is an empirical threshold. Now the logic structure presented in Figs. 5.2, 5.3 is used to update the parameters. The motivation behind this logic is tied with the priority logic alluded to earlier. This is discussed below.

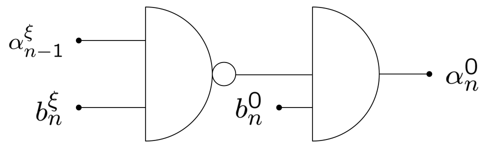


Figure 5.2: Logic for α_n^0 update.

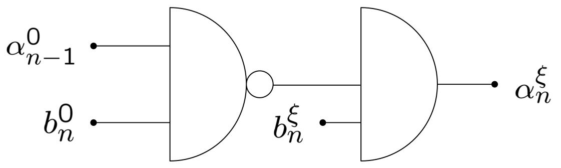


Figure 5.3: Logic for α_n^ξ update.

5.3.2 Priority switching and reference model adaptation

The form of parameter update in Figs. 5.2, 5.3 captures the notion of *priority switching*. Here the joint state $\alpha_0^0 = 1, \alpha_0^\xi = 1$ is disallowed, which implies only one of two cues is "ON" at the initialisation stage. To understand the notion of priority switching, consider the joint state $\alpha_{n-1}^0 = 1, \alpha_{n-1}^\xi = 0$, which is to be read as the RFT-filter (see Sec. 4.4) is "prioritised" over the color based filter. From the truth tables 5.1, 5.2 it can be seen that if $b_n^0 = 1$, then irrespective of the state of b_n^ξ it holds that

α^{ξ}_{n-1}	b^{ξ}_{n}	b^{0}_{n}	α^{0}_{n}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Table 5.1: Truth table for α^0_n update.

α^{0}_{n-1}	b^{0}_{n}	b^{ξ}_{n}	α^{ξ}_{n}
0	0	0	0
0	1	0	0
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	0
1	0	1	1
1	1	1	0

Table 5.2: Truth table for α^{ξ}_n update.

$\alpha^0_n = 1, \alpha^{\xi}_n = 0$. This ensures $\alpha^0_n = 1$ or equivalently, it retains its priority, which implies that the λ_n continues to receive messages from the point trackers even though the color measurements are declared consistent. Now, alternatively, if $\alpha^0_{n-1} = 1, \alpha^{\xi}_{n-1} = 0$ and $b^0_n = 0, b^{\xi}_n = 1$, then $\alpha^0_n = 0, \alpha^{\xi}_n = 1$, which is a "switch" in the priority to the color based filter and so on.

If $b^0_n = 0, b^{\xi}_n = 0$, then $\alpha^0_n = 0, \alpha^{\xi}_n = 0$, irrespective of which filter held priority. This results in a *degenerate posterior* for λ_n. Strong occlusions could possibly generate such an event. From this state on, consider the event $b^0_n = 1, b^{\xi}_n = 1$, which results in $\alpha^0_n = 1, \alpha^{\xi}_n = 1$. In practice, it is rare that both filters successfully relock onto the target (or something very similar to it) at the same instant. If it so happens then one of the filters should be artificially prioritised before tracking is continued.

The switch in priority between filters should be taken as an opportunity to perform target reference model adaptation. In this context, the following three possible courses need to be considered.

$\alpha^0_{n-1} = 0, \alpha^{\xi}_{n-1} = 1 \longrightarrow \alpha^0_n = 1, \alpha^{\xi}_n = 0$:

This is a case when measurements from the set of point trackers are deemed inconsistent. At such an event, the entire set of point trackers and point templates are discarded and a new set of points (meaning point trackers) are sampled from the belief of λ_n. Each of these sampled points is associated to its point template, which is a small rectangular patch centered around the point. The notion behind discarding point trackers and sampling new ones is simple to understand. Since the belief λ_n is composed only of the message sent from the hidden state associated to the color cue, these new point trackers are then effectively sampled from the filtering distribution of ξ_n. So the change in the appearance model of the filter based on point trackers is "guided" by the color based filter.

$\alpha^0_{n-1} = 1, \alpha^{\xi}_{n-1} = 0 \longrightarrow \alpha^0_n = 0, \alpha^{\xi}_n = 1$:
This reverse transition is a sign of inconsistent measurements arising from the color cue. The color histogram is chosen to remain unchanged, as this is taken as the *anchor* component of the entire set appearance model of the target. Therefore, as a corrective measure the particle filtering process is restarted with the filtering distribution of λ_n as the prior distribution. With this correction, the filtering equation now reads as follows.

$$p\big(\xi_{n+1}|Y_{1:n+1}\big) \propto l_{\xi}\big(y^{\xi}_{n+1}|\xi_{n+1}\big)p\big(\xi_{n+1}\big), \tag{5.3.4}$$

where,

$$p\big(\xi_{n+1}\big) \equiv \delta_{\lambda_n}\big(\xi_{n+1}\big)p\big(\lambda_n|Y_{1:n};\Phi_n\big). \tag{5.3.5}$$

The posterior of λ_n, which is composed by the message sent from the set of point trackers, is made to play the role of a proposal density. Thereby the particles associated with the color based filter are *channelled* through the posterior of λ_n.

$\alpha^0_{n-1} = 0, \alpha^{\xi}_{n-1} = 1$ or $\alpha^0_{n-1} = 1, \alpha^{\xi}_{n-1} = 0 \longrightarrow \alpha^0_n = 0, \alpha^{\xi}_n = 0$:
This setting of parameter values makes the posterior of λ_n degenerate, as both color measurements and set of point tracker measurements are unreliable. Such an event can result from some form of occlusion. In this case, it is clear that no reference model update is advisable until at least one of the two cues is resurrected. It is easy to understand, from the fact that the holistic color model is more stable in time than detailed point templates, that the color based filter is more potent to relock onto the target at the end of an occlusion. In order to aid this relock, sequential importance resampling (SIR) procedure used in the color based filter is temporarily arrested (see [Arulampalam et al., 2002]. This broadens the spatial spread of the importance samples of with a hope that some samples will "hit" the reappearing target. Assuming there is no clutter during such an event, if some samples do hit the target, then a return to consistency of the color measurements will be reflected by a distinct lowering of the rejection probability (see Sec. 5.3.1).
For the sake of stability, replacement of outlier point trackers is arrested during the event of an occlusion. If and once the target is relocked using the color based filter, new point trackers can once again be brought into play by sampling as discussed earlier.

Thus occlusions are effectively handled in this multi-cue setup.

No switch in priority

If the priorities are not switched from an instant to the the next and both filters are deemed consistent, then outlier point trackers are replaced by draws from the posterior of λ_n (or the posterior of the median-aged point tracker as discussed in Chapter 4). Similarly, importance samples for the color based particle filter can be drawn from its effective prior or even from the posterior of λ_n as a way to keep the two filters tied together (this technique is employed in the construction of the multi-cue switch tracker). If importance samples are drawn from the posterior of λ_n, then such an approach is equivalent to restarting the color based particle filter as described earlier. At this juncture, the key issue of reference model adaptation need to be reemphasized.

Set appearance model: The target model employed in this tracker is a discrete set appearance model. A subset of this set is composed of the gray level templates. Due to point tracker replacements, at any tracking instant, it is likely that the age of each element (template) in this subset is possibly different from the rest. Therefore, this subset consists of templates having lifespans of a few to several tens of frames and thus plays the role of the *dynamically adapted* part of the entire target model.

The second subset is the constant color reference histogram of the target. This is the *static* part of the two part target appearance model and does not interact with the first subset. The histogram is deliberately kept constant to avoid false adaptation due to illumination, orientation and size changes.

It is interesting to note that this two part model bears similarity to the scheme proposed by Fleet et. al [Jepson et al., 2003]. Their scheme involves explicit construction of a fused object reference model with static, slowly varying and fast varying components. In this scheme, the extreme static part is the color histogram and the other two components can be mapped to the point tracker appearance templates of varying ages. All the discussions on multi-cue switch tracking is recapitulated in the pseudo-code presented in Algorithm 2.

The following section proceeds to set up the experiments to test the multi-cue fusion tracker.

Algorithm 2: Multi-cue switch tracking

```
input    : Video sequence with $L$ frames and target bounding box in first frame;
Initialization;
  $\lambda_0$ = Center of bounding box;
  Color based particle filter (CPF): Draw particle, set reference color histogram and state
  space model parameters;
  RFT-filter: Draw point trackers and set point tracking parameters;
  Assign priority $\alpha_1^0, \alpha_1^\xi$ by user choice;
  if $\alpha_1^0 = 1, \alpha_1^\xi = 0$ then RFT-filter is prioritised;
  else if $\alpha_1^0 = 0, \alpha_1^\xi = 1$ then CPF is prioritised;
  for $n \leftarrow 2$ to $L$ do
      $b_n^0 \leftarrow$ RFTFiltering ();
      $b_n^\xi \leftarrow$ CPF ();
      $\{\alpha_n^0, \alpha_n^\xi\} \leftarrow$ ParameterUpdate $\left(\alpha_{n-1}^\xi, \alpha_{n-1}^0, b_n^\xi, b_n^0\right)$; /* See sec.  5.3.1 */
      $p_{\lambda_n} \equiv p\left(\lambda_n | Y_n, y_{1:n-1}^\xi; \Phi_n, \left\{q_{n-1}^f\right\}_{f=1:F}\right) \leftarrow$ ComposePosterior (); /*See sec.  5.3*/
      if $p_{\lambda_n}! = 0$ then
          output   : Bounding box centered on $\mathbf{E}[p_{\lambda_n}]$;
      end
      else
          output   : Bounding box centered on the average value of the posterior of the last
                     registered priority filter;
      end
      /*See sec.  5.3.2*/
      if $\alpha_n^0 = 1, \alpha_n^\xi = 0$ then
          ReplaceOutlierPointTrackers ($p_{\lambda_n}$);
          if $\alpha_{n-1}^0 = 0, \alpha_{n-1}^\xi = 1$ then
              RestartCPF ($p_{\lambda_n}$);
          end
          else
              ResampleCPF ();
          end
      end
      else if $\alpha_n^0 = 0, \alpha_n^\xi = 1$ then
          if $\alpha_{n-1}^0 = 1, \alpha_{n-1}^\xi = 0$ then
              ReplaceAllPointTrackers ($p_{\lambda_n}$);
          end
          else
              ReplaceOutlierPointTrackers ($p_{\lambda_n}$);
          end
          ResampleCPF ();
      end
      else if $\alpha_n^0 = 0, \alpha_n^\xi = 0$ then
          Arrest replacement of point trackers and resampling of particles;
      end
      else if $\alpha_n^0 = 1, \alpha_n^\xi = 1$ then
          Artificially prioritise one of the two filters and perform necessary
          replacement/resampling/restart according to assigned priority;
      end
  end
```

5.4 Experimental setup

The tracker was tested on several ground-truthed color sequences provided by the CMU VIVID-PETS project [Collins et al., 2005]. The sequences were all aerial videos of moving cars or military vehicles. These sequences provide wide variety of challenges like small sized targets with variable motions, defocus blurs, scale changes due to camera zoom variations and target pose changes, extreme illumination changes due to sunlight glare and partial to complete occlusions. The project provides an online third party evaluation system to analyse the performance of the algorithm against several evaluation parameters, primary of which is the percentage of total frames tracked. Other parameters like shape matches are beyond the scope of these experiments and so not discussed here. The evaluation system also provides comparative results against several fundamental/state of the art trackers. The results of this evaluation are produced in Table 5.3 for all the test sequences. If functional, the statistics produced in this table can also be found online at the CMU-VIVID project website: *http://www.videvaluation.ri.cmu.edu/main.html*.

Apart from the VIVID database sequences the standard Jepson and Fleet sequence [Jepson et al., 2003] is also used in the tests. Other long and very challenging sequences such as the "Snake-eyes" sequence and the "Children of men" sequence are introduced. These sequences are rare long take sequences from cinema consisting of drastic illumination variations, shadow effects, multiple occlusions, rotations in depth, camera jerks and scaling. These sequences have not been ground-truthed and so only qualitatively evaluated.

All tracking runs were manually initialised using a bounding box. The parameters for the point trackers and the corresponding RFT-filter were retained from the experiments performed in Chapter 4. Temporary modifications, if any, are highlighted in the result samples. 200 samples and a first order Markovian kinematic prior with variance proportional to the dimension of the target was used for the color based filter. The rejection probability threshold τ was empirically set at 0.9, meaning $b_2 = 0$, if the computed rejection probability exceeded this threshold. It is prescribed that this threshold be varied roughly in inverse proportionality to the standard deviation of the kinematic prior.

The RGB space reference color histogram model is a $8 \times 8 \times 8$ bin histogram. The population of each bin is incremented if the quantized RGB value of a *target pixel* falls into the range represented by that bin. The experimentalist is also free to explore

histograms constructed on more complex color spaces like the HSV space [Perez et al., 2002].
At each time instant, a constant size bounding box (scale is not estimated in the proposed approach) centered on the $\mathbf{E}\left[p\left(\lambda_n|Y_n, y^{\xi}_{1:n-1}; \Phi_n, \left\{q^f_{n-1}\right\}_{f=1:F}\right)\right]$ is displayed. This box is for display purposes only and it must be borne in mind that a full posterior distribution describing the uncertainty in this position estimate is available.

5.5 Results and discussions

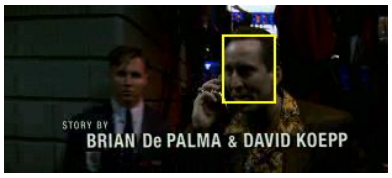
The results of multi-cue switch tracking of human heads and vehicles are discussed below.

Head tracking

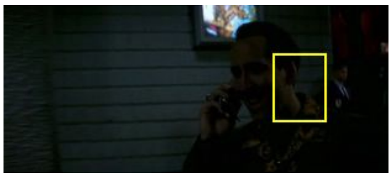
As a means to compare multi-cue and single cue tracking, snapshots of color based particle filtering [Perez et al., 2002] on the 1000 frame Snake-eyes sequence is presented in Fig.5.4. The instability of the color tracker under strong illumination changes, Figs.5.4(b),5.4(c) and clutter, Figs.5.4(g),5.4(h), 5.4(m), 5.4(n), 5.4(o), 5.4(p), 5.4(q) can be observed here. On the other hand, its ability to relock onto the target after occlusions (of about 5-6 frames) can be seen in Figs.5.4(f),5.4(l). Overall, the filter wavers from the correct track at various periods and loses track completely near the end of the sequence. For the sake of comparison, the reader is referred to the results of head tracking on this sequence using the RFT-filter developed in Sec.4.4. The track, although smooth is quite short and ends at the first occlusion itself. It is clear that neither one of these filters is capable of tracking the head consistently over the entire length of the sequence. In contrast, Fig. 5.5 strives to demonstrate the clear advantage of bringing together these two filters over employing individual filters.

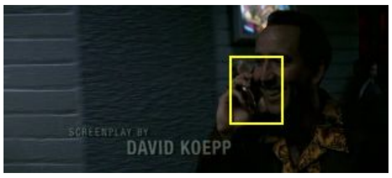
The strong changes in *illumination* at different periods in this sequence are handled by the RFT-filter, see Figs. 5.5(27,31,32) for an example. The posterior in Fig.5.5(6) is more diffuse than in Fig. 5.5(5) as the object undergoes a *pose change*. Similar observations can be made from samples of image, posterior pairs throughout the result samples in Figs.5.5(9,12), Figs.5.5(13,16), Figs.5.5(14,17), Figs.5.5(27,30), Figs.5.5(31,34), Figs.5.5(32,35), Figs.5.5(33,36), Figs.5.5(37,40) and Figs.5.5(39,42). Two instances where the color based filter guides the RFT-filter are shown in Figs.5.5(15,38).

Effective *recovery from occlusion* can be seen Figs.5.5(19,20,21). The spread of the

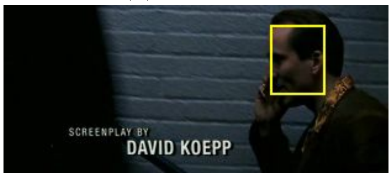


(a). Frame 1

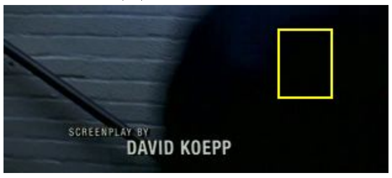

(b). Frame 99



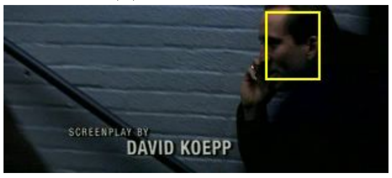
(c). Frame 110



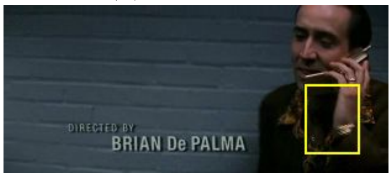
(d). Frame 169



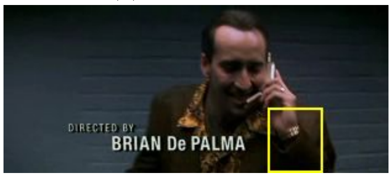
(e). Frame 175



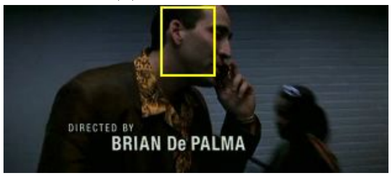
(f). Frame 177



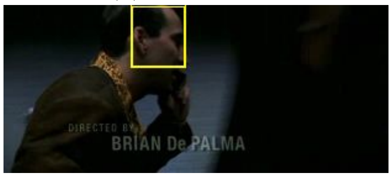
(g). Frame 233



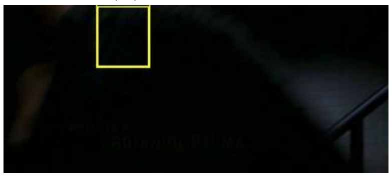
(h). Frame 257

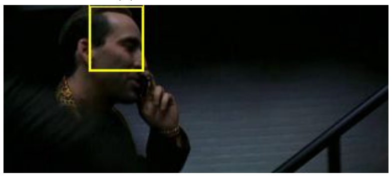


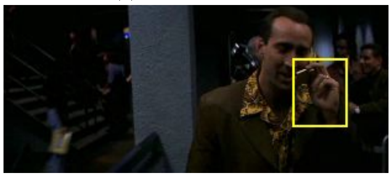
(i). Frame 304

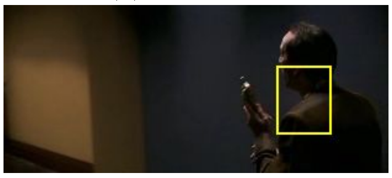


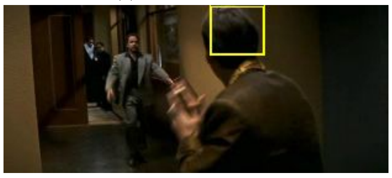
(j). Frame 319

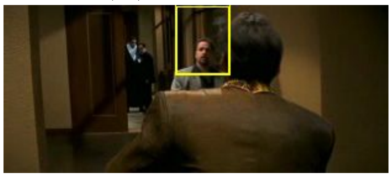

(k). Frame 326

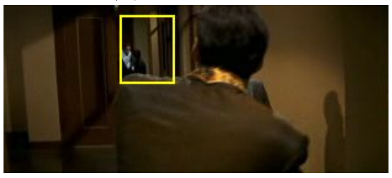

(l). Frame 329


(m). Frame 628


(n). Frame 792


(o). Frame 840


(p). Frame 858


(q). Frame 871

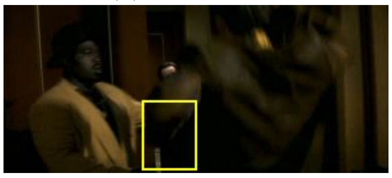
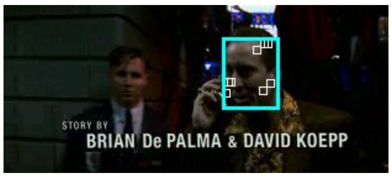
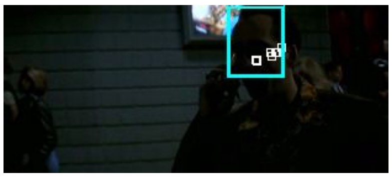
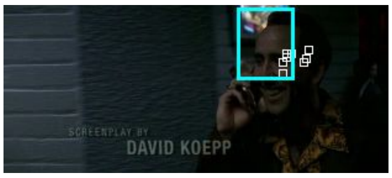

(r). Frame 999

Figure 5.4: Color based particle filter based head tracking in the Snake-eyes sequence using 200 particles.



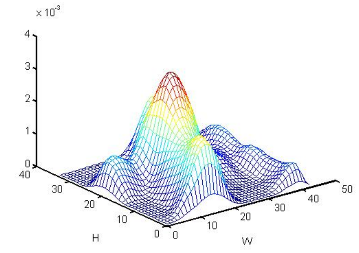
(1). Frame 1


(2). Frame 92

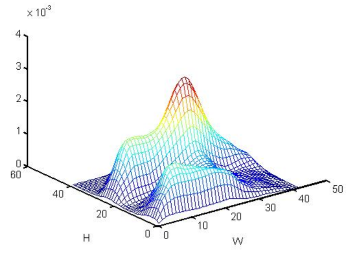


(3). Frame 110

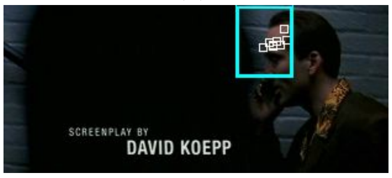
(4).



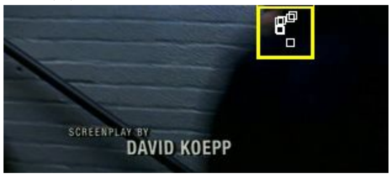
(5). Posterior at frame 92



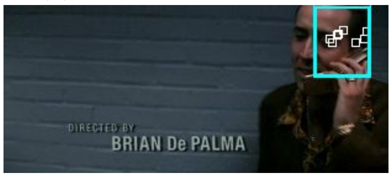
(6). Posterior at frame 110



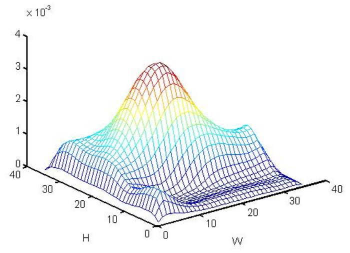
(7). Frame 172



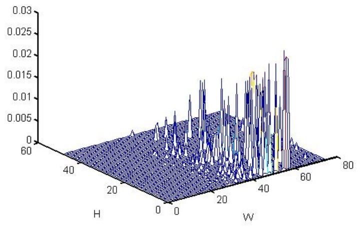
(8). Frame 176



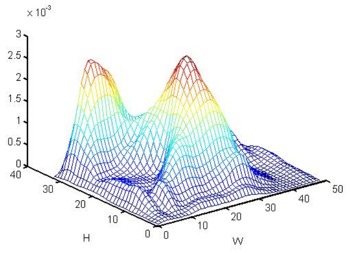
(9). Frame 233



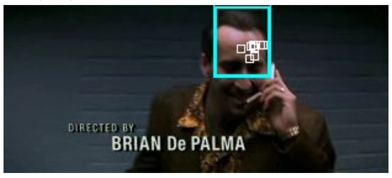
(10). Posterior at frame 172



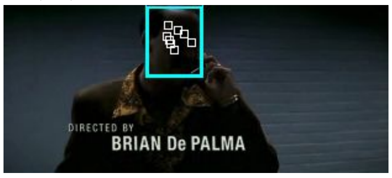
(11). Posterior at frame 176



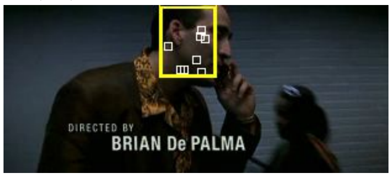
(12). Posterior at frame 233



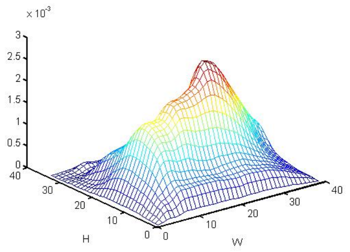
(13). Frame 257



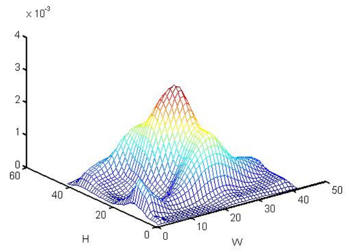
(14). Frame 281



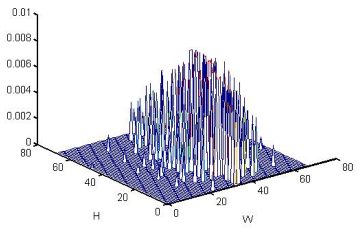
(15). Frame 304



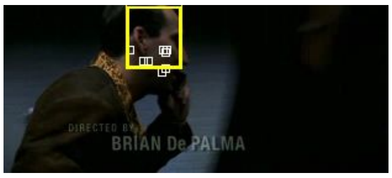
(16). Posterior at frame 257



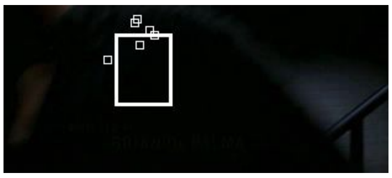
(17). Posterior at frame 281

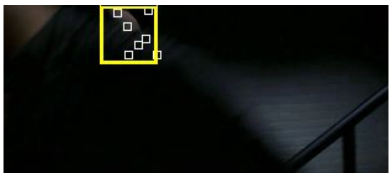


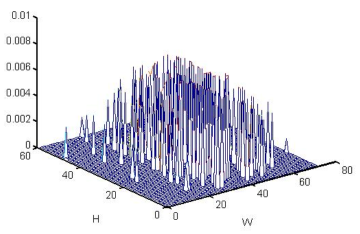
(18). Posterior at frame 304



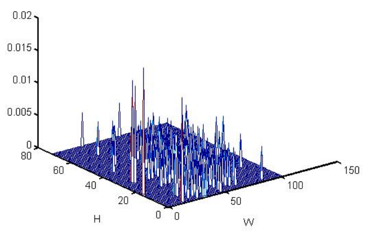
(19). Frame 319


(20). Frame 326

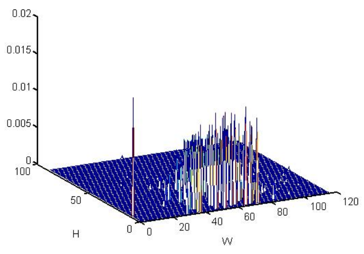

(21). Frame 327



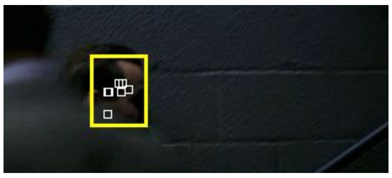
(22). Posterior at frame 319

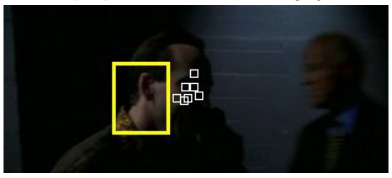


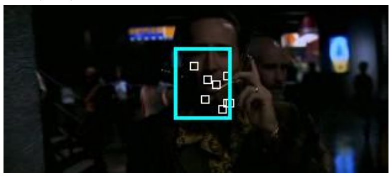
(23). Posterior of x^1_{326}

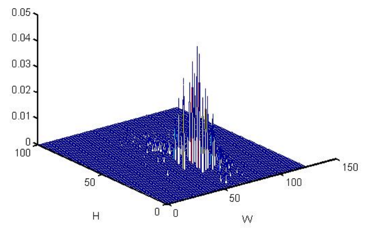


(24). Posterior at frame 327

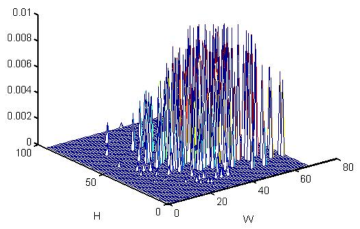

(25). Frame 411


(26). Frame 485

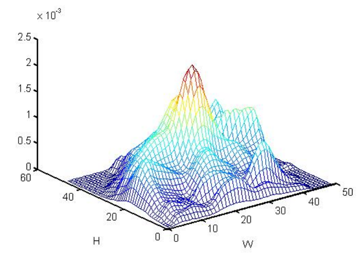

(27). Frame 559



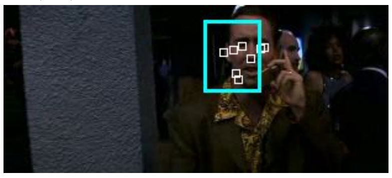
(28). Posterior at frame 411

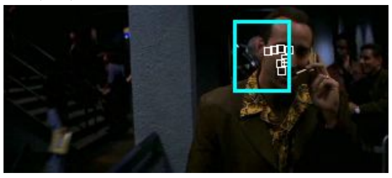


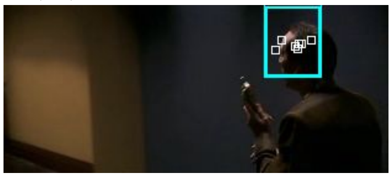
(29). Posterior at frame 485

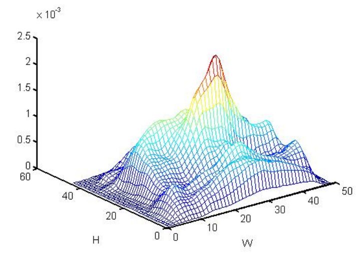


(30). Posterior at frame 559

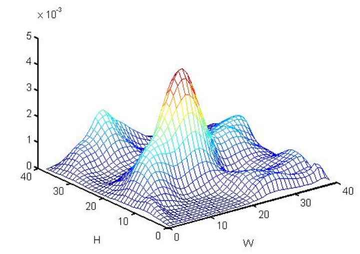

(31). Frame 588


(32). Frame 628

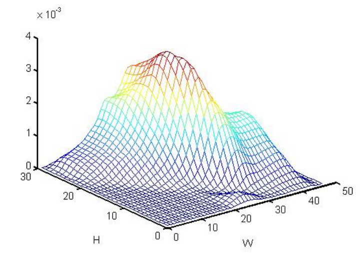

(33). Frame 792



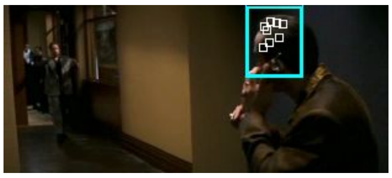
(34). Posterior at frame 588

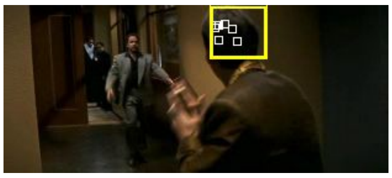


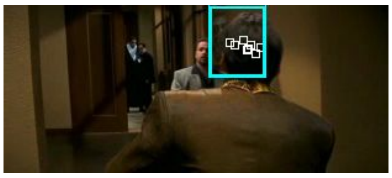
(35). Posterior at frame 628

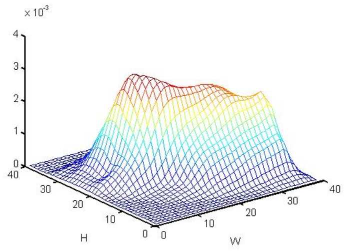


(36). Posterior at frame 792

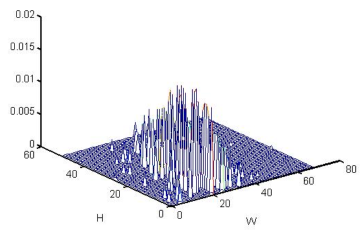

(37). Frame 818


(38). Frame 840

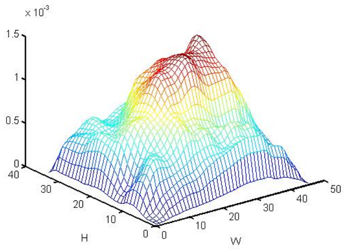

(39). Frame 858



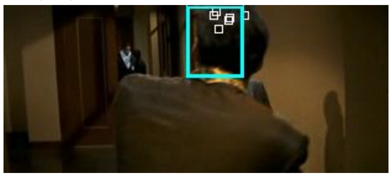
(40). Posterior at frame 818

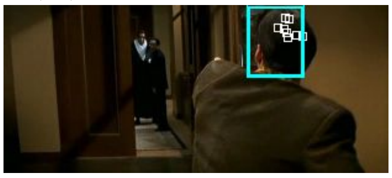


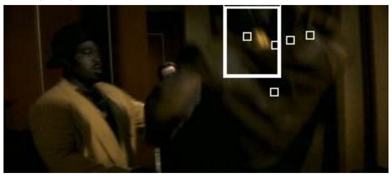
(41). Posterior at frame 840

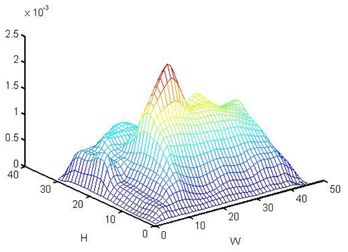


(42). Posterior at frame 858

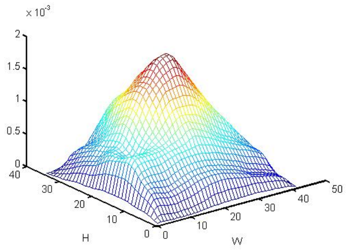

(43). Frame 871


(44). Frame 944


(45). Frame 999



(46). Posterior at frame 871



(47). Posterior at frame 944

(48). Degenerate posterior at frame 999

Figure 5.5: Multi-cue switch tracking on the Snakeeyes sequence. Priority switching is demonstrated by changing the color of the bounding box. Blue boxes signify the priority lies with the set of point trackers and yellow boxes indicate priority for the color based particle filter. White boxes signify degenerate posteriors. The priority switches 25 times in this result sample.

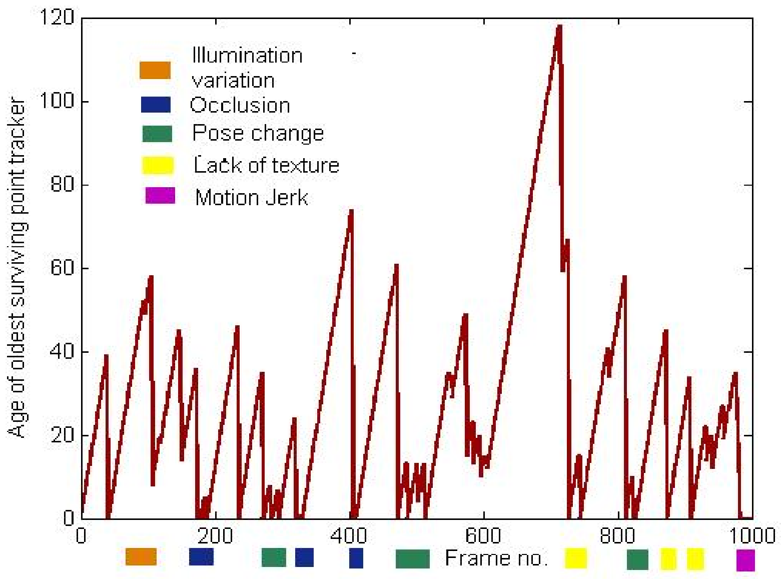


Figure 5.6: A plot of the age of the oldest point template versus the frame number for the snakeeyes sequence in Fig.5.5. Loss of point trackers to several events in the sequence is analysed using color bands.

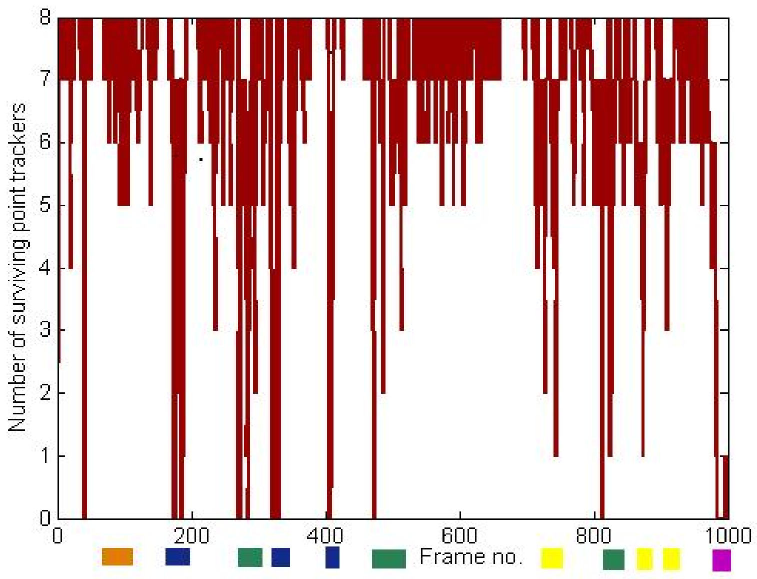


Figure 5.7: A plot of the number of templates retained at each frame versus the frame number for the snake eyes sequence in Fig.5.5. The color bands have the same interpretation as in Fig. 5.6.

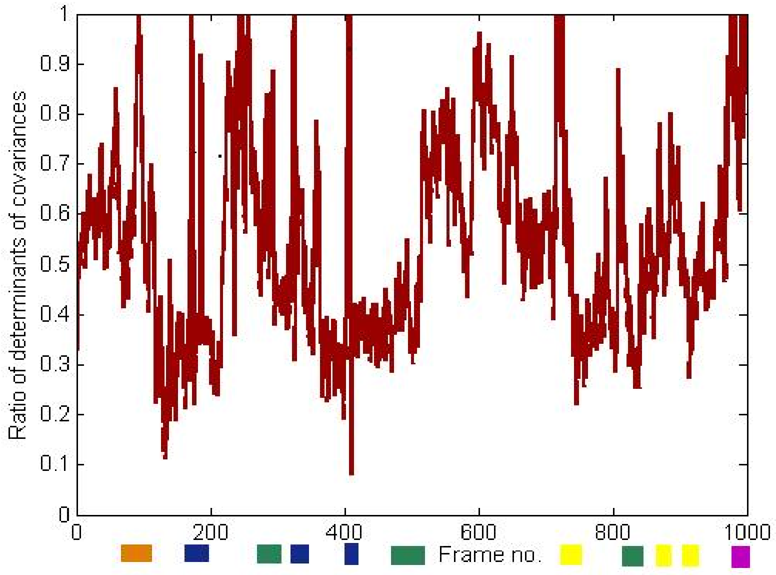


Figure 5.8: A plot of the ratio of covariance determinants versus the frame number for the snakeeyes sequence in Fig.5.5. The color bands have the same interpretation as in Fig. 5.6.

samples increases under occlusion, see Fig.5.5(23), and decreases when the object begins to reappear, see Fig.5.5(24). The reader must bear in mind that the posterior is effectively *degenerate* during occlusion (Fig.5.5(20)), meaning $p\big(\lambda_n|Y_{1:n};\Phi_n\big) \equiv 0$. However the posterior of the color based particle filter is deliberately shown instead to draw comparisons with the results of the next frame where the target is relocked by this filter. A similar observation can be made after the target reappears after the fourth and final occlusion, see the posterior in Fig.5.5(28).

A noticeable *drift* in the estimate of the target location can be seen at several instances (due to a lack of adaptation of the compatibility functions), see Figs.5.5(3,7,9) and Figs.5.5(13,32,33). Fig.5.5(26) presents a situation wherein the method misguided by a *shadow effect*, hands over the priority to the color based particle filter. The tracker however recovers soon enough from this event, see Fig.5.5(27). The distress caused to the RFT-filter due to a *lack of texture* can be clearly observed from the diffuse posteriors in pairs Figs.5.5(39,42), Figs.5.5(43,46) and Figs.5.5(44,47).

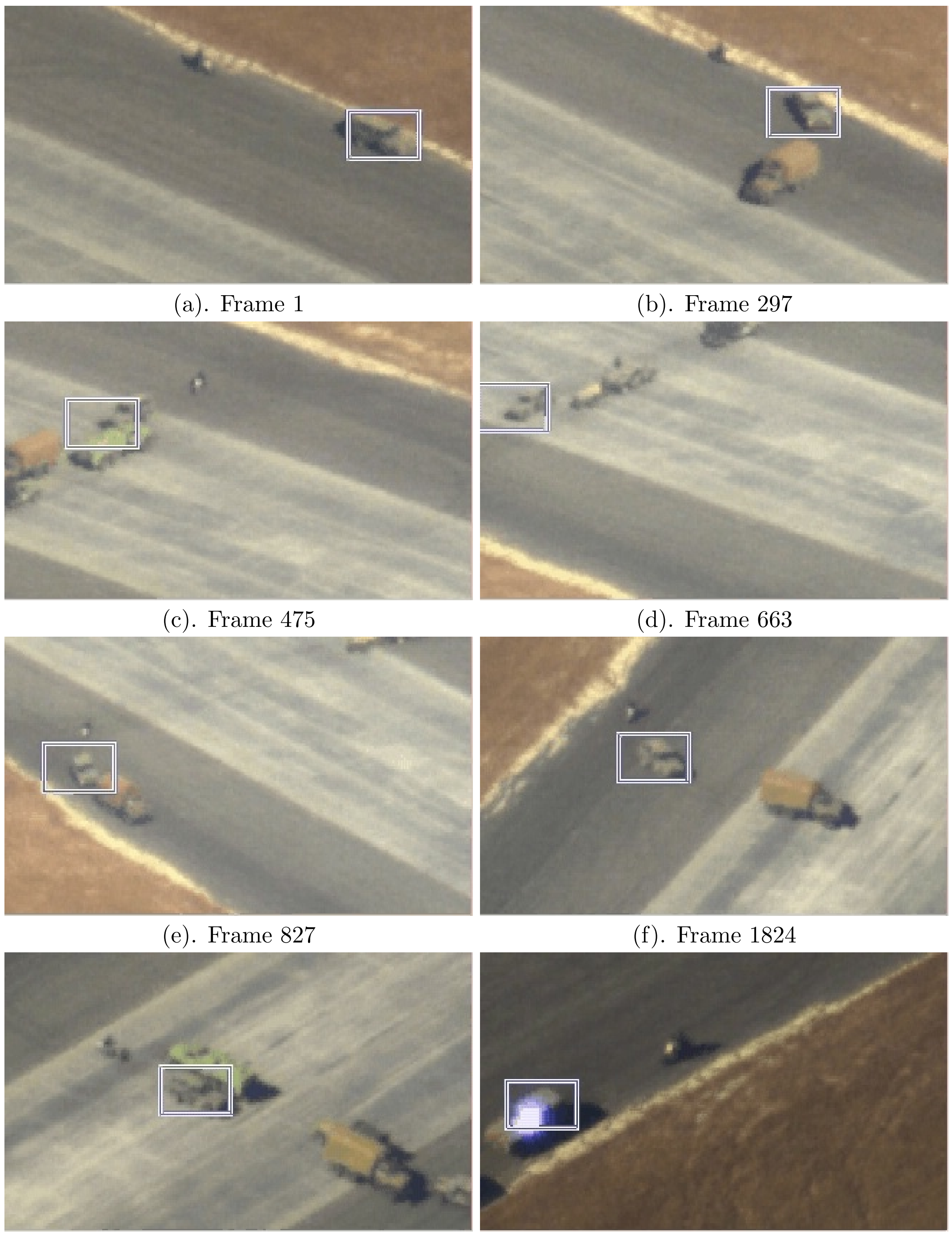
The mutually beneficial effect of multi-cue tracking is particularly visible near the end of the video sequence when the target undergoes a 180° pose change. The RFT-filter guides the color based particle filter in Figs.5.5(33,37), while the color based filter guides the RFT-filter through a drastic pose change in Fig.—5.5(38), thus enabling tracking till nearly the very end of the sequence. The tracker is "shaken-off" at the very end due to a strong *motion jerk* coupled with very low illumination. A full view of the results may be gathered from the video provided in the supplementary material.

To corroborate the facts discussed so far, *event annotated plots* of the age, in frame units, of the oldest (most consistent) point tracker at each instant, Fig.5.6, the number of point trackers retained after each execution of rejection control, Fig.5.7, and the evolution of the ratio of determinants (proportional to rejection probability p_r), Fig.5.8 are provided. As expected, occlusions and large pose changes eliminate a large number of point trackers. On the other hand, these NCC point trackers are resilient to variations in illumination. The average most consistent point tracker lasts between $30 - 40$ frames. Lack of texture leads to elimination of point trackers, but sampling from the posterior of the λ_n replenishes them, which is another evidence of beneficial interactions.
The ratio of the determinants of covariance matrices is close to 1.0 when the illumination changes, see Figs.5.5(2,3,14), during occlusions, see Figs.5.5(7) and Fig.5.5(8) for an example and often in the presence of distractive clutter, see Fig.5.5(13). In contrast, the color based particle filter is fairly resilient to pose changes and lack of texture, see Figs.5.5(15,38).

Vehicle tracking

Table 5.3 presents the results of tracking vehicles in aerial videos provided by the CMU-VIVID project. The results of this automatic evaluation system indicate that the multi-cue tracker outperforms most state of art trackers and in cases where it does not, it only lags by a negligible percentage. In these tests, the dimensions of the point tracker appearance templates and their search spaces were hand tuned at the start to accomodate for the variation in target sizes and motion magnitudes.
Fig. 5.9 shows result samples of tracking one of the sequences from the VIVID project. The multi-cue tracker overcomes challenges of pose changes, clutter and strong glares in this 2300 frame sequence. The search space for point tracking was increased to 81×81 to accomodate camera jerks. All other parameters were the same as for the head tracking test. All contesting trackers meet failure due to very long occlusions presented in test sequences EgTest04 and EgTest05.

The table also compares the performance of the multi-cue switch tracker when the initial priority is alternated between the two filters, that is, between the two configurations $\alpha_0^0 = 1, \alpha_0^\xi = 0$ and $\alpha_0^0 = 0, \alpha_0^\xi = 1$ at initialisation frame. It can be observed that



(a). Frame 1 (b). Frame 297

(c). Frame 475 (d). Frame 663

(e). Frame 827 (f). Frame 1824

(g). Frame 2012 (h). Frame 2455

Figure 5.9: Multi-cue switch tracking of vehicles in aerial video sequences. Courtesy CMU-VIVID database.

	Test sequences and % of total frames tracked in each							
Candidate Algorithms	Egtest01	Egtest02	Egtest03	Egtest04	Egtest05	RedTeam	Snakeeyes	Fleet
Proposed Algorithm (C†)	100.0%	100.0%	100.0%	39.89%	13.64%	100.0%	98.7%	100.0%
Color proportion*	99.4%	19.9%	11.1%	99.6%	60%	27.5%	18%	48%
Proposed Algorithm (T†)	100.0% ()	100.0%	100.0%	39.89%	13.64%	100.0%	98.7%	100%
Template proportion*	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	78%	100%
PeakDiff	100.0%	30.77%	12.06%	3.83%	13.64%	98.43%		
Mean Shift	17.58%	39.23%	20.62%	9.84%	13.64%	84.29%		
FgBg Ratio	100.0%	39.23%	17.90%	8.74%	13.64%	100.0%		
Adaptive Shape Color	100.0%	100.0%	20.23%	9.84%		100.0%		
Variance Ratio	29.12%	27.69%	12.06%	9.84%	13.64%	100.0%		
Particle Filter	99.45%			40.44%		100.0%		
Template Match	87.91%	21.54%	12.84%	2.73%	17.61%	16.23%		
GraphCut	2.20%	15.38%	3.11%	0.55%	2.27%	34.03%		

Table 5.3: The figures indicate the percentage of total frames tracked in each sequence. C†, T† indicates the initial priority was assigned to the color based particle filter and RFT-filter respectively. * indicates the proportion of frames for which the filter was used for composing the posterior. The blank boxes indicate non availability of results. Courtesy: VIVID Evaluation Collins et al. [2005].

this does not affect the overall results. However, in the VIVID project sequences, when the color based particle filter is assigned the initial priority then there is at least one switch, which is not the case when the RFT-filter is assigned the priority. This brings out the strength of the RFT-filter as a useful tracker, but the reader must bear in mind that since the multi-cue tracker is stochastic, these results must only be interpreted in an average sense. In a particular test run multiple switches may occur. Apart from the vehicle sequences the table also contains results for the Snake-eyes sequence and the Jepson and Fleet sequence [Jepson et al., 2003]. In the Snake-eyes sequence, there is at least one switch irrespective of the initial priority. To conclude, the fact that the overall performance on any test sequence does not change when the initial priority is alternated confirms its insensitivity to the initial configuration of the parameters and the robustness that is achieved due to multi-cue interactions.

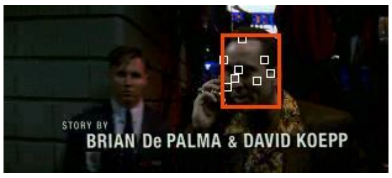
5.6 Multi-cue combination tracker

The logic structure used to construct the multi-cue switch tracker disallowed the state $\alpha_n^0 = 1, \alpha_n^\xi = 1$, meaning the posterior of λ_n was composed of only one of the two messages, even if both forms of measurements was deemed consistent. When two filters are involved, there is always the possibility of a *dichotomy* among the contesting hypotheses, that is, having two *non-overlapping hypotheses* for the target state. The multi-cue tracker resolves this dichotomy by virtue of its priority logic, simply, the prioritised filter composes the posterior of the target state. With such logic, constructive *information* from the non-prioritised filter could possibly be discarded. Note that this is only a consequence of the form of parameter update and not the graphical model itself. For instance, if $\alpha_n^0 = b_n^0, \alpha_n^\xi = b_n^\xi$ is taken to be the form of the parameter update, then another robust tracker can be derived from the model based on message combinations. This tracker is termed the multi-cue combination tracker.

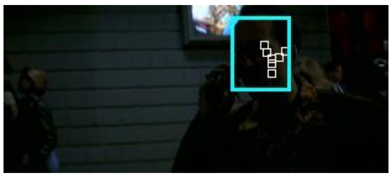
If the terms b_n^0, b_n^ξ represent the *relevance* (terminology employed in [Patras and Hancock, 2007]) of the cues then these can be directly ”plugged-in” to the model through the parameters α_n^0, α_n^ξ. In this way all permutations of messages for composing the filtering distribution of the target state are allowed. In particular, when both forms of measurements are deemed relevant then the filtering distribution is a *linear mixture* of the two messages, see Eqn .5.2.3. Apart from this fact, the multi-cue

Algorithm 3: Multi-cue combination tracking

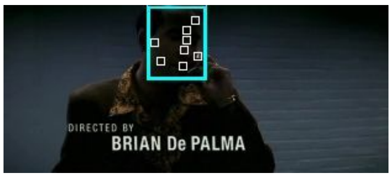
```
input    : Video sequence with $L$ frames and target bounding box in first frame;
Initialization;
  $\lambda_0$ = Center of bounding box;
  Color based particle filter (CPF): Draw particle, set reference color histogram and state
  space model parameters;
  RFT-filter: Draw point trackers and set point tracking parameters;
  for $n \leftarrow 2$ to $L$ do
      $b_n^0 \leftarrow$ RFTFiltering ( );
      $b_n^\xi \leftarrow$ CPF ( );
      $\left\{\alpha_n^0, \alpha_n^\xi\right\} \leftarrow$ ParameterUpdate $\left(\alpha_{n-1}^\xi, \alpha_{n-1}^0, b_n^\xi, b_n^0\right)$; /* See sec. 5.3.1 */
      $p_{\lambda_n} \equiv p\left(\lambda_n | Y_n, y_{1:n-1}^\xi; \Phi_n, \left\{q_{n-1}^f\right\}_{f=1:F}\right) \leftarrow$ ComposePosterior ( ); /*See sec. 5.3*/
      if $p_{\lambda_n} != 0$ then
          output   : Bounding box centered on $\mathbf{E}\left[p_{\lambda_n}\right]$;
      end
      else
          output   : Bounding box centered on the average value of the posterior of either one
                     of the filters;
      end
      /*See sec. 5.6*/
      if $\alpha_n^0 = 1, \alpha_n^\xi = 0$ then
          ReplaceOutlierPointTrackers ($p_{\lambda_n}$);
          if $\alpha_{n-1}^0 = 0, \alpha_{n-1}^\xi = 1$ then
              RestartCPF ($p_{\lambda_n}$);
          end
          else
              ResampleCPF ( );
          end
      end
      else if $\alpha_n^0 = 0, \alpha_n^\xi = 1$ then
          if $\alpha_{n-1}^0 = 1, \alpha_{n-1}^\xi = 0$ then
              ReplaceAllPointTrackers ($p_{\lambda_n}$);
          end
          else
              ReplaceOutlierPointTrackers ($p_{\lambda_n}$);
          end
          ResampleCPF ( );
      end
      else if $\alpha_n^0 = 0, \alpha_n^\xi = 0$ then
          Arrest replacement of point trackers and resampling of particles;
      end
      else if $\alpha_n^0 = 1, \alpha_n^\xi = 1$ then
          ReplaceOutlierPointTrackers ($p_{\lambda_n}$);
          RestartCPF ($p_{\lambda_n}$);
      end
  end
```



(1). Frame 1

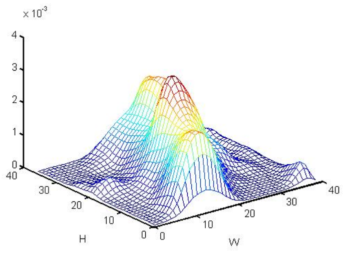


(2). Frame 92

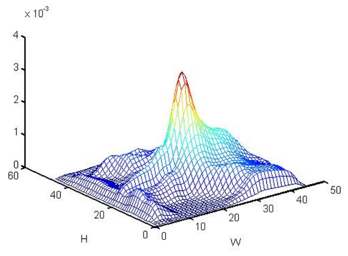


(3). Frame 283

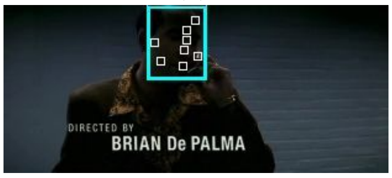
(4).



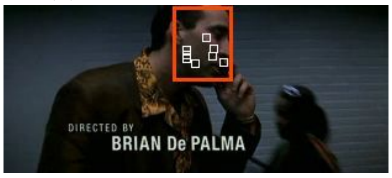
(5). Posterior at frame 92



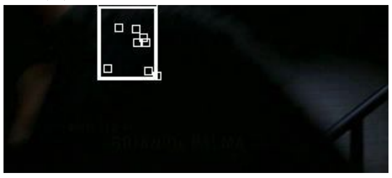
(6). Posterior at frame 283



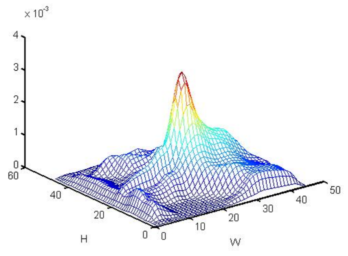
(7). Frame 283



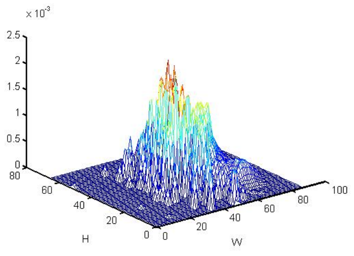
(8). Frame 304



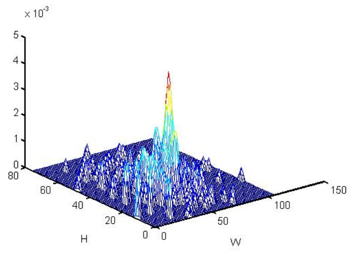
(9). Frame 326



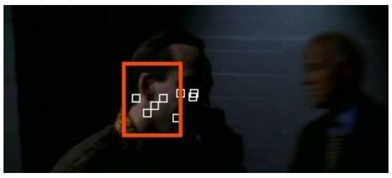
(10). Posterior at frame 281



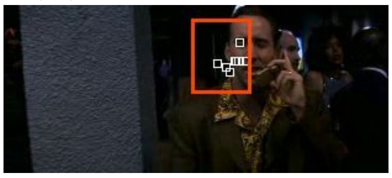
(11). Posterior at frame 304



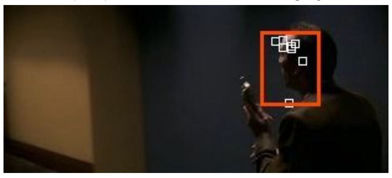
(12). Posterior of x^1_{326}



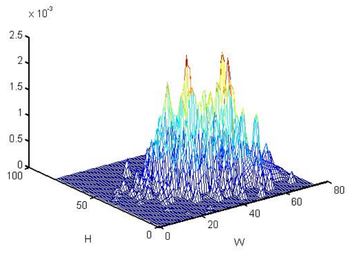
(13). Frame 485



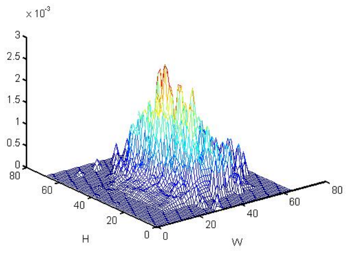
(14). Frame 588



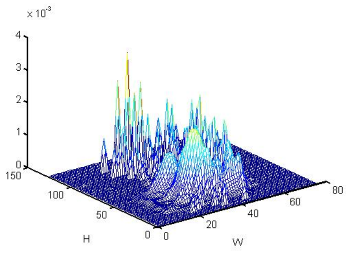
(15). Frame 792



(16). Posterior at frame 485



(17). Posterior at frame 588

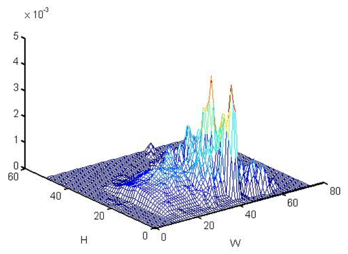


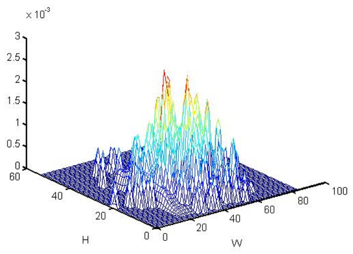
(18). Posterior at frame 792

(19). Frame 840

(20). Frame 871

(21). Frame 999


(22). Posterior at frame 840


(23). Posterior at frame 871

(24). Degenerate posterior at frame 999

Figure 5.10: Multi-cue combination tracking on the Snake-eyes sequence. Various permutations of messages are demonstrated by changing the color of the bounding box. Blue boxes signify $\alpha_n^0 = 1, \alpha_n^\xi = 0$. Yellow boxes signify $\alpha_n^0 = 0, \alpha_n^\xi = 1$. White boxes signify degenerate posteriors and orange boxes signify $\alpha_n^0 = 1, \alpha_n^\xi = 1$.

combination tracker utilizes the same concepts of reference model adaptation described for the multi-cue switch tracker. Using the same setting of the parameters as in the previous experiments, the combination tracker is put to a head tracking test on the Snake-eyes sequence. A few result samples of this test are provided in Fig. 5.10 for the sake of comparison. These samples indicate that the overall result does not vary significantly from Fig. 5.4. However the resulting "track" is smoother than the one for the multi-cue switch tracker. In particular, glitches in the track caused by the dichotomies are smoothed out. A pseudo-code for multi-cue combination tracking is presented in Algorithm 3.

The following section discusses the drawbacks and possible corrective measures for the tracking methods proposed in this chapter.

5.7 Drawbacks and suggestions

The proposed filters do not *estimate* target scale or *track* scale changes over time. Tracking or estimating target scale changes is not a trivial problem when dealing with unconstrained sequences and unknown camera parameters. Although, in principle, it is possible to estimate scale changes of the target by solving for the parameters of a motion model by using the tracks of a few point trackers, it is unreliable in practice due to the fact that ambient image noise levels are comparable in magnitude to feature point displacements. A second difficulty arises from the fact that reliable model parameter estimation is dependent on outlier data rejection, while these outliers are difficult to signal when the target motion is unknown. The strategy of using optic flow based motion model computation [Arnaud et al., 2004] may provide some respite here. Another option is to employ a model free framework where scale changes are automatically *tracked* without resorting to external estimations (see the next chapter).

The second drawback is that the tracking fails when there is gradually increasing occlusion over a few frames, in which case the replaced point tracker appearance templates have a tendency to accrue parts of the occluding object. This causes the point trackers to drift onto other objects, failing the entire scheme. The test sequences Egtest04 and Egtest05 present such a situation. One prospective way to avoid such drifting is to impose kinematic priors on x_n^0 (see the strategy used in [Arnaud et al., 2004]) and associating it with measurements which incorporate foreground/background models. However the process of deriving the posterior of x_n^0 would be more complex due to the introduction of a temporal component.

5.8 Conclusion

A general thumb rule to know if multi-cue tracking could provide reliable tracks is to check the extent of tracking performance with individual component cues themselves and if overlapping parts of the sequence can be tracked with at least one of the component cues, then multi-cue tracking can in principle provide robust tracks. It would be useful in general to have a "bag of single cue filters" from which an appropriate subset of filters can be drawn and inserted into a multi-cue model able to accomodate these filters to achieve robust tracking. The appropriate subsets themselves can be determined by a thumb rule such as the one prescribed above.
A set of point tracker filters and a color based particle filter formed the contents of the bag of single cue filters from which two robust multi-cue tracking filters were constructed here. The first of which is the multi-cue switch model based tracker which employs a novel priority switching logic for filtering. At any instant, the target is tracked by two tracking filters, the RFT-filter and a color based particle filter, with the state estimate being derived from the prioritised filter. It is important to observe that although the state estimate is derived from one filter, the other is in operation too.

Therefore, the priorities are switched and it is not that when a filter fails a new filter is brought in to play at that very instant. This switch tracker was put to extensive tests on several video sequences with targets like human heads and vehicles. Both qualitative and quantitative results assert the robustness of this tracker.

Unlike the multi-cue switch tracker, the multi-cue combination tracker employs the strategy of message combination for state estimation when both single cue filters are declared consistent. In other cases the behaviour is the same as the multi-cue switch tracker. The key functional difference between these two trackers lies in the way they handle dichotomies in the hypotheses of the component filters. The switch model uses a priority structure and is therefore capable of maintaining multiple hypotheses until one of the filters fails. On the other hand the combination model fuses the hypotheses of the component filters. Each of these two techniques could claim superiority in particular situations.

The common aspect between these two trackers is the use of an external diagnostic test to determine the current relevance of their measurements. Although it may be argued that the posteriors of the RFT-filter and color based particle filter themselves represent uncertainties in state estimation, it is important to note that these posteriors are derived using finite sample approximations for computational reasons. Therefore, employing external monitors with some empirical knowledge are useful in increasing the robustness of tracking.

The graphical model presented in this chapter is general enough to bear further modifications and additions of other filters to improve robustness of tracking. Among the immediate prospects to improve upon the multi-cue switch and combination models would be the introduction of a prior for the state x_n^0 based on optic flow computations and a measurement model based on an appropriate cue. Other interesting prospects include developing new trackers based on the concept of co-dependent particle filters presented here and its introduction into the multi-cue switch and combination graphical model.

Finally, another important point must be borne in mind when employing multi-cue trackers. It should be ensured that the interaction of cues must be symbiotic and not parasitic (mitigating each others beneficial qualities). This point is asserted in a subtle manner by way of experiments on the proposed trackers, encouraging their use.

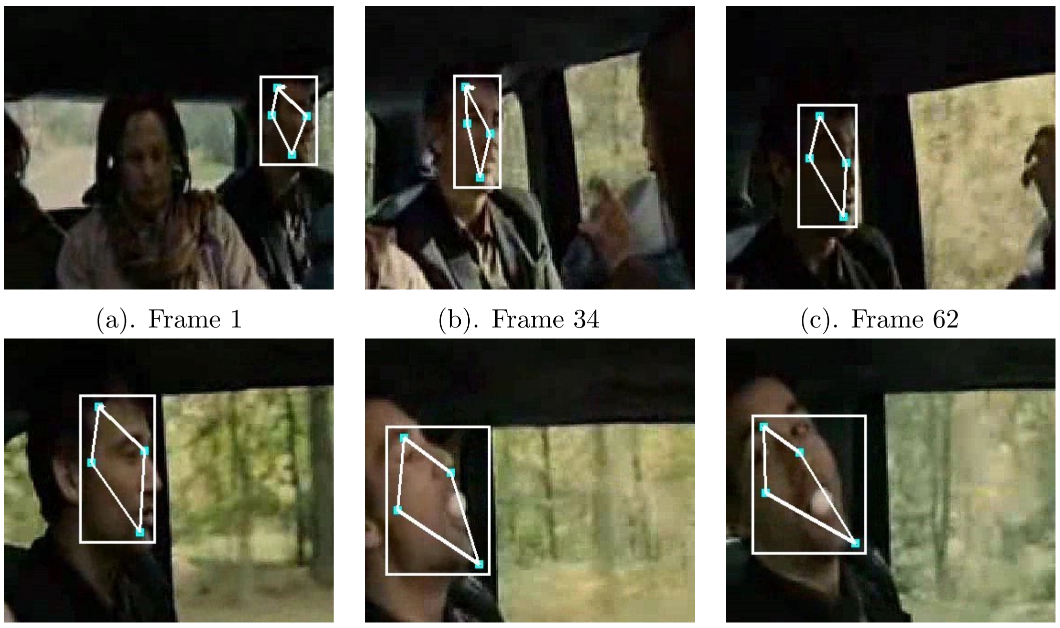
In the following chapter, the focus shifts from position tracking to tracking arbitrarily shaped layouts imposed on targets, in effect moving from lower to higher dimensional state spaces.

6

Multi-part layout based object tracking

6.1 Introduction

Visual tracking algorithms vary in their degree of robustness against distractive measurements [Hua et al., 2006] and their ability to precisely outline the boundaries of the target object [Kass et al., 1987]. Trackers working in *state spaces of small dimensions* like the ones proposed in Chapter 5 or in [Badrinarayanan et al., 2007b; Yang and Wu, 2005] perform well under drastic illumination changes, short occlusions and small out of image plane (depth) rotations of the target using various methods for target appearance model update. The update strategies range from complex holistic updates using foreground-background analysis [Yang and Wu, 2005], linear subspace or manifold based modeling [Ho et al., 2004; Lee et al., 2005] to online replacement of feature points [Badrinarayanan et al., 2007b]. Within these schemes it is only low dimensional attributes such as the position of a target which is actually tracked while other attributes such as scale and orientation are estimated in some other ad-hoc manner. For instance, the scale of a target is commonly approximated to one of a few discrete scales, as in [Comaniciu and Meer, 1999; Ho et al., 2004] or estimated using a parametric motion model derived from point matches, which is often erroneous due to its inability to combat noise in the image data or because of an oversimplified modeling of a more complex motion. Thus these techniques are in general suitable for *imprecise* or equivalently, low dimensional tracking over lengthy durations.



(a). Frame 1 (b). Frame 34 (c). Frame 62

(a). Frame 107 (b). Frame 172 (c). Frame 204

Figure 6.1: Illustration of results on the challenging *Children of Men* sequence. The little blue patches represent the centres of the local patch trackers, the rhombus represents the estimated geometric layout and the rectangle is a semi-precise bound.

In contrast, *contour trackers* [Kass et al., 1987] work in high dimensional state spaces to accurately delineate a target boundary, but need frequent user interaction to steer the unstable parts of the contour. Such trackers are usually employed for *precise* tracking over short time periods with some user interaction. A second differentiating factor between trackers working in low and high dimensional spaces is that in the former, trackers rely on appearance models like color histograms/templates of the target to estimate the *state* (location and scale) of the target at each new frame, whereas, in the latter, trackers are mainly driven by geometric cues and gradients at the occluding contours of targets.

The scheme introduced in this chapter lies *mid-way between low dimensional and high dimensional tracking.* It employs local patch trackers [Badrinarayanan et al., 2007b] and their interconnections to output a *layout* at each new frame from which the location and scale of the target can be inferred directly. The form of the layout may be used to derive other attributes like the orientation or even a *semi-precise structure* of the target, for instance, a B-spline may be passed through the vertices to capture a rudimentary shape of the target for cut-out purposes. A motivational illustration is provided in Fig. 6.1. A rhombus shaped layout is imposed on the target with a local patch tracker at each vertex of the layout. Over the sequence the layout changes in shape and size to track the target in a semi-precise manner from which attributes of interest such as position, scale and orientation can be derived. The rectangle bounds the layout in one, albeit very imprecise, manner.

From the foregoing descriptions the proposed scheme must not be misconstrued as a contender for shape/contour tracking. It is based on rough layouts imposed on the target, as opposed to precise contour based tracking as in [Zhou et al., 2005]. As the scheme is designed with a view to work with *unconstrained sequences* and be *free from offline learning of object specific priors* it cannot be seen in contention with methods which perform ball or hand tracking, as in [Isard and Blake, 1998], with shape deformation priors, deformable object detection using mesh models and feature point correspondences as in [Pilet et al., 2005] or problems like face alignment in mug-shot faces as in Liu [Liu, 2007], which requires extensive training of weak classifiers for computing mesh alignment. Others contributions such as [Sudderth et al., 2003], wherein the addressed problem is to locate several features on human faces, when some are occluded, also learn spatial interconnections between features offline as a probabilistic prior.

Among some of the examples in recent literature which employ the notion of multi-part tracking is the color based tracking using rigid geometric cues that can be found in [Perez et al., 2002], in particular, joint head and torso tracking. The work by [Yang et al., 2006] uses a distributed set of interconnected trackers to jointly track auxiliary objects and a principal target. [Veeraraghavan et al., 2006] use a set of templates linked in a geometric fashion to track moving vehicles of very small dimension. All these techniques do not however attempt to deal specifically with the difficult and important issue of multi-part target tracking with online (unsupervised) update of the interconnections between the parts.

The work reported in this chapter considers the problem of multi-part tracking of general targets and online update of the interconnections or equivalently the *layout prior*. The issue of *implicitly* accomodating changing object appearances due to rotations in depth (measuring rotations in depth in arbitrary sequences with no known camera parameters is an extremely difficult problem), scale and environment induced changes is considered here. A probabilistic graphical model framework is postulated for a concerted participation of an *ensemble of local patch trackers* for layout tracking. Methods based on sequential Monte Carlo simulations are invoked to fuse *distributed measure-*

ments arising from local patch tracking and the layout prior.
The remainder of this chapter is divided in two main sections. Section 6.2 is dedicated to the multi-part tracking problem. Section 6.2.1 describes the probabilistic graphical modeling of the problem. Section 6.2.2 describes the proposed algorithm in detail. Section 6.2.3 details the experimental setup. The results of the experiments, qualitative comparisons and discussions regarding the strengths and drawbacks of the approach form Section 6.2.4. Pointers to extension and prospective work are given in Section 6.2.5. Section 6.3 describes interactive multi-part tracking, a prospective extension to multi-part tracking described in Sec. 6.2. The probabilistic graphical model for interactive multi-part tracking is presented in Section 6.3.2, followed by the proposed nature of interactions in Section 6.3.3 and some sample results in Section 6.3.4. The chapter is concluded in Section 6.4.

6.2 Multi-part tracking

6.2.1 Probabilistic graphical model

Consider the graphical model shown in Fig. 6.2. The joint hidden state variable is denoted as $X_n = \{x_n^i,\, i\epsilon\mathcal{V}\}$ and the corresponding set of measurements is denoted as $Y_n = \{y_n^i,\, i\epsilon\mathcal{V}\}$, where $\mathcal{V}$ is the set of nodes (vertices) in the graph. Although not explicitly denoted, these measurements are indeed dependent on sequentially arriving image data $I_{0:n}$ as seen from Fig. 6.2. The filtering distribution of X_n conditional on the full observed data $I_{0:n}, Y_n$ and parameterized by $\theta, \mathcal{P_R}$ is taken to have the following form.

$$p\big(X_n|Y_n, I_{0:n}; \theta_n, \mathcal{P_R}\big) \propto l\big(Y_n|X_n, I_{0:n}\big) l_{\mathcal{P}}\big(X_n|\mathcal{P_R}\big) \prod_{(i,j)\epsilon\Gamma} \Psi\big(x_n^i, x_n^j|\theta_n\big), \tag{6.2.1}$$

where $l\big(Y_n|X_n, I_{0:n}\big)$ is a joint likelihood function. The the prior is factored into the following terms enumerated below.

1. a *layout similarity function* $l_{\mathcal{P}}\big(X_n|\mathcal{P_R}\big)$ with *fixed deterministic parameter* $\mathcal{P_R}$. A short note on this function is provided below.

   The layout similarity function:

   Layouts are in general *regular or non-regular polygons.* At the initialization frame

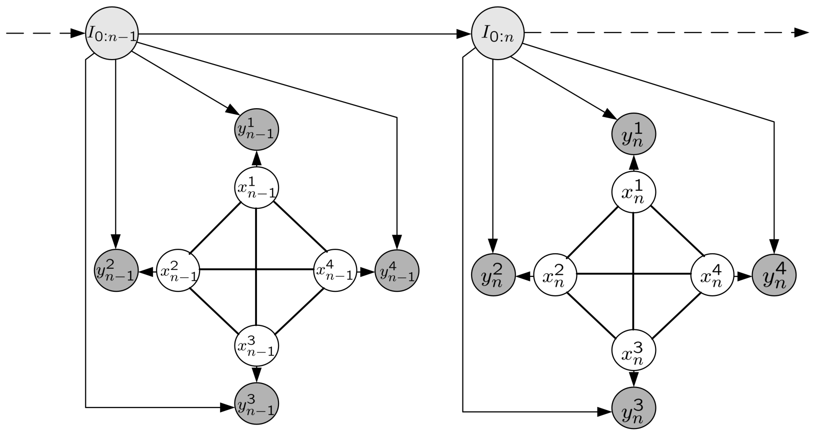


Figure 6.2: Graphical model for multi-part tracking with local patch trackers.

for the tracking sequence the polygon connecting the local patch trackers is stored as a target *reference polygon* $\mathcal{P}_{\mathcal{R}}$. A polygon represented by sample X^s_n, denoted as $\mathcal{P}\left(X^s_n\right)$ is compared with this reference to derive a measure of similarity for this sample. This comparison is done in a polygon code space composed by low dimensional coded representations of polygons. This chapter utilises the polygon turning angle based coding algorithm prescribed by [Arkin et al., 1991] to code polygons. The measure of similarity is then taken to be $\mathcal{L}_2$ distance between the *representational codes* of $\mathcal{P}_{\mathcal{R}}$ and $\mathcal{P}\left(X^s_n\right)$. This matching is in practice invariant to rotation to a reasonable extent. The layout similarity *function* is then defined as shown below:

$$l_{\mathcal{P}}\big(X^s_n|\mathcal{P}_{\mathcal{R}}\big) \triangleq \frac{1.0}{\|\mathcal{P}_{\mathcal{R}} - \mathcal{P}\left(X^s_n\right)\|}. \tag{6.2.2}$$

By construction, the above term can be evaluated for any sample of the joint hidden state.

2. A product of pairwise potential (or compatibility) functions $\left\{\Psi\big(x^i_n, x^j_n|\theta_n\big),\ (i,j)\,\epsilon\Gamma\right\}$, where θ_n is a *variable parameter* and Γ denotes the set of all edges in the graph. It is to be noted that the set of pairwise potential functions represents the layout and from here on the term *potentials* will be used interchangeably with the term layout.

Given a realization of the sequential image data $I_{0:n}$ and measurement Y_n derived from it, the objective is to *infer* the marginal posteriors $\{p\big(x^i_n|Y_n, I_{0:n};\theta_n,\mathcal{P}_{\mathcal{R}}\big),\ i\epsilon\mathcal{V}\}$ or *beliefs* at each node in the set $\mathcal{V}$, of the graph. Some interesting algorithms including

Belief/Loopy-Belief Propagation have been introduced in recent tracking literature by [Sudderth et al., 2003; Hua and Wu, 2004] for such inference tasks. The assumption underlying these algorithms is that a *prior* (the potential function) on the joint hidden state variable has been learnt offline via training. However, in an unconstrained tracking context such priors are difficult to obtain and especially when there is large *scaling* and *rotations in depth* of the target. Therefore, in an unconstrained context these priors must be updated *online*, preferably in a sequential manner, as new data arrives.

A direct application of the iterative Expectation Maximisation (EM) algorithm is not suitable for updating the parameters of the prior as the normalization factor or *partition function* of the joint posterior in Eqn. 6.2.1 is difficult to obtain in closed form in general cases (non-linear likelihood functions). Therefore, recourse is taken to Monte Carlo sampling based alternatives to approximate the posterior from a known θ_n. Subsequently, this sample based approximation of the posterior is used to update parameter θ_n using the Iterative Conditional Estimation (ICE) technique of [Salzenstein and Pieczynski, 1995]. This entire *non-sequential* iterative multi-part tracking algorithm is detailed in the following section.

6.2.2 Proposed algorithm

A set of local patches are tracked or matched from one frame to the next, independently of each other, using some convenient method. To avoid digression, discussions on the specific patch tracking method employed in this chapter is postponed to Sec. 6.2.3.1. For the present, consider the following general form of measurement derived by such methods.

$$y_n^i = \underset{\hat{y}_n^i \epsilon S_n}{\operatorname{argmin}} \mathcal{D}\big[f\big(\hat{y}_n^i, I_n\big), f^*\big(I_{0:n-1}\big)\big], \tag{6.2.3}$$

where $\mathcal{D}$ is a distance function, typically template/patch cross-correlation [J.P.Lewis, 1995] or SSD [Nickels and Hutchinson, 2002], $f\big(\hat{y}_n^i, I_n\big)$ is the extracted local patch centered around location $\hat{y}_n^i$ in image I_n and $f^*\big(I_{0:n-1}\big)$ is a reference model for the tracked local patch derived using past image data $I_{0:n-1}$. S_n is a search space on the image I_n centered on a estimate of x_{n-1}^i. Now the following relationship is postulated between the measured variable y_n^i and the hidden state x_n^i

$$y_n^i = x_n^i + \mathcal{N}\big(0, \Sigma_n^i\big), \tag{6.2.4}$$

where $\mathcal{N}\big(0, \Sigma_n^i\big)$ is a zero mean Gaussian noise with *empirically determined* diagonal covariance matrix Σ_n^i. With this measurement model and the assumption of independence of measurements conditional on the joint hidden state (see Fig. 6.2), the joint likelihood function can be written as follows.

$$l\big(Y_n|X_n, I_{0:n}\big) = \prod_{i \epsilon \mathcal{V}} \mathcal{N}\big(y_n^i; x_n^i, \Sigma_n^i\big), \tag{6.2.5}$$

where, the notation $\mathcal{N}\big(y_n^i; x_n^i, \Sigma_n^i\big)$ represents a Gaussian distributed variable y_n^i with mean parameter x_n^i and variance parameter Σ_n^i.

The pairwise potential functions describing the relationship between the local state variables are assumed to have a Gaussian form as shown below;

$$\Psi\big(x_n^i, x_n^j|\theta_n\big) = \mathcal{N}\big(|x_n^i - x_n^j|; \mu_n^{ij}, \Sigma_n^{ij}\big),\ (i,j)\,\epsilon\Gamma, \tag{6.2.6}$$

by which the parameter $\theta_n = \Big\{\mu_n^{ij}, \Sigma_n^{ij};\ (i,j)\,\epsilon\Gamma\Big\}$. A note of interest here: the *form of the potential is not a typical multi-variate Gaussian*, as it is based on the *absolute difference* of state variables. This clearly brings out the inadequacy of attempting to learn this type of prior offline; if attempted, even a simple scaling of the target would affect the learnt prior considerably. Interestingly, most algorithms in tracking literature based on graphical model formulations [Sudderth et al., 2003; Briers et al., 2005; Khan et al., 2004] do not attempt to deal with situations were the potentials undergoes scaling, or assume that the scale normalization is performed by some external device.

6.2.2.1 Importance sampling approximation of the joint posterior

Given a realisation of $I_{0:n}$ and an initial estimate of the parameter θ_n^{k-1}, at *iteration* $k-1$, the filtering distribution of X_n (see Eqn.6.2.1) can be approximated using importance sampling. To this end, importance samples are drawn from a suitable proposal or importance sampling density developed as described below.

Developing a proposal density:
Letting aside the layout based prior term found in Eqn.6.2.1 which makes sampling unpractical, consider the following model of the filtering distribution parameterized by the estimate θ_{k-1};

$$g\big(X_n|Y_n, I_{0:n}; \theta_n^{k-1}\big) = l\big(Y_n|X_n, I_{0:n}\big) \prod_{(i,j)\epsilon\Gamma} \Psi\big(x_n^i, x_n^j|\theta_n^{k-1}\big). \tag{6.2.7}$$

Probabilistic inference on this model using standard belief propagation [Yedidia et al., 2001] results in marginal posteriors $\{g(x_n^i|Y_n;\theta_n^{k-1}),\ i\epsilon\mathcal{V}\}$ or beliefs, denoted as $\{\mathbf{b}_{k-1}(x_n^i), i\epsilon\mathcal{V}\}$. It is re-emphasized that *belief propagation is necessary as the priors are not multi-variate Gaussian* which disallows analytic computations. The proposal density is then developed using these beliefs as shown below.

$$q(X_n|Y_n, I_{0:n};\theta_n^{k-1}) = \prod_{i\epsilon\mathcal{V}} \mathbf{b}_{k-1}(x_n^i). \tag{6.2.8}$$

The above form of the proposal is easy to sample from and is also a way to digest global prior information in course of developing the proposal density. Samples $\{X_n^s \sim q(X_n|Y_n, I_{0:n};\theta_n^{k-1}),\ s = 1 : M\}$ are drawn and their unnormalized importance sampling weights w_{k-1}^s computed as shown below:

$$w_{k-1}^s = \frac{l(Y_n|X_n^s, I_{0:n}) l_{\mathcal{P}}(X_n^s|\mathcal{P}_{\mathcal{R}}) \prod_{(i,j)\epsilon\Gamma} \Psi(x_n^{i,s}, x_n^{j,s}|\theta_n^{k-1})}{\prod_{i\epsilon\mathcal{V}} \mathbf{b}_{k-1}\left(x_n^{i,s}\right)}. \tag{6.2.9}$$

Following the above computation, the filtering distribution of the joint hidden state, parameterized by estimate θ_n^{k-1}, can be approximated as indicated below.

$$p(X_n|Y_n, I_{0:n};\theta_n^{k-1}, \mathcal{P}_{\mathcal{R}}) \approx \sum_{s=1:M} \frac{w_{k-1}^s}{\sum_{s=1:M} w_{k-1}^s} \delta_{X_n^s}(X_n). \tag{6.2.10}$$

In order to arrive at the above approximation, it is necessary that the layout similarity function of the prior, $l_{\mathcal{P}}(X_n|\mathcal{P}_{\mathcal{R}})$, can be evaluated pointwise (a fact herein ensured by construction).

The sample-set approximations of filtering distributions of X_n can be visualized in Fig. 6.3. The joint hidden state samples shown are *true samples* drawn from sample-set approximations. Therefore, their spatial spread provides an insight into the uncertainty of the layout estimate at that instant. In the next step, the parameters of the potentials are updated based on the approximated filtering distribution.

6.2.2.2 Parameter update

The Iterative Conditional Estimation (ICE) technique [Salzenstein and Pieczynski, 1995] prescribes parameter update from *complete data*, $\{X_n, Y_n, I_{0:n}\}$. The essence of this technique lies in the following iteration;

$$\theta_n^k = \mathbf{E}_{p\left(X_n|Y_n,I_{0:n};\theta_n^{k-1},\mathcal{P}_{\mathcal{R}}\right)} \Theta(X_n, Y_n, I_{0:n}), \tag{6.2.11}$$

where $\Theta\big(X_n, Y_n, I_{0:n}\big)$ is a statistical estimator of θ_n. Substituting the sample-set approximation from Eqn. 6.2.10 in Eqn.6.2.11 leads to a sample based approximation of the above expectation as shown below:

$$\theta_n^k \approx \sum_{s=1:M} \widetilde{w}_{k-1}^s \Theta\big(X_n^s, Y_n, I_{0:n}\big). \tag{6.2.12}$$

In the experiments reported in this paper, $\Theta\big(X_n, Y_n, I_{0:n}\big)$ is chosen to be the Maximum Likelihood Estimator (MLE);

$$\Theta\big(X_n, Y_n, I_{0:n}\big) = \underset{\theta_n}{\operatorname{argmax}}\, p\big(X_n | Y_n, I_{0:n}; \theta_n, \mathcal{P}_{\mathcal{R}}\big). \tag{6.2.13}$$

For the case of the Gaussian form of the prior postulated here, Eqns.6.2.11, 6.2.13 lead to parameter update of the following form.

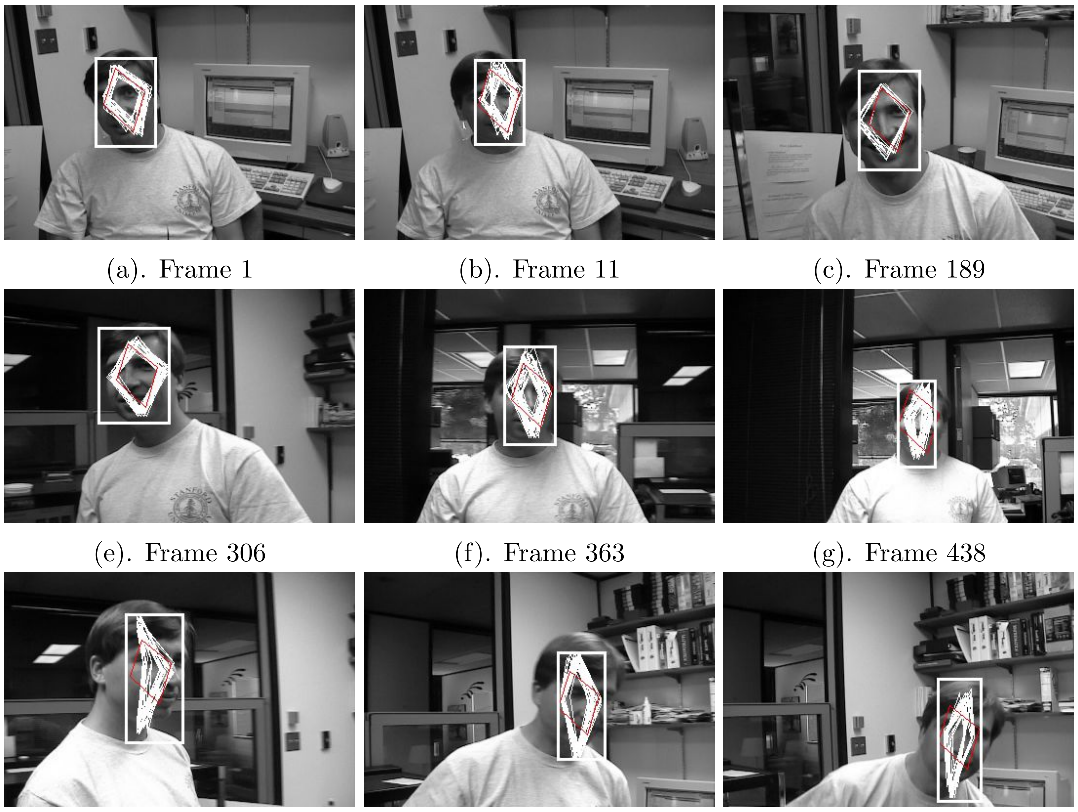
$$\begin{aligned}
\mu_n^{ij,k} &= \sum_{s=1:M} \widetilde{w}_{k-1}^s |x_n^{i,s} - x_n^{j,s}|, \ (i,j)\,\epsilon\Gamma, \\
\Sigma_n^{ij,k} &= \sum_{s=1:M} \widetilde{w}_{k-1}^s \big(|x_n^{i,s} - x_n^{j,s}| - \mu_n^{ij,k}\big)^T \big(|x_n^{i,s} - x_n^{j,s}| - \mu_n^{ij,k}\big), \ (i,j)\,\epsilon\Gamma.
\end{aligned} \tag{6.2.14}$$

The steps described in Sections 6.2.2.1 and 6.2.2.2 can be iterated until convergence. Algorithm. 4 provides a pseudo code summarising the discussions so far. The next section focuses on the experimental setup to test this algorithm.

6.2.3 Experimental setup

The video sequences used in all the experiments had targets undergoing significant rotations in depth and/or large scale changes. In addition poor recording quality, rapid changes in illumination and motion jerks are also found in some of them (See 6.7). Occlusion handling is currently beyond the scope of this chapter and therefore, the *targets are unoccluded* in all the sequences.

Quantitative comparisons are difficult to perform for the proposed approach as there are no direct contenders specifically dealing with the issue of multi-part tracking and online update of prior in unconstrained sequences. Tests are performed with standard sequences to the extent possible to aid visual comparison with other results in literature. Standard video sequences like the Fleet sequence [Jepson et al., 2003], the Behzad sequence (Honda/UCSD database) and the deflating Balloon sequence were used for testing the scheme. The sources for these sequences and, if available, links to some *comparable results*, which include techniques like adaptive template matching, are provided below the appropriate results. Further new challenging video sequences



(a). Frame 1 (b). Frame 11 (c). Frame 189

(e). Frame 306 (f). Frame 363 (g). Frame 438

(h). Frame 547 (i). Frame 572 (j). Frame 589

Figure 6.3: Tracking the *Jepson and Fleet* sequence. 30 joint hidden state samples from the sample-set approximation of the filtering distribution are displayed to give an insight into the uncertainty of the state estimate. The red polygon is the unscaled user defined reference model, superimposed on each frame for the sake of comparison.

Algorithm 4: Multi-part layout based tracking

```
input  : Video sequence with $L$ frames and local patch bounding boxes $\{\mathcal{B}_i, i \in \mathcal{V}\}$ in first
         frame;
Initialization;
  InitializePatchTrackers( ); /* See Sec.6.2.3.1 */
  $\mathcal{P}_{\mathcal{R}} \leftarrow$ SetLayoutReferencePolygon$\Big(\{\mathcal{B}_i, i \in \mathcal{V}\}\Big)$; /* Set the parameters of the Gaussian
  prior */
  $\{c_i, i \in \mathcal{V}\} \leftarrow$ BoundingBoxCenters$\Big(\{\mathcal{B}_i, i \in \mathcal{V}\}\Big)$;
  $\theta_0 \equiv \{\mu_0^{ij} = |c_i - c_j|, \Sigma_0^{ij} =$ large initial value; $(i,j)\, \epsilon \Gamma\}$; /* See Sec.6.2.3.1 */
  for $n \leftarrow 1$ to $L$ do
      for $i \in \mathcal{V}$ do
          $\mathcal{N}(y_n^i; x_n^i, \Sigma_n^i) \leftarrow$ TrackLocalPatch$(i, I_{0:n})$;
      end
      /* Iterative Monte Carlo update, see Secs.6.2.2.1,6.2.2.2 */
      $\theta_n^0 = \theta_{n-1}$
      for $k \leftarrow 0$ to $K$ do
          $\mathbf{b}_k(x_n^i), i\epsilon\mathcal{V} \leftarrow$ DevelopProposal$\Big(\{\mathcal{N}(y_n^i; x_n^i, \Sigma_n^i), i \in \mathcal{V}\}, \theta_n^k\Big)$;
          $p(X_n|Y_n, I_{0:n}; \theta_n^k, \mathcal{P}_{\mathcal{R}}) \approx \sum_{s=1:M} \frac{w_k^s}{\sum_{s=1:M} w_k^s} \delta_{X_n^s}(X_n) \leftarrow$
          ApproximateJointStatePosterior$\Big(\{\mathbf{b}_k(x_n^i), \mathcal{N}(y_n^i; x_n^i, \Sigma_n^i), i \in \mathcal{V}\}, \theta_n^k\Big)$;
          $\theta_n^{k+1} \leftarrow$ MLParameterUpdate$\Big(\sum_{s=1:M} \frac{w_k^s}{\sum_{s=1:M} w_k^s} \delta_{X_n^s}(X_n)\Big)$;
      end
      for $i \in \mathcal{V}$ do
          if *found_outliers* then
              ReplaceOutliers( );
          end
      end
      output  : Layout $\equiv \mathbf{E}\big[p(X_n|Y_n, I_{0:n}; \theta_n^K, \mathcal{P}_{\mathcal{R}})\big]$;
  end
```

like the Children of men sequence Figs. 6.1, 6.7 are introduced to test the efficacy of the approach.

At each frame, one iteration of deriving the approximate joint hidden state filtering distribution and subsequent parameter update was seen to be sufficient, as only one measurement Y_n is available at any instant n. Before concluding the details of the experimental setup, the characteristics of the local patch trackers used to derive these measurements Y_n are now described.

6.2.3.1 Local patch tracking

Nature of a patch and patch tracking:

A patch is constructed or equivalently defined by a set of feature points and their, *possibly overlapping*, appearance templates. With this definition, a patch can be of any *arbitrary shape* and is seldom rectangular. An example patch constructed using 4 feature points and their templates is shown in Fig. 6.4.

In practice, the location of a local patch i is tracked using the RFT-filter proposed in Chapter 4. The resulting filtering distribution is approximated to a multi-variate Gaussian distribution $\mathcal{N}_i(\hat{\mu}_n^i, \hat{\Sigma_n^i})$ by moment matching. This distribution is then used to derive the statistics of the measurements, for instance, at local patch (vertex) i, $y_n^i = \hat{\mu}_n^i, \Sigma_n^i = \hat{\Sigma_n^i}$ (See Eqn. 6.2.4) and similarly for the other vertices.

As in RFT-filtering, at each instant n, a subset of outlier feature points in a local patch (and thereby their appearance templates) are replaced online (see Chapter 4) and therefore, the patch *evolves* over time. An illustrative procedural detail of patch tracking is provided in Fig. 6.4 alongwith additional explanations.

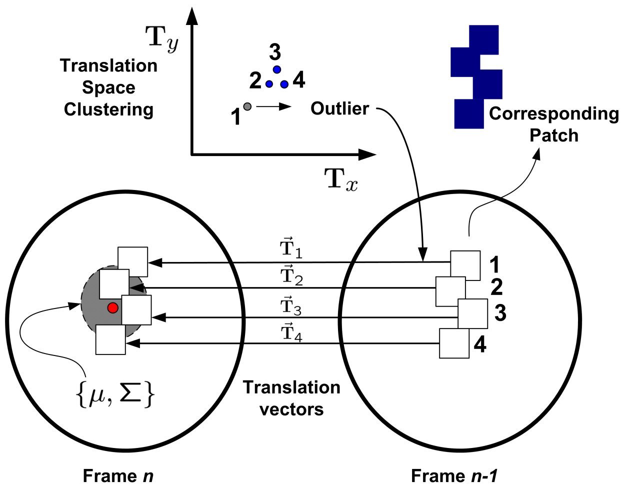


Figure 6.4: Local patches and patch tracking. A patch, see top right hand corner, is constructed using $F = 4$ feature points, with hand tuned template and search dimensions. Templates are marked as small rectangles. RFT-filtering is used to track the location of the patch. μ, Σ are the parameters of the Gaussian likelihood derived from the RFT-filtering distribution (see text).

Local patch selection:

Patches are manually selected at the *periphery of the tracked object* in a way that the *layout of the patches are isomorphic to the general shape of the object*, for instance, an elliptical layout is a natural choice for human faces. The number of patches can be varied depending on the complexity of the layout and the computational power at one's disposal.

Patch evolution:

A patch evolves because some of the feature points in its defining set are replaced online as and when these points are signalled as outliers (see Sec.4.4.3). An important point arises in this context. When feature points are replaced, new appearance templates are associated to these new points, as a consequence of which the patch composition changes. These *new templates* correspond to *new parts* on the target derived from current data. Therefore, some templates in a patch could possibly be from the distant past (if a particular point in the patch has been consistently tracked for a long duration), some others could be relatively new and the rest completely new. Clearly, as the set of templates also form the reference model for the patch, replacing feature points (thereby bringing in new templates) means changing this reference model. This change is essential to adapt the patch to changing appearances of the target. The method by which feature points are replaced is described below.

Feature point replacement methodology:
In the result samples shown in Fig. 6.5, feature points at some patch (vertex) need to be eliminated and replaced in order to adapt to deformations or rotations in depth. The outlier feature points (See Sec.4.4.3 for an explanation of how outliers are detected) in each patch are eliminated from the set of F feature points and replacement feature points, denoted $\left\{F_i^R,\ i\epsilon\mathcal{V}\right\}$, are drawn from corresponding densities:

$$F_i^R \sim \mathcal{N}_{x_n^i}\left(\mathbf{E}_{p\left(x_n^i|Y_n,I_{0:n};\theta_n,\mathcal{P}_{\mathcal{R}}\right)}\delta_{x_n^{i,s}}\left(x_n^i\right),\mathcal{S}\right),\ \ i\epsilon\mathcal{V} \tag{6.2.15}$$

where the expectation is evaluated using the MC approximation of the belief and $\mathcal{S}$ is a fixed sampling covariance (diagonal) matrix (It is set to $25.0\mathcal{I}_{2\times 2}$ in the experiments).

6.2.4 Results and discussions

The results presented in this section are roughly arranged in an *increasing order of difficulty*, starting from contrived lab test videos to outdoor cinema sequences. The first sequence shown in Fig.6.5 (http://esm.gforge.inria.fr/ESMdownloads.html) is a deflating and deforming balloon sequence presenting large scale changes but with little rotation in depth. The scale is quite accurately tracked throughout the sequence.
The Behzad1 test sequence from the Honda/UCSD database shown in Fig.6.6 (http://vision.ucsd.edu/leekc/HondaUCSDVideoDatabase/ HondaUCSD.html) presents a greater challenge to the proposed algorithm with frequent in-plane and out of plane rotations, scale changes and fast motions. Frequent replacement of feature points can be observed in several frames to quickly adapt to changing appearances of the target.

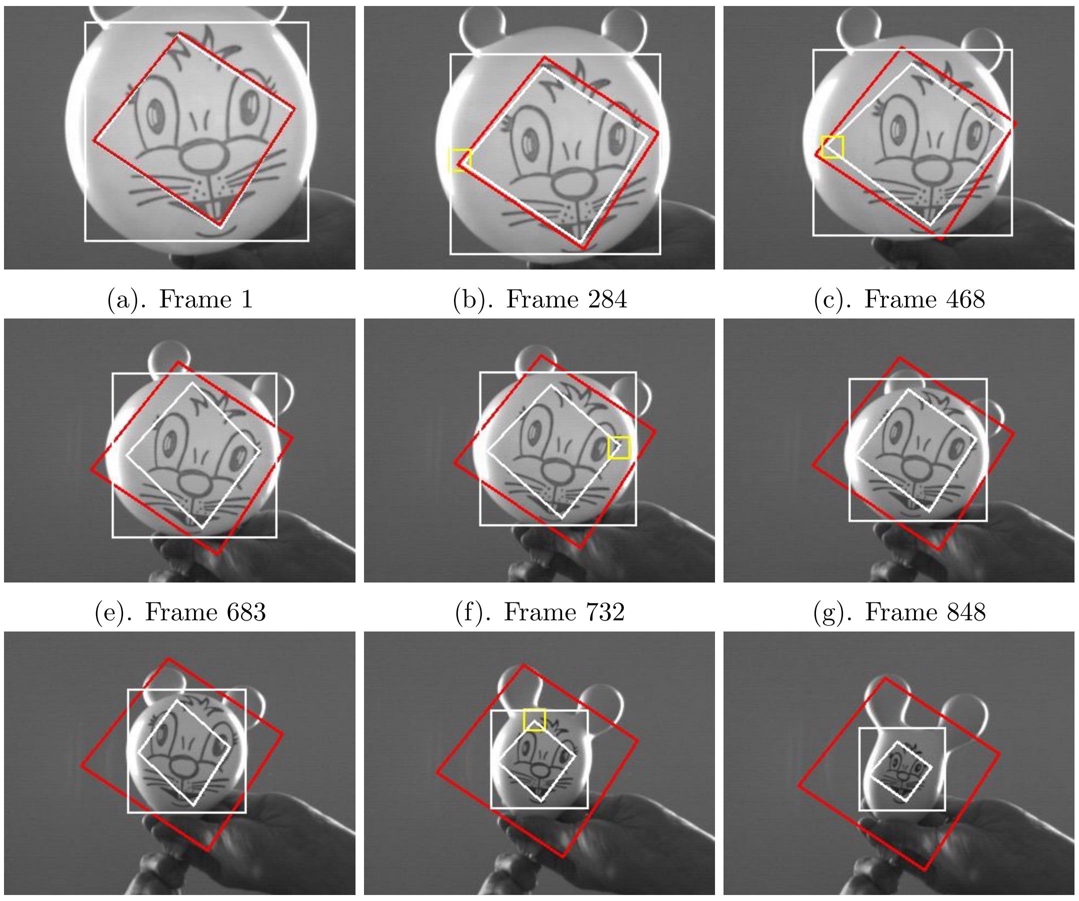
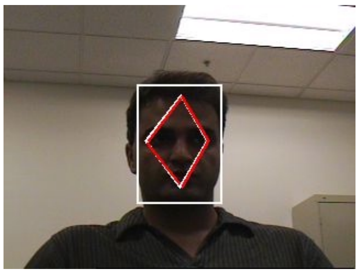
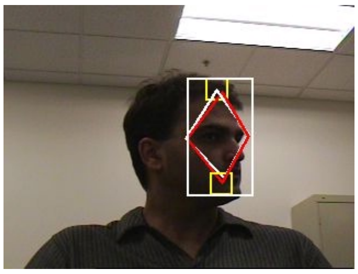
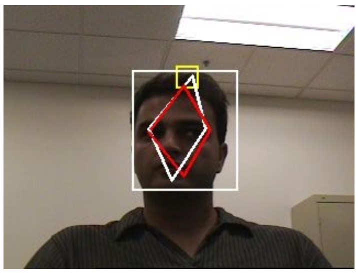
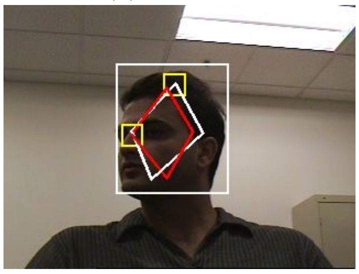


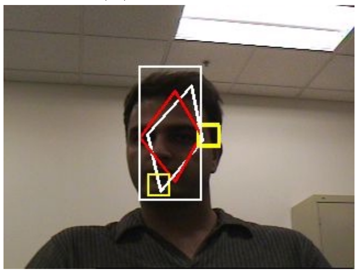
Figure 6.5: Results on the *Deflating Balloon*(Mouse) sequence. The white polygon represents the estimate of the geometric layout derived as the mean of the joint posterior. Newly sampled feature points are displayed in small yellow rectangles. Source video can be found at http://esm.gforge.inria.fr/ESMdownloads.html

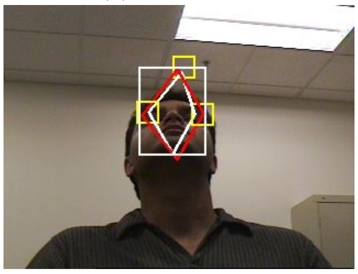

(a). Frame 1

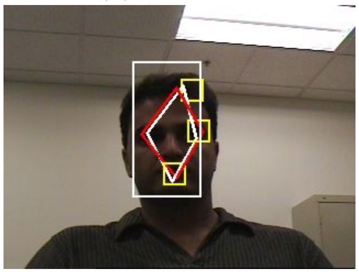

(b). Frame 29

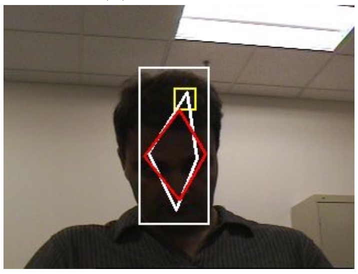

(c). Frame 47

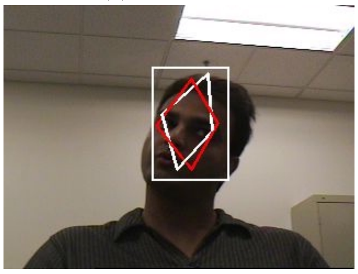

(d). Frame 53

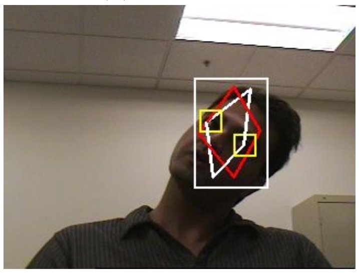

(e). Frame 64

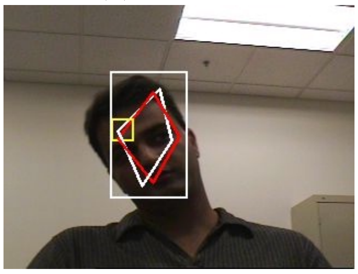

(f). Frame 77

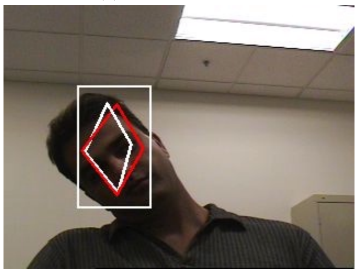

(g). Frame 85

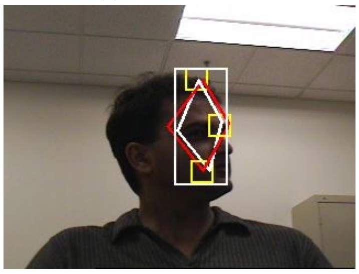

(h). Frame 91

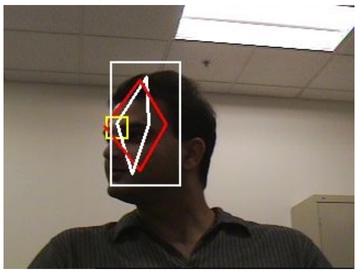

(i). Frame 107


(j). Frame 113


(k). Frame 128


(l). Frame 135


(m). Frame 154


(n). Frame 169

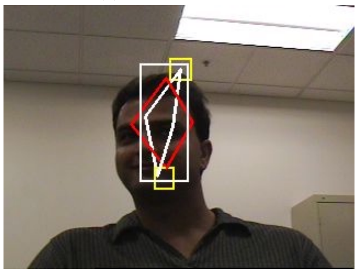
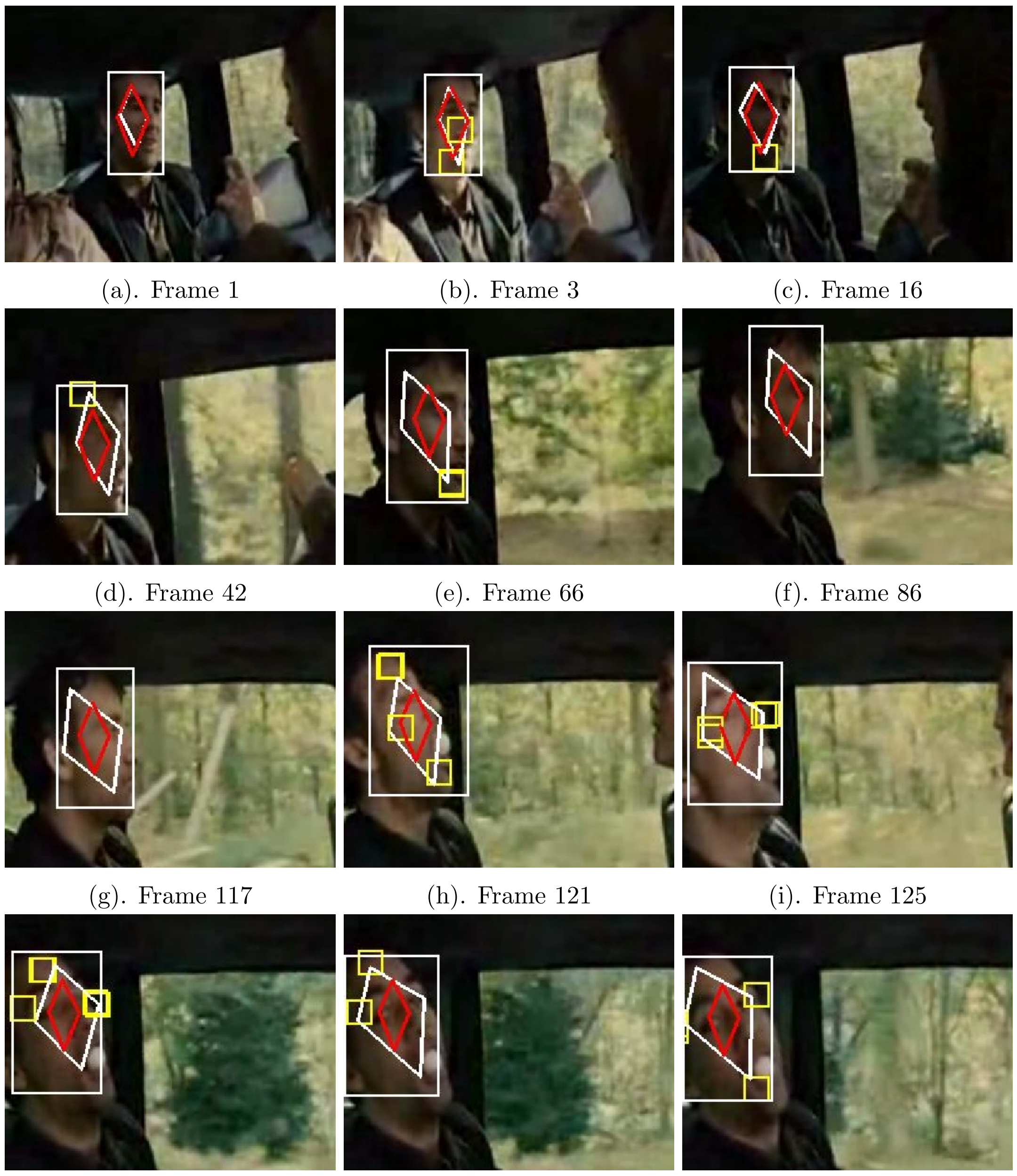

(o). Frame 178

Figure 6.6: Results on the Honda/UCSD Database *Behzad1* test sequence. Source video can be found at http://vision.ucsd.edu/leekc/HondaUCSDVideoDatabase/ HondaUCSD.html.



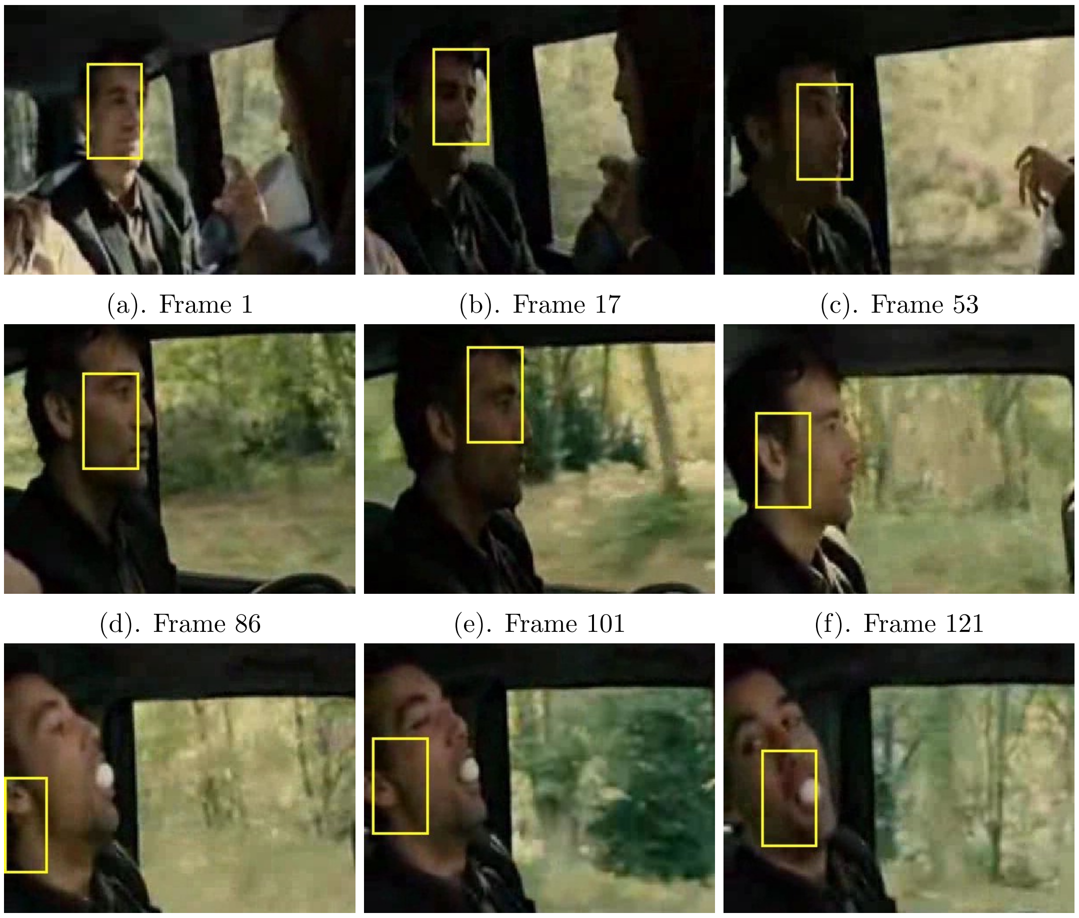
(a). Frame 1 (b). Frame 3 (c). Frame 16

(d). Frame 42 (e). Frame 66 (f). Frame 86

(g). Frame 117 (h). Frame 121 (i). Frame 125

(j). Frame 141 (k). Frame 143 (l). Frame 146

Figure 6.7: Results on the *Children of Men* sequence. This video has poor recording quality. Large motion jerks eventually cause tracking failure.


(a). Frame 1 (b). Frame 17 (c). Frame 53

(d). Frame 86 (e). Frame 101 (f). Frame 121

(g). Frame 146 (h). Frame 160 (i). Frame 169

Figure 6.8: Qualitative comparison with a color based particle filter of [Perez et al., 2002]. The frequent sliding of the estimated position is clearly noticeable.

The proposed tracker is able to follow the key part of the face for most parts of the sequence. The scale of the target is also assessed reasonably well (See Figs. 6.6 (f) and 6.6(h)). In Figs. 6.6(f) and 6.6(m) replacement features are positioned at the threshold between the target and the background, but the presence of the layout prior prevents tracking drift. However, after several rotations in depth of the target, the tracking quality detoriates, primarily due to a poor proposal density warranted by absence of a kinematic prior on the hidden states and reliance on the likelihood and layout prior alone. This leads to misplaced feature points leading to imperfections (See Fig.6.6(p)) in tracking. In consequence the algorithm also fails to accurately track the orientation of the target.

The ability of the tracker to perform on poor quality videos is tested in Fig. 6.7.

The target is tracked despite large illuminations changes caused by shadowing, due to normalised cross correlation based feature point tracking and timely feature point replacements. In comparison, a 200 particle color based filter of [Perez et al., 2002] is distracted by this frequent change in illumination and results in jittery estimates of the position of the target as seen from the result samples in Fig. 6.8. A scheme to adapt the reference color model is necessary but this problem of *holistic* color model adaptation is indeed a difficult one. In contrast, patch evolution via feature point replacement handles necessary *partial* adaptation of the target reference model implicitly. The tracker also performs well in presence of some out of plane rotations due to target pose changes and ego-motion of the camera. The implicit following of scale changes is brought out in this sequence. In current day literature on filtering via Monte Carlo simulations, scale estimation is generally *decoupled* from position estimation due to dimensionality problems. Even if these dimensionality problems are resolved the issue of defining an adequate likelihood model for a "scale variable" is difficult.
Finally, between 30-50 joint hidden state samples were used in all the experiments and the computational time for the tracker on a 2GHz CPU machine was estimated to be in the order of 10-12 frames/sec without in-depth optimization of the code.

6.2.5 Prospects

An interesting prospect would be to improve the proposal distribution for importance sampling. The high dimensionality of the joint hidden state and the MRF model makes it difficult to develop, easy to sample yet adequate proposals. Nevertheless, attention need to be paid towards incorporating kinematic priors on the hidden states, although this would lead to the complicated issue of performing probabilistic inference on a Dynamic Bayesian Network (DBN) model. Additional forms of measurements, say, via tracking edges close to the vertices, could be included in the development of a better proposal.
It is informative to note that the graphical model on which the multi-part tracking algorithm is based is free from any topological constraints, therefore, other graphs modeling a different conditional independency structure could also be experimented with. The Gaussian nature of the geometric potentials can also be relaxed to include non-Gaussian potentials, although at the expense of additional complexity during the inference stage. Although most results shown in this paper are on human faces, the algorithm inherently is free from any object specific prior which is an added advantage.

The algorithm's use in arbitrary environments is still restricted due to the absence of occlusion handling capabilities. An immediate extension which can be envisaged is the introduction of color based holistic observations to recover from occlusions and kinematic priors. Finally, an alternative strategy would be to exploit the very concept of multi-part layout based tracking in an interactive setup. Several vision applications where tracking algorithms find place can afford a reasonable amont of user interaction if the interaction is intuitive and easy. Such interactions are indeed indispensible for several reasons including, lack or difficulty in obtaining a good prior, re-steering of faulty tracks resulting from lack of robustness in tracking, need of quick and efficient ways to generate semi-precise tracks/mattes for other applications, for instance in compositing cinema sequences or even defining seeds and ground truths for other experiments such as video target segementation. The following part of this chapter is dedicated to discussions on interactive multi-part tracking addressing these and such needs.

6.3 Interactive tracking with evolving patches and color

6.3.1 Introduction

Several video processing applications such as video editing, alpha-matting and colour correction rely on visual tracking schemes to ease their burden. These applications are usually semi-automatic, meaning it requires user interaction to achieve satisfactory results. Visual tracking schemes can be modified to take advantage of these interactions to robustify their functioning. This section describes an interactive visual tracking scheme based on an extension of the probabilistic multi-part tracking model proposed in the previous chapter. This scheme has the ability to deliver tracks of rough arbitrarily shaped layouts in unconstrained videos allowing for some user interaction. Such tracks can then lead to rough video object cut-outs useful for purposes such as, making *video collages*, as *seeds* for finer object segmentation in a track and batch-segment scheme, developing *target mattes*, to access *video tubes* of human faces or other objects for training other algorithms and even generation of *ground truths* for tests.

In recent literature there have been some schemes proposed for interactive visual tracking. To set the context for the contents of this section a review of these schemes is given below under two categories.

1. **Batch processed interactive visual tracking**
   The defining aspect of such schemes is that the test videos are preprocessed before the tracking run. An example in this category is the interactive feature tracking scheme of [Buchanan and Fitzgibbon, 2006]. The problem they tackle is the tracking of small image patches over lengthy durations. Initially the patches around each pixel in each frame of the video sequence are projected onto a patch basis (constructed using some randomly chosen patches in the video sequence) and so encoded.
   When a patch is to be tracked, the M best matches (equivalently patch detections) to this patch in each frame are computed. Between one frame to the next pairwise match errors for these best matches are computed based on a linear combination of a motion smoothness term, a term for the appearance similarity between the two consecutive frames and a third appearance similarity term measuring the similarity of the current image patch to the user provided set of feature templates (from chosen key frames). These matches and their corresponding match errors form the states at each stage (frame) of a Dynamic Programming (DP) table. Apart from these matches each stage in the table also includes an occlusion state with a user controlled occlusion penalty. The optimal track for the patch is then computed using the standard principle of optimality of DP.
   The user interaction lies firstly in the adjustment of the occlusion penalty and the coefficients of the linearly combined match error. Secondly, upon noticing an error in the track at some frames, the user can augment key frame feature templates. To aid the choice of these templates the authors propose an auto-track feature which uses the dynamic programming track optimization approach. After a run of DP, the image patch around a point on the optimal track most unlike the ones in the user provided set of feature templates is added to this set. This eases somewhat the burden on the user. However as the authors point out this procedure assumes there are few false positives on the optimal track. Once these templates are augmented the DP track optimization is rerun. These two steps are repeated until the error converges.
   To provide some insight into the capabilities of their interactive setup, the authors track a patch on a human face for a lengthy duration with few key frame templates and parameter adjustments, and show that it is difficult to obtain a comparable result using techniques like mean shift tracking [Comaniciu et al., 2000] or KLT

[Tomasi and Kanade, 1991] tracking even with manual restarts after each failure. Results on tracking features undergoing occlusion are also shown. If the user has the convenience of pre-processing the videos, as could be the case in most offline applications, this setup is very useful. Tracking very discriminative features like corners, eyes and such can be performed using this method with few interactions. However, there is no study reported in their work on what amount of interactive effort could possibly be incurred on tracking less discriminative points. Following which a question arises as to whether there is an obvious extension of this approach to larger patches or regions, which would be particularly useful for color correction purposes or inserting special effects. Further more, if the user desires to track N different patches on an object then it effectively multiplies the necessary interactive effort by at least a factor of N, if the important cue of geometric relationships between the patches are not harnessed. Aside from this fact, the necessary parameter adjustments and setting of the occlusion penalty is not very intuitive for an operator. In conclusion, an extension of this approach to tracking both the position and scale of larger patches, regions or layouts is awaited.

A second example of interactive tracking of rectangular regions using color is the offline scheme proposed by [Wei et al., 2007]. Their approch relies on the paradigm that a combination of user interaction, object detection, tracking and batch optimization can result in good quality tracking within a reasonable duration and effort. Their strategy is a three fold one. First, select some key frames where the object bounding box is marked by the user. Construct boosted color histogram features of the target from this input and train a set of weak classifiers based on these features. A strong classifier which is constructed as a combination of these weak classifiers is used to pick the best N detections at evenly sampled I-frames over the entire sequence. In stage two, beginning from these I-frames a mean shift tracker is used to grow object trajectories forwards and backwards. From these trajectories and a slightly involved best first strategy, N object candidates are selected in each of the non I-frames. Building a DP table with these candidates, DP based track optimization is used to minimize a cost function similar to the one proposed by [Buchanan and Fitzgibbon, 2006] which also includes an occlusion penalty. The optimal track is so obtained.

The key to the success of this approach is the trajectory growing technique which eliminates a large number of false positive detections and a robust appearance model construction using a combination of weak classifiers. The method is shown to produce reasonable results with sequences presenting several occlusions and scale changes with few key frame markings. Overall such a technique is computationally less intensive than the interactive feature tracking technique and is able to track regions. However issues of parameter adjustments remain and the authors do not provide statistics about the number of adjustments required to achieve the results. Aside from this there is no key insight provided into how to choose the key frames (if there is an auto-track type feature which could be used) as the computationally intensive trajectory growing technique needs to be repeated when a new key frame is inserted. Another possible issue is in constructing the appearance model from these key frames, if, say, a key frame is chosen at every distinctive color variation presented by the object, then there is a risk of over learning the classifiers. Tests on lengthy and difficult videos would present such issues.

2. **Near sequential interactive visual tracking**
   All interactive approaches which do not treat the tracking problem as an optimization problem with some fixed constraints (set with user provided key frame samples) are classified here as near sequential approaches. This includes approaches where the user restarts a tracker, like a mean-shift tracker, when it is led astray by confusion in its measurements, adjusting nodes of an active contour tracking scheme and the like. These are the most straightforward modes of interactions, but nevertheless useful and practiced in most application scenarios. It is in this category that lies the contribution of this section, which is primarily an investigation into how the prior on the hidden states in the graphical model of Fig. 6.2 can be interactively modified to obtain desirable tracking results. If the prior itself is Gaussian, then the user needs to control only two intuitive parameters, namely the sufficient statistics. In comparison with the batch processed methods, the key capability of this interactive setup is the ability to track arbitrary layouts with reasonable interactive effort over long sequences.
   The judgement of the quality of the resulting track due to interactions is a very subjective issue, especially in the absence of ground truth, which is so often difficult to obtain. Ironically, interactive tracking is itself indispensible to generate

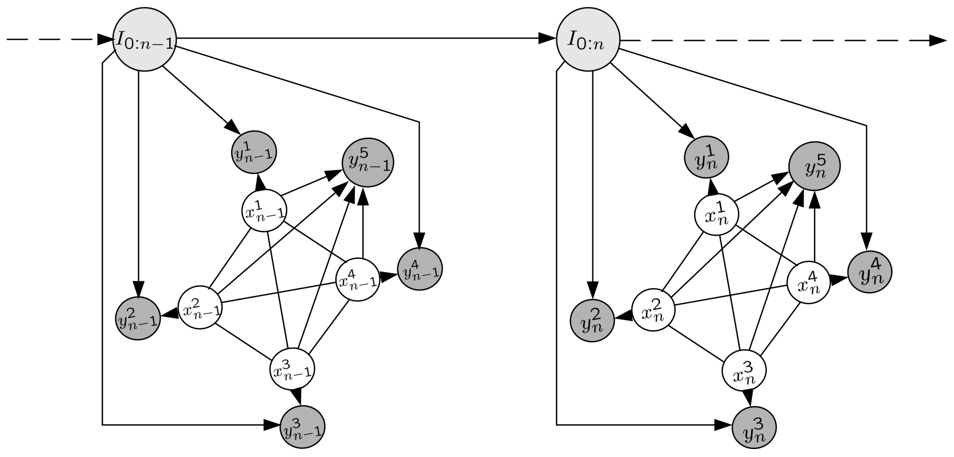


Figure 6.9: Graphical model for interactive multi-part tracking with local patch trackers and global color based likelihood.

ground truths in long sequences. Secondly, the amount of interactive effort required to generate a particular track depends on the task in hand, the sought precision, the nature of the underlying algorithm and the easy of interaction. Due to these reasons, only a qualititative assessment of the results are made and argued that it is still beneficial to introduce task specific user interactions.

6.3.2 Extended probabilistic graphical model and inference

Employing the notations defined in Sec. 6.2.1, the joint hidden state is denoted as $X_n = \{x^i_n, i\epsilon\mathcal{V}\}$ and the associated measurement as $Y_n = \{y^1_n, \ldots, y^5_n\}$, in which $y^1_n, \ldots, y^4_n$ are measurements derived from patch tracking, as before, and y^5_n is a color based measurement associated to the joint hidden state (see [Perez et al., 2002] for definition). $I_{0:n}$ is the variable representing sequential image data upto instant n. With these definitions the filtering distribution of X_n is given as follows.

$$p\big(X_n|Y_n, I_{0:n}; \theta_n, \mathcal{P}_{\mathcal{R}}\big) \propto l_C\big(y^5_n|X_n, I_{0:n}\big)\prod_{i\epsilon\mathcal{V}} l\big(y^i_n|x^i_n, I_{0:n}\big) l_{\mathcal{P}}\big(X_n|\mathcal{P}_{\mathcal{R}}\big) \prod_{(i,j)\epsilon\Gamma} \Psi\big(x^i_n, x^j_n|\theta_n\big), \tag{6.3.1}$$

where parameters $\theta_n, \mathcal{P}_{\mathcal{R}}$ have the same meaning as in Eqn. 6.2.1. The procedure for approximating this filtering distribution by a sample-set using iterative Monte Carlo simulations are on the same lines as described in Sec. 6.2.2. If the color based likelihood component $l_C\big(y^5_n|X_n, I_{0:n}\big)$ is set aside, the intermediate model used to develop the proposal density remains the same as in Eqn. 6.2.7. Thereby, the proposal density for drawing joint hidden state samples remains the same too. The only key difference lies in the additional color based likelihood evaluation required to compute the importance weights for each joint hidden state sample. Compactly, if at iteration k, joint hidden state samples are drawn as follows,

$$X_n^s \sim q\big(X_n|Y_n, I_{0:n}; \theta_n^{k-1}\big) \equiv \prod_{i\epsilon\mathcal{V}} \mathbf{b}_{k-1}\big(x_n^i\big),\ s = 1\ldots M, \tag{6.3.2}$$

then,

$$w_{k-1}^s = \frac{l_C\big(y_n^5|X_n^s\big)\prod_{i\epsilon\mathcal{V}} l\big(y_n^i|x_n^{i,s}, I_{0:n}\big) l_{\mathcal{P}}\big(X_n^s|\mathcal{P}_{\mathcal{R}}\big)\prod_{(i,j)\epsilon\Gamma}\Psi\big(x_n^{i,s}, x_n^{j,s}|\theta_n^{k-1}\big)}{\prod_{i\epsilon\mathcal{V}}\mathbf{b}_{k-1}\left(x_n^{i,s}\right)}, \tag{6.3.3}$$

giving,

$$p\big(X_n|Y_n, I_{0:n}; \theta_n^{k-1}, \mathcal{P}_{\mathcal{R}}\big) \approx \sum_{s=1:M}\frac{w_{k-1}^s}{\sum_{s=1:M} w_{k-1}^s}\delta_{X_n^s}\big(X_n\big). \tag{6.3.4}$$

The color based likelihood *function* is introduced to act as an *additional verification factor* through which each sample's weight is influenced by a globally determined term (a function of X_n). The other intention behind this term is a hope to minimize tracking drift to the extent possible. The form of this function is taken from [Perez et al., 2002], from which it is based on the Euclidean distances between RGB color space histograms as shown below.

$$l_C\big(y_n^5|X_n^s\big) = \exp -\varsigma\|H\big(X_n^s\big) - H_{ref}\|_B, \tag{6.3.5}$$

where, ς is an empirical control weight (see [Perez et al., 2002] for details), $H\big(X_n^s\big)$ is the color histogram derived from the image data inside the bounding box set around sample X_n^s, H_{ref} is the reference color histogram of the target built at the initialisation stage and $\|.\|_B$ is the Bhattacharyya distance used in Mean-shift iterations [Comaniciu et al., 2000].

6.3.3 The case for simple interactions

In spite of some encouraging results presented in Sec. 6.2.4, unsupervised online learning (update) of the prior remains difficult due to inconsistencies in measurements (tracking unwarranted local patches or patches on the background providing false notions of high confidence), lack of a smoothness prior on the hidden states and on the parameters of the potentials itself (including them makes inference more complex). Under these conditions, a supervised update of the prior can be leveraged, if circumstances are favourable. To do so, while endowing the operator with an intuitive feel for the interactions, would require a convenient form for the prior. A parametric form is a good

Algorithm 5: Interactive multi-part layout based tracking

```
input    : Video sequence with $L$ frames and local patch bounding boxes $\{\mathcal{B}_i, i \in \mathcal{V}\}$ in first frame;
Initialization;
  InitializePatchTrackers( ); /* See Sec.6.2.3.1 */
  $\mathcal{P}_{\mathcal{R}} \leftarrow$ SetLayoutReferencePolygon$\Big(\{\mathcal{B}_i, i \in \mathcal{V}\}\Big)$; /* Set the parameters of the Gaussian prior */
  $\{c_i, i \in \mathcal{V}\} \leftarrow$ BoundingBoxCenters$\Big(\{\mathcal{B}_i, i \in \mathcal{V}\}\Big)$;
  $\theta_0 \equiv \Big\{\mu_0^{ij} = |c_i - c_j|, \Sigma_0^{ij} = \text{large initial value};\ (i,j)\,\epsilon\Gamma\Big\}$; /* See Sec.6.2.3.1 */

  for $n \leftarrow 1$ to $L$ do
      automatic_mode = true;
      interactive_mode = false;
      for $i \in \mathcal{V}$ do
          if !blocked then
              $\mathcal{N}\big(y_n^i; x_n^i, \Sigma_n^i\big) \leftarrow$ TrackLocalPatch$\big(i, I_{0:n}\big)$;
          end
          else
              $\mathcal{N}\big(y_n^i = y_{n-1}^i; x_n^i, \Sigma_n^i = \infty\big)$;
          end
      end
      $\theta_n^0 = \theta_{n-1}$
      for $k \leftarrow 0$ to $K$ do
          $p\big(X_n|Y_n, I_{0:n}; \theta_n^k, \mathcal{P}_{\mathcal{R}}\big), \theta_n^{k+1} \leftarrow$ MCUpdate$\Big(\big\{\mathcal{N}\big(y_n^i; x_n^i, \Sigma_n^i\big), i \in \mathcal{V}\big\}, \theta_n^k\Big)$; /* See Alg.  4
          */
      end
      interactive_mode $\leftarrow$ PingForInteraction( ); /* Check for any user interaction */
      if interactive_mode then
          PauseTracking( );
          automatic_mode = false;
          if layout_distortion_is_adjudged_true then
              $n = n - F \leftarrow$ RetractFrames($F$); /* Requires bookeeping of past layout estimates
              */
              $\theta_{n-F} \leftarrow$ RetrievePrior($F$); /* Requires bookeeping of past prior parameters */
              $\theta_{n-F} \leftarrow$ UpdatePriorInteractively($\theta_{n-F}$); /* Mean update can also include feature
              point replacements in modified vertices
          end
          else
              $\theta_n \leftarrow$ UpdatePriorInteractively($\theta_n^K$);
              if partial_occlusion_is_adjudged_true then
                  Vertex indices $\leftarrow$ BlockMeasurements( ); /* Measurements from atmost half the
                  vertices can be blocked */
              end
          end
          interactive_mode = false;
          output    : Interactively modified layout;
      end
      else if automatic_mode then
          for $i \in \mathcal{V}$ do
              if found_outliers then
                  ReplaceOutliers( );
              end
          end
          output    : Layout $\equiv \mathbf{E}\big[p\big(X_n|Y_n, I_{0:n}; \theta_n^K, \mathcal{P}_{\mathcal{R}}\big)\big]$;
      end
  end
```

choice. For instance, if the prior is in the form of a Gaussian, as in Eqn. 6.3.1, then only the sufficient statistics (mean and co-variance) need to be interactively controlled to affect the entire prior. In general, exponential distributions are convenient as they can be fully specified by a finite number of moments.

While viewing the results of multi-part layout based tracking online the operator can interactively control the co-variance of the Gaussian prior via a simple graphical interface (a single sliding bar is enough if the Gaussian is symmetric or a graphic equalizer otherwise) to directly balance the contributions of the measurements versus the layout prior. For instance, the operator could reduce the *inertia to change* introduced by the layout prior to follow scale changes of the target at the correct rate. Differently, the operator could sustain the rigidity of the layout by tilting the balance in favour of the layout prior to avoid unnecessary distortions (example, in the case of rotations in depth).

If the layout begins to distort unnecessarily, then the operator can stop the tracking process, reset a vertex of the layout and recommence tracking. This would directly change the mean of the Gaussian prior. Indeed these operations are simple and could be applied to almost any tracking scheme, but it is more fruitful to apply these in a high dimensional tracking setup like the one proposed here as several target attributes like position, scale, orientation or even a rough shape can be extracted together. In addition under partial occlusions which pose a difficult challenge to most tracking schemes, measurements from occluded parts can be interactively "blocked" from the inference process via a single click for as long as it is deemed necessary and re-included at a later period. Overall these highlights argue in favour of interactive multi-part tracking as an attractive proposition. To summarise the discussions so far in this section, a pseudo code for interactive multi-part tracking is presented in Algorithm 5.

6.3.4 Results and discussions

The result samples in Fig.6.10 give an insight into the possible form of tracks that could be obtained from "operator in the loop" multi-part tracking. This non-standard geometrical layout track is generated with 15 mouse clicks and captures the variations of the layout throughout the sequence. The red boxes around the *new location* of the vertices in Figs. 6.10(c),(d), (f), (j) indicate user interaction used to modify the vertex from its existing position (the position of the layout before this modification is also superimposed on the samples for comparison). The blue box in Figs.6.10 indicates the

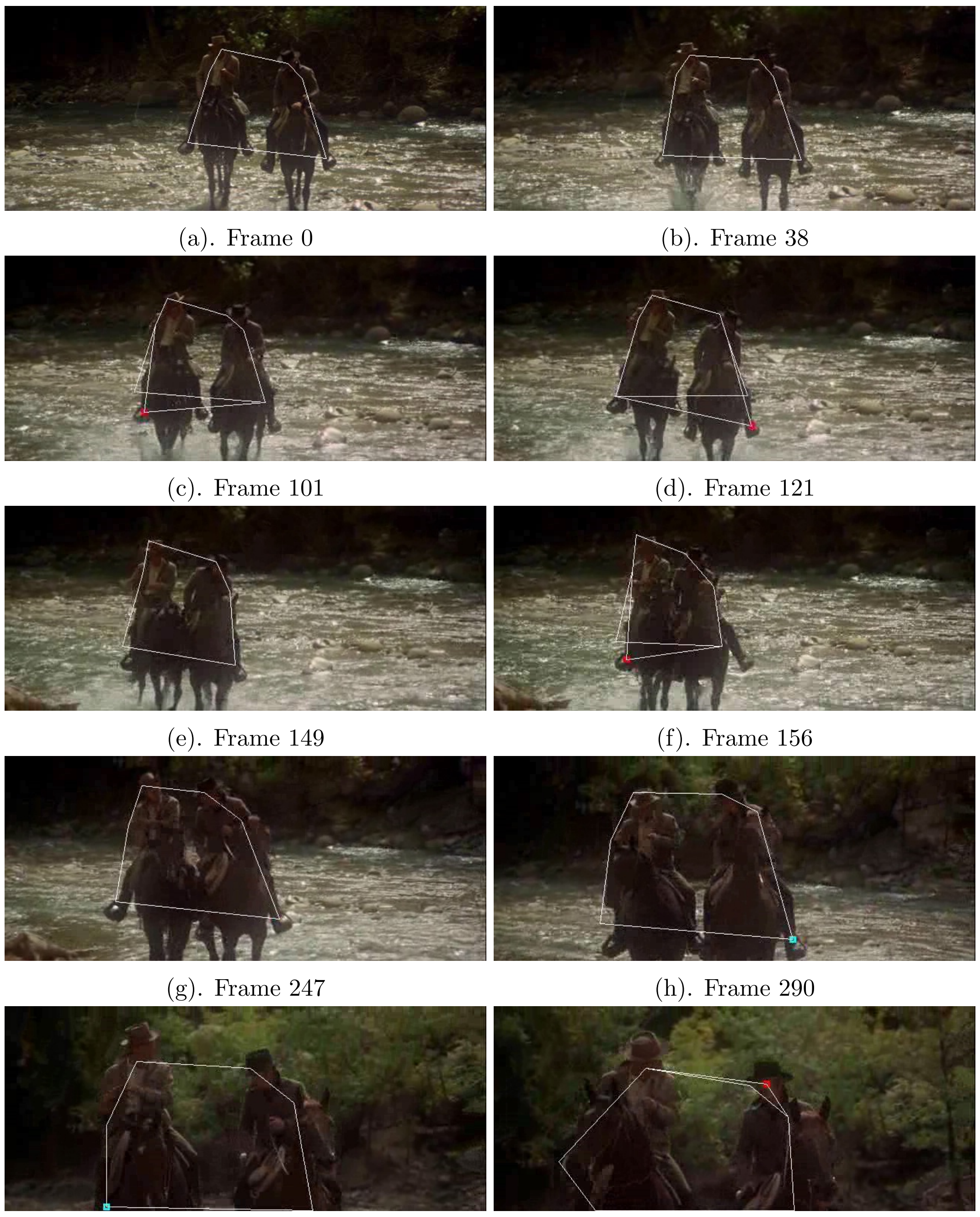

(a). Frame 0 (b). Frame 38
(c). Frame 101 (d). Frame 121
(e). Frame 149 (f). Frame 156
(g). Frame 247 (h). Frame 290
(i). Frame 338 (j). Frame 362

Figure 6.10: Interactive multi-part layout based tracking on the Butch Cassidy sequence. A red box around a vertex represents the interactively modified location of the vertex. A blue box around a vertex implies the measurements arising from that vertex is temporarily not considered during inference. In all there were 15 clicks used to generate this layout track of length 370 frames.

fact that the user signals to the inference mechanism to disregard the measurements arising from these vertices. It can be noticed that the vertices which require maximum interaction lie roughly on the feet of these cowboys which are quite often disturbed by splashes of water from the river, thus hightening the confusion. Hence blocking the measurements from these vertices (equivalent to tracking these vertices) is often advantageous and due to belief propagation predictions of the location of these vertices can be obtained.

6.4 Conclusion

Traditionally, multi-part tracking is associated to tracking, say, different parts of the human body, such as the limbs, torso, head and hands. In contrast, the contributions of this chapter equates multi-part tracking to multiple local patch tracking, wherein the patches need not necessarily correspond to semantic parts of an object. In doing so, the tracking problem is brought into a more general setup where it becomes necessary to somehow update online the prior on the hidden states. This is indeed a difficult problem but nevertheless one having very broad implications for several applications. The multi-part tracking scheme presents a Monte Carlo simulation based technique for both layout tracking and prior update. The results of experiments on this tracker encourages further work in this direction. To start with, the graphical model from which this multi-part scheme is derived is a simple one, lacking a kinematic prior on the hidden states and a prior on the parameter θ. Alternately, the θ update problem can indeed be turned into a filtering problem by including θ as a hidden variable in the model with suitable priors. In a similar vein a kinematic prior can be imposed on the hidden states to smooth the variations of the layout, but this carries an additional burden for inference due to the inclusion of temporal loops. Attempting message passing [Yedidia et al., 2001] for inference on such models is not viable due to a combinatorial explosion in the number of message paths introduced by sequential addition of layers of hidden states. Some other alternatives like variational approximations must be invoked to simplify the inference in such cases. Thereby this multi-part tracking problem is also an active platform to attempt different inference techniques.

The second prospective development comes from the direction of interactive tracking. An insight into fruitful possibilities arising from interactions with multi-part tracking was provided in this chapter. The interaction directly modifies the layout prior to steer the layout when necessary and control its distortion. The layout tracks can be quickly obtained without any preprocessing of the input sequence and this utility is handy for many vision tasks. Therefore, it is hoped that all these prospects would motivate further research into these and related directions.

7

Conclusion

This manuscript presented several probabilistic graphical models for visual tracking in low and higher dimensional spaces. A highlight of the tracking schemes based on these models was their genericity, that is, no knowledge of the kind or class of the tracked object was used, thereby broadening their scope. The requirements of the industry too demanded that the algorithms be kept general enough to be moulded into various applications. It is of the belief now that the algorithms presented in this thesis indeed can be applied for tasks such as color correction, matting, object highlighting and for developing video collages. Aside from the specific vision related applications the more abstract graphical models proposed in this thesis are themselves of sufficient interest.
For low dimensional tracking, the probabilistic message switching and combination strategy has met with considerable success for tracking generic objects in complex videos. Based on the use of parametric priors, these models are able to discard inconsistent measurements via consensus arguments and reinclude them whenever necessary to achieve stable long duration tracking. The algorithms based on these two strategies outperformed several well known tracking methods in current day literature. The success of these models emphasizes the need to integrate not just multiple (complementary) cues but also multiple filters. It demonstrated the need to leverage the benefits of both an enhanced, part wise evolving target reference model and various kinematic priors.
A part of this work also bridged the gap between key deterministic tracking approaches like point trackers and probabilistic filters by casting them as psuedo simulation filters. In the process a tracker termed the randomized feature point tracker was developed, which in essence embodies the idea behind the fusion of multiple filters through message switching. This pseudo generative viewpoint can lead to new and more robust

formalisms of point tracking itself in the near future.

The multi-part tracking scheme presented here is an example of tracking in higher dimensional spaces. The filtering problem tackled in this context is a very general one: filtering in high dimensional state space with partially unknown process model (the parameters are unknown). Several difficulties need to be overcome to present a solution to this problem. First is to simulate the high dimensional state since analytical approaches are rendered intractable due to the structure of interconnections. The second problem is to update the process prior (that is, the layout prior) online in an unsupervised fashion. Third is to design a local patch tracker which is flexible enough to accomodate changing appearances of parts of the target. A solution based on a combination of importance sampling and a Monte Carlo iterative conditional estimate technique was proposed to deal with these problems. Although the results were promising it revealed the need to solve this problem in an entirely Bayesian setting (without resorting to heuristic parameter update techniques) and develop message passing techniques for sequential loopy Bayesian networks. Several prospects exist in this direction, including the introduction of interactive aid, as discussed in this thesis, to obtain quick and reliable tracks of arbitrarily shaped (including semantically complex) targets.

In the current day visual tracking is an active research problem, either studied individually or in conjunction with other problems such as video object segmentation. The tracking problem has wide scope and applications and contrary to some ill-conceived beliefs that trackers can be replaced by detectors, it must be researched further, incorporating new developments in computer vision. It is hoped that techniques, methods and models presented in this thesis have spawned some new avenues of research into visual tracking which will be actively pursued in the near future.

8

Prospects

A brief description of some prospective extensions to the graphical models presented in this thesis are presented below.

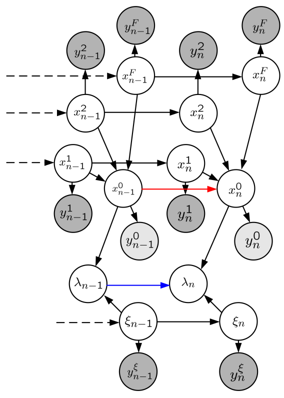


Figure 8.1: Graphical model for multi-cue fusion. λ_n is the *fused* hidden state with kinematic priors.

Tracking in low dimensional state spaces

Some interesting propositions are briefly enumerated below.

1. **Extending multi-cue message switch/combination based tracking**
   The graphical model shown in Fig. 8.1 is an extension of the multi-cue switching/combination model introduced in Chapter 5. In comparison, additional links connect the fused hidden state λ_n from one instant to the next, thereby adding a kinematic smoothness prior to it. A similar kinematic smoothness prior is added to the hidden state x_n^0, along with a measurement y_n^0 of some convenient (and complementary) form. Due to the form of this graph, the posterior distribution

of x_n^0 is composed by a product of its predictive prior, y_n^0 and the sum of messages sent from the individual feature point trackers. This has the effect of guiding the randomized feature point tracker (RFT-filter) globally. The smoothness prior for x_n^0 can be parameterized by, say, local optic flow measurements to prevent unwarranted tracking drift causing erroneous replacement of feature point trackers. An interesting fact to note from this model is that inclusion of additional links do not affect the messages sent from the point trackers to x_n^0. Here again the idea is to take a balanced risk when introducing a new prior and measurement at x_n^0 in *parallel* with the existing point tracker measurements. In contrast, an hierarchical approach to controlling the kinematic prior of point trackers by, say, global optic flow measurements may lead to undesirable results if the control measurement (optic flow) is itself inconsistent.

Although a prospective model, a few questions need to be answered to create a robust tracker from this extended model: what should be the nature of the measurement y_n^0 which makes it complementary to both color and local template based measurements?. Can local optic flow measurements be relied upon to parameterize the smoothness prior for x_n^0?. If not, then can other forms of measurements be used to develop the smoothness prior.. The answers to these questions can help enhance the robustness of tracking schemes.

2. **Pseudo Data Association Based Approach**

   This approach has the flavor of a typical multi-target tracking method. In that, let a target be sensed by N observers, each with possibly a different way (cue) to sense. Denote $Y_n = \left\{y_{1:n}^1, \dots y_{1:n}^N\right\}$, the collection of measurements from all the observers. Consider the following filtering law, assuming the measurements at each observer is independent from the rest.

$$p(x_n|Y_n) \propto \prod_{i=1}^{N} \left[\beta_{occ}^i + \beta_{val}^i p(y_n^i|x_n)\right] p(x_n|Y_{n-1}); \tag{8.0.1}$$

   where β_{occ}^i is the probability that the target is *undetected or occluded* at observer i (this can be seen as a prior, see [Vermaak et al., 2005] for an example) and β_{val}^i is the posterior probability of a *valid* measurement at observer i. This posterior probability can be evaluated as shown below.

$$p(\beta_{val}^i|Y_n) \propto p(\beta_{val}^i) \int p_{\beta_{val}^i}(y_n^i|x_n) p(x_n|Y_{n-1}) dx_n. \tag{8.0.2}$$

It can be seen from the equation above that the first term in the proportionality is the model evidence of the observation model (cue) at that observer. As a note of interest, this method is a *pseudo multi target tracking and data association* formulation, wherein a single target is sensed by several observers each making only one measurement.

This formulation is an advancement over likelihood factorisation (see Chapter 5) as the filtering distribution is effectively evaluated on *hindsight* taking into account the balancing probabilities (model evidence), thereby maintaining the possibility to down weight diffuse measurements. It is however simplified (all the fusion is subsumed in the likelihood stage itself) as compared to a more general model utilising multiple interconnected hidden states, each associated with a different cue and belonging to different subspaces.

3. **Co-dependent particle filters**

In this thesis two robust trackers were developed on the basis of the graphical model presented in Sec. 5.2. Setting aside their differences, there methods share some commonalities. To recapitulate, all point trackers utilise their own prior as proposal densities. When the RFT-filter is consistent, outlier point trackers are replaced with draws from the filtering distribution of x_n^0 (or λ_n) or when inconsistent, all point trackers are replaced with draws from the filtering distribution of ξ_n. In the same spirit, when consistent, the color based particle filter utilises its effective prior as the proposal density for importance sampling and if not, the filtering distribution of λ_n is used as the guiding proposal density. In these trackers, interactions between the RFT-filter and the color based particle filter only occur when an inconsistency is signalled in one of the filters. This abrupt form of interaction can, in principle, be replaced by a more subtle form of continuous interaction between the constituent filters. Such a strategy is termed the co-dependent particle filter.

The co-dependent particle filtering strategy is based on a *proposal density level interaction* between filters. This approach has the virtue of hindsight for importance sampling which is very useful to disassociate or weigh down uninformative models instantaneously. This is key to robust tracking. In contrast to the multi-cue switch tracker or multi-cue combination tracker the filters interact at each instant. At any instant, the level of dependence of a filter on another depends on their *figures of merit* at that instant. Depending on the figure of merit, the filter lets itself be guided, partially by its own prior and partially by the other filter. This manifests as a continuous correction process, instead of a correct-at-failure strategy used in the earlier propositions. The reader is referred to Appendix 9 for relevant mathematical formulations.

If trackers based on this co-dependent particle filtering strategy are to be constructed, it demands an ability to evaluate posterior distributions at arbitrary sample locations. This is indeed difficult when such distributions are represented as sample sets. Kernel based representations of posteriors [Han et al., 2005] are more suited to such a task.

Tracking in high dimensional state spaces

The graphical model shown in Fig. 8.2 is a generalization of the multi-part model

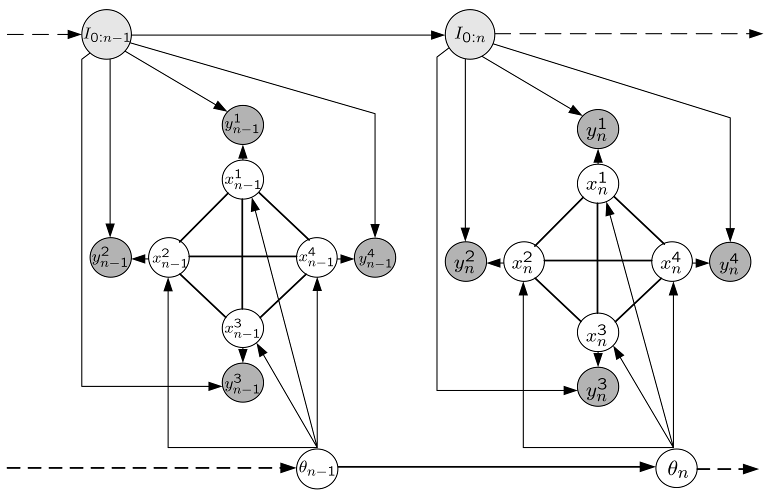


Figure 8.2: Graphical model for multi-part tracking. θ_n is the hidden variable parameterizing the layout prior.

introduced in Chapter 6. The parameter θ_n controlling the process prior, equivalently the layout prior, is now introduced as a hidden variable in the model. This is motivated by the fact that heuristic parameter update of θ_n (based on maximum likelihood formulations) leads to inconsistent updates and secondly, it is prudent to introduce a smoothness prior on the parameter itself to avoid drastic changes. In doing so, the filtering problem is extended to include the parameter θ_n as an additional hidden state with its own process model, thus increasing the dimensionality of the problem.

The potential advantage of this model over the earlier one (see Chapter 6) is the complete Bayesian formulation of the multi-part tracking problem with online update (filtering) of the process model parameters. Formulated as such, inconsistencies such as

$\theta_n \rightarrow 0$ which arise in maximum likelihood updates (see [Minka, 2001]) can be avoided by introducing a prior on the parameter θ_n. This in turn can avoid unwarranted updates of the layout prior which could lead to inconsistent updates of the patch reference models (by feature point replacement). Overall it is safe to expect a better performance over the tracker proposed in Chapter 6.

9

Appendix

Co-dependent particle filters - general formulation

Consider the following two state space models in space S.

$$\begin{aligned} M1 : x_n &\sim d_1\big(x_n|x_{n-1}\big) \\ y_n &\sim l_1\big(y_n|x_n\big); \end{aligned} \tag{9.0.1}$$

$$\begin{aligned} M2 : x_n &\sim d_2\big(x_n|x_{n-1}\big) \\ y_n &\sim l_2\big(y_n|x_n\big), \end{aligned} \tag{9.0.2}$$

where d_1, d_2 are first-order Markovian kinematic priors and l_1, l_2 are data likelihoods of models $M1$ and $M2$ repectively. At instant $n-1$, let the posterior distributions *tracking* the evolution of the state x_n with each of the above models be approximated by *sample-sets* as shown below.

$$\begin{aligned} p_1\big(x_{0:n-1}|y_{1:n-1}\big) &\approx \left\{w^i_{1,n-1}, x^i_{0:n-1}\right\}, i \in 1 \dots N_1 \\ p_2\big(x_{0:n-1}|y_{1:n-1}\big) &\approx \left\{w^j_{2,n-1}, x^j_{0:n-1}\right\}, j \in 1 \dots N_2, \end{aligned} \tag{9.0.3}$$

where N_1, N_2 are the number of samples (particles) in each approximation respectively. Associate two *normalized measures of certainty* or *figures of merit* $\{\varphi^1_n, \varphi^2_n\}\epsilon[0,1]$ to each of the above posteriors respectively. These measures could, for instance, be derived in some empirical fashion, a general form of which is indicated below.

$$\varphi_n^1 = \frac{C_1\Big[p_1\big(x_{0:n-1}|y_{1:n-1}\big)\Big]}{C_1\Big[p_1\big(x_{0:n-1}|y_{1:n-1}\big)\Big] + C_2\Big[p_2\big(x_{0:n-1}|y_{1:n-1}\big)\Big]}$$

$$\varphi_n^2 = \frac{C_2\Big[p_2\big(x_{0:n-1}|y_{1:n-1}\big)\Big]}{C_1\Big[p_1\big(x_{0:n-1}|y_{1:n-1}\big)\Big] + C_2\Big[p_2\big(x_{0:n-1}|y_{1:n-1}\big)\Big]}, \tag{9.0.4}$$

where C_1, C_2 are some scalar measures of *peakedness* (determinant or trace of covariance matrix is an example) of the two densities respectively. With these definitions, the evolution of the posterior distributions is considered in the following.

Filtering

The evolution of the density p_1 is studied as a demonstration. Let p_1 be expressed in a *partitioned* form as a two component mixture shown below.

$$p_1\big(x_{0:n-1}|y_{1:n-1}\big) \approx \sum_{m=1}^{2} \pi_{n-1,m} p_{m,1}\big(x_{0:n-1}|y_{1:n-1}\big), \tag{9.0.5}$$

where $\pi_{n-1,m}$ are the *normalized mixture weights* and their respective component densities are approximated by sample-sets as shown below.

$$p_{m,1}\big(x_{0:n-1}|y_{1:n-1}\big) \approx \left\{w_{n-1}^i, x_{0:n-1}^i\right\}, i \in I_m, \tag{9.0.6}$$

with the cardinality of the partitions satisfying $|I_1| = \varphi_n^1 N_1$ and $|I_1| + |I_2| = N_1$. With the arrival of data at instant n, each of the partitions I_m is subject to a different importance sampling based filtering procedure. The weight update for the importance samples in each of the partitions is carried out as follows.

$$\tilde{w}_n^i \propto \begin{cases} w_{n-1}^i \dfrac{l_1\big(y_n|x_n^i\big)d_1\big(x_n^i|x_{n-1}^i\big)}{q_1^1\big(x_n^i|x_{n-1}^i, y_n\big)}; i\epsilon I_1 \\ \dfrac{l_1\big(y_n|x_n^i\big)d_1\big(x_n^i|x_{n-1}^i\big)p_1\big(x_{0:n-1}^i|y_{1:n-1}\big)}{q_1^2\big(x_{0:n}^i\big)}; i\epsilon I_2, \end{cases} \tag{9.0.7}$$

where the proposal densities are developed as shown below.

$$\begin{aligned} q_1^1 &\sim d_1\big(x_n|x_{n-1}\big), \\ q_1^2 &\sim d_2\big(x_n|x_{n-1}\big)p_2\big(x_{0:n-1}|y_{0:n-1}\big). \end{aligned} \tag{9.0.8}$$

Given the unnormalized weights for each partition, consistent mixture weights $\pi_{n,m}$ are computed as follows:

$$\pi_{n,m} = \frac{\pi_{n-1,m} \sum_{i \epsilon I_m} \tilde{w}_n^i}{\pi_{n-1,1} \sum_{i \epsilon I_1} \tilde{w}_n^i + \pi_{n-1,2} \sum_{i \epsilon I_2} \tilde{w}_n^i} \tag{9.0.9}$$

With these mixture weights the posterior is as shown below.

$$p_1\big(x_{0:n}|y_{1:n}\big) \approx \sum_{m=1}^{2} \pi_{n,m} p_{m,1}\big(x_{0:n}|y_{1:n}\big), \tag{9.0.10}$$

with each of the mixture components approximated as follows;

$$p_{m,1}\big(x_{0:n}|y_{1:n}\big) \approx \sum_{i \in I_m} \frac{\tilde{w}_n^i}{\sum_{i \epsilon I_m} \tilde{w}_n^i} \delta_{x_{0:n}^i}\left(x_{0:n}\right). \tag{9.0.11}$$

Following this the measure φ_n^1 can be computed using any one of the chosen forms. For the form described in Eqn. 9.0.4, the certainty measure is computed using *unnormalized* importance weights as follows.

$$\varphi_n^1 \propto C_1\left[\left\{\frac{\tilde{w}_n^i}{\sum_{i=1:N_1} \tilde{w}_n^i}, x_{0:n}^i\right\}, i = 1 : N_1\right]. \tag{9.0.12}$$

A similar partition respective sampling based scheme is adopted to evolve the density p_2 over to instant n, following which the measure φ_n^2 is computed appropriately. In accordance with these measures the posterior distributions are repartitioned (demonstrated for p_1 only) as demonstrated below.

$$\begin{aligned} p_{m,1}\big(x_{0:n}|y_{1:n}\big) &\approx \sum_{m=1}^{2} \pi_{n,m} \sum_{i \in I_m} w_n^i \delta_{x_{0:n}^i}\big(x_{0:n}\big) \\ &= \sum_{i=1}^{N_1} \pi_{n,c(i)} w_n^i \delta_{x_{0:n}^i}\big(x_{0:n}\big) \\ &= \sum_{m=1}^{2} \hat{\pi}_{n,m} \sum_{i \epsilon \hat{I}_m} \hat{w}_n^i \delta_{x_{0:n}^i}\big(x_{0:n}\big), \end{aligned} \tag{9.0.13}$$

where $|\hat{I}_1| = \varphi_n^1 N_1$, $|\hat{I}_1| + |\hat{I}_2| = N_1$, $c(i) \epsilon \{1, 2\}$ and the mixture and particle weights are recomputed as shown below.

$$\hat{\pi}_{n,m} = \sum_{i \epsilon \hat{I}_m} \pi_{n,c(i)} w_n^i, \tag{9.0.14}$$

$$\hat{w}_n^i = \frac{\pi_{n,c(i)} w_n^i}{\hat{\pi}_{n,m}}. \tag{9.0.15}$$

Note that the partitioning is independent of the particle index, meaning it is arbitrary. Equivalently, the repartioning could also be performed after resampling the posteriors [Arulampalam et al., 2002] to retain particles with large weights.
At instant $n-1$;

$$x^i_{0:n-1} \sim \sum_{i=1}^{N_1} \pi_{n-1,c(i)} w^i_{n-1} \delta_{x^i_{0:n-1}}\big(x_{0:n-1}\big), i \in 1 \dots N_1,$$

$$p_1\big(x_{0:n-1}|y_{1:n-1}\big) \approx \frac{1}{N_1} \sum_{i=1}^{N_1} \delta_{x^i_{0:n-1}}\big(x_{0:n-1}\big). \tag{9.0.16}$$

Repartitioning the above approximation;

$$p_1\big(x_{0:n-1}|y_{1:n-1}\big) \approx \sum_{m=1}^{2} \hat{\pi}_{n-1,m} \sum_{i \epsilon I_m} \delta_{x^i_{0:n-1}}\big(x_{0:n-1}\big), \tag{9.0.17}$$

where $\hat{\pi}_{n-1,1} = \varphi^1_{n-1}$ and $\hat{\pi}_{n-1,1} + \hat{\pi}_{n-1,2} = 1$. From this approximation onwards the filtering steps are on the same lines as described earlier.

Bibliography

A. Blake, R. C. and Zisserman, A. (1993). A framework for spatio-temporal control in thetracking of visual contours. *IJCV*, 11(2):127–145.

Arkin, E., Chew, L., Huttenlocher, D., Kedem, K., and Mitchell, J. (1991). An efficiently computable metric for comparing polygonal shapes. *IEEE Trans. on PAMI*, 13(3):209–216.

Arnaud, E., Memin, E., and Cernuschi-Frias, B. (2004). Conditional filters for image sequence based tracking - application to point tracking. *IEEE Trans. on Image Processing*, 14(1):63–79.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/ non-gaussian bayesian tracking. *IEEE Trans. on Signal Processing*, 50(2):174–188.

Avidan, S. (2004). Support vector tracking. *IEEE Trans. on PAMI*, 8(26):1064–1072.

B. Williams, P. S. and Reid, I. (2007). Automatic relocalisation for a single-camera simultaneous localisation and mapping system. In *International Conference on Robotics and Automation, Rome.*

Badrinarayanan, V., Perez, P., Clerc, F. L., and Oisel, L. (2007a). On uncertainties, random features and object tracking. In *14th IEEE International Conference on Image Processing (ICIP), San Antonio, Texas.*

Badrinarayanan, V., Perez, P., Clerc, F. L., and Oisel, L. (2007b). Probabilistic color and adaptive multi-feature tracking with dynamically switched priority between cues. In *ICCV, Rio de Janeiro, Brazil.*

Badrinarayanan, V., Perez, P., Clerc, F. L., and Oisel, L. (2008). Geometric layout based graphical model for multi-part object tracking. In *8th Workshop on Visual Surveillance, 10th European Conference on Computer Vision (ECCV), Marseille, France.*

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

Bar-Shalom, Y. and Fortmann, T. (1988). *Tracking and Data Association.* Academic Press.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B.*, 36:192236.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

Black, M. J. and Jepson, A. (1998). Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *IJCV*, 26(1):63–84.

Blake, A., Bascle, B., Isard, M., and MacCormick, J. (1998). Statistical models of visual shape and motion. *Phil. Trans. R. Soc. Lond. A*, 356:1283–1302.

Blake, A., Isard, M., and Reynard, D. (1995). Learning to track the visual motion of contours. *Artificial Intelligence*, 78:101–133.

Boyen, X. and Koller, D. (1998). Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, Wisconsin.*

Briers, M., Doucet, A., and Singh, S. S. (2005). Sequential auxiliary particle belief propagation. In *Proc. Fusion.*

Buchanan, A. and Fitzgibbon, A. (2006). Interactive feature tracking using k-d trees and dynamic programming. In *CVPR.*

Buchanan, A. and Fitzgibbon, A. (2007). Combining local and global motion models for feature point tracking. In *CVPR.*

Collins, R. T., Zhou, X., and Teh, S. K. (2005). An open source tracking testbed and evaluation web site. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS).*

Comaniciu, D. and Meer, P. (1999). Mean shift analysis and applications. In *ICCV, Kerkyra, Greece.*

Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *CVPR.*

Comaniciu, D., Ramesh, V., and Meer, P. (2001). The variable bandwidth mean shift and data-driven scale selection. In *ICCV, Vancouver, British Columbia, Canada.*

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564 – 577.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory.* John Wiley and Sons, Inc.

Doucet, A., de Freitas, N., and Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice.* Springer.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10:197–208.

Fan, Z., Wu, Y., and Yang, M. (2005). Multiple collaborative kernel tracking. In *CVPR,San Diego,CA*.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Info. Theory*, 21:32–40.

Ghahramani, Z. and Jordan, M. I. (1995). Factorial hidden markov models. In *Proc. Conf. Advances in Neural Information Processing Systems, (NIPS), Cambridge, MA*.

Grabner, M., Grabner, H., and Bischof, H. (2007). Learning features for tracking. In *CVPR*.

Han, B., Joo, S.-W., and Davis, L. S. (2007). Probabilistic fusion tracking using mixture kernel-based bayesian filtering. In *ICCV, Rio de Janeiro*.

Han, B., Zhu, Y., Comaniciu, D., and Davis, L. (2005). Kernel-based bayesian filtering for object tracking. In *CVPR, San Diego, CA*.

Han, T. X. and Huang, T. S. (2005). Articulated body tracking using dynamic belief propagation. In *IEEE International Workshop on Human-Computer Interaction, ICCV, Beijing*.

Harris, C. and Stephens, M. J. (1988). A combined corner and edge detector. In *In Alvey Vision Conference*.

Hastings, W. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97109.

Ho, J., Lee, K.-C., Yang, M.-H., and Kriegman, D. (2004). Visual tracking using learned subspaces,. In *CVPR, Washington, DC*.

Hua, G. and Wu, Y. (2004). Multi-scale visual tracking by sequential belief propagation. In *CVPR, Washington, DC*.

Hua, G., Wu, Y., and Fan, Z. (2006). Measurement integration under inconsistency for robust tracking. In *CVPR, New York City, NY*.

Isard, M. and Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In *ECCV, Cambridge, UK*.

Isard, M. and Blake, A. (1998). A mixed-state condensation tracker with automatic model-switching. In *ICCV*.

Isard, M. and MacCormick, J. (2001). Bramble: A bayesian multiple-blob tracker. In *ICCV, Vancouver, British Columbia, Canada*.

Jaakkola, T. S. (2000). Tutorial on variational approximation methods. Technical report, MIT Artificial Intelligence Laboratory.

Jepson, A., Fleet, D., and El-Maraghi, T. (2003). Robust online appearance models for visual tracking. *IEEE Trans. on PAMI*, 25(10):1296–1311.

J.P.Lewis (1995). *Fast Normalized Cross-Correlation.* Canadian Image Processing and Pattern Recognition Society.

J.Shi and C.Tomasi (1994). Good features to track. In *CVPR, Seattle.*

Julier, S. J. and Uhlmann, J. K. (1997). A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, Florida.*

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45.

Kass, M., Witkin, A., and Terzopoulos, D. (1987). Snakes: Active contour models. *IJCV*, 1(4):321–331.

Khan, Z., Balch, T., and Dellaert, F. (2004). An mcmc-based particle filter for tracking multiple interacting targets. In *ECCV, Copenhagen.*

K.Nummiaro, Koller-Meier, E., and Gool, L. V. (2003). An adaptive color-based particle filter. *Image and Vision Computing*, 1:99–110.

Kolsch, M. and Turk, M. (2004). Fast 2d tracking with flocks of features and multi-cue integration. In *CVPR Workshop.*

Lee, K.-C., Ho, J., Yang, M.-H., and Kriegman, D. (2005). Visual tracking and recognition using probabilistic appearance manifolds. *CVIU*, 99:303–331.

Leichter, I., Lindenbaum, M., and Rivlin, E. (2006). A generalised framework for combining visual trackers - the "black boxes approach". *IJCV*, 67(3):343–363.

Leichter, I., Lindenbaum, M., and Rivlin, E. (2007). Visual tracking by affine kernel fitting using color and object boundary. In *ICCV, Rio de Janeiro, Brazil.*

Li, Y., Ai, H., Yamashita, T., Lao, S., and Kawade, M. (2007). Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans. In *CVPR, Minneapolis.*

Liu, X. (2007). Generic face alignment using boosted appearance model. In *CVPR, Minneapolis.*

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110.

Mackay, D. (1998). Introduction to monte carlo methods. In *Learning in Graphical Models.*

Malladi, R., Sethian, J. A., and Vemuri, B. (1995). Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:158–175.

Minka, T. (1999). From hidden markov models to linear dynamical systems. Technical report, MIT.

Minka, T. (2004). Exemplar-based likelihoods using the pdf projection theorem. Technical report, Microsoft Research Cambridge.

Minka, T. P. (2001). Pathologies of orthodox statistics. Technical report, MIT.

Moreno-Noguer, F., Sanfeliu, A., and Samaras, D. (2008). Dependent multiple cue integration for robust tracking. *IEEE Trans. on PAMI*, 30(4):670–685.

Nguyen, Q. A., Robles-Kelly, A., and Shen, C. (2007). Kernel-based tracking from a probabilistic viewpoint. In *CVPR, Minneapolis,MN*.

Nickels, K. and Hutchinson, S. (2002). Estimating uncertainty in ssd-based feature tracking. *Image and Vision Computing*, 20:47–68.

Osher, S. and Sethian, J. A. (1988). Fronts propagating with curvature dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49.

Pao, L. Y. (1994). Multisensor multitarget mixture reduction algorithms for tracking. *Journal of Guidance, Control and Dynamics*, 17:1205–1211.

Papoulis, A. and Pillai, S. U. (2002). *Probability, Random Variables and Stochastic Processes*. McGraw Hill.

Patras, I. and Hancock, E. R. (2007). Regression tracking with data relevance determination. In *CVPR, Minneapolis*.

Pearl, J. (1997). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kauffman.

Perez, P., Hue, C., Vermaak, J., and Gangnet, M. (2002). Color-based probabilistic tracking. In *ECCV, Copenhagen, Denmark*.

Perez, P., Vermaak, J., and Blake, A. (2004). Data fusion for visual tracking with particles. *Proc. IEEE*, 92(3):495–513.

Pilet, J., Lepetit, V., and Fua, P. (2005). Real-time non-rigid surface detection. In *CVPR, San Diego, CA*.

Pitt, M. K. and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94:590–599.

Qi, Y., Szummer, M., and Minka, T. P. (2005). Bayesian conditional random fields. In *Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS), Barbados*.

Ramanan, D. and Forsyth, D. (2003). Finding and tracking people from the bottom up. In *CVPR, Madison, Wisconsin*.

Rasmussen, C. and Hager, G. D. (1998). Joint probabilistic techniques for tracking multi-part objects. In *CVPR, Santa Barbara, California.*

Rasmussen, C. and Hager, G. D. (2001). Probabilistic data association methods for tracking complex visual objects. *IEEE Trans. on PAMI*, 23(6):560–576.

Reid, I. D. and Murray, D. W. (1996). Active tracking of foveated feature clusters using affine structure. *IJCV*, 18(1):41–60.

R.J. Kauth, A. P. and Thomas, G. (1977). Blob: An unsupervised clustering approach to spatial preprocessing of manuscripts imagery. In *Proc. 11th Intl Symp. Remote Sensing of the Environment, Ann Arbor, Michigan.*

Rosenberg, Y. and Werman, M. (1997). Representing local motion as a probability distribution matrix for object tracking and other applications. In *CVPR,San Juan, Puerto Rico.*

Sakagaito, J. and Wada, T. (2007). Nearest first traversing graph for simultaneous object tracking and recognition. In *CVPR, Minneapolis, Minnesota.*

Salzenstein, F. and Pieczynski, W. (1995). Unsupervised bayesian segmentation in hidden markovian fields. In *ICASPP, Detroit.*

Sand, P. and Teller, S. (2006). Particle video: Long-range motion estimation using point trajectories. In *CVPR, New York.*

Schutter, T. L. . H. B. . J. D. (2001). Kalman filters for non-linear systems: a comparison of performance. *International Journal of Control*, 77:639–653.

Siebel, N. T. and Maybank, S. (2004). The advisor visual surveillance system. In *Workshop on Applications of Computer Vision (ACV), Prague, Czech Republic.*

Sigal, L., Isard, M., Sigelman, B. H., and Black, M. J. (2003). Attractive people: Assembling loose-limbed models using non-parametric belief propagation. In *NIPS, Vancouver, Canada.*

Song, Y., Feng, X., and Perona, P. (2000). Towards detection of human motion. In *CVPR, Hilton Head Island, South Carolina.*

Storvik, G. (1994). A bayesian approach to dynamic contours through stochastic sampling and simulated annealing. *IEEE Trans. on PAMI*, 16(10):976–986.

Sudderth, E. B., Ihler, A. T., Freeman, W. T., and Willsky, A. S. (2003). Nonparametric belief propagation. In *CVPR, Madison, Wisconsin.*

T.F.Cootes and C.J.Taylor (1992). Active shape models. In *BMVC, Leeds, UK.*

Tomasi, C. and Kanade, T. (1991). Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University.

Toyama, K. and Horvitz, E. (2000). Bayesian modality fusion: Probabilistic integration of multiple vision algorithms for head tracking. In *ACCV, Taipei, Taiwan.*

Triesch, J. and von der Malsburg, C. (2000). Self-organized integration of adaptive visual cues for face tracking. In *Proceedings of the Fourth International Conference on Automatic Face and Gesture Recognition, Grenoble, France.*

van der Merwe, R., de Freitas, J. F. G., Doucet, A., and Wan., E. A. (2000). The unscented particle filter. Technical report, University of Cambridge.

Veeraraghavan, H., Schrater, P., and Papanikolopoulos, N. (2006). Robust target detection and tracking through integration of motion, color and geometry. *CVIU*, 103:121–138.

Vermaak, J., Doucet, A., and Perez, P. (2003). Maintaining multi-modality through mixture tracking. In *ICCV*.

Vermaak, J., Godsill, S., and Prez, P. (2005). Monte carlo filtering for multi-target tracking and data association. *IEEE Trans. on Aerospace and Electronic Systems.*, 41(1):309–332.

Vermaak, J., Perez, P., Gangnet, M., and Blake, A. (2002). Towards improved observation models for visual tracking: selective adaptation. In *ECCV, Copenhagen, Denmark.*

Wan, E. A. and van der Menve, R. (2000). The unscented kalman filter for nonlinear estimation. In *IEEE Symposium on Adaptive Systems for Signal Processing, Communications and Control, Lake Louise, Alberta, Canada.*

Wang, T., Diao, Q., Zhang, Y., Song, G., Lai, C., and Bradski, G. (2004). A dynamic bayesian network approach to multi-cue based visual tracking. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR), Cambridge, UK.*

Wei, Y., Sun, J., Tang, X., and Shum, H.-Y. (2007). Interactive offline tracking for color objects. In *ICCV, Rio De Janeiro.*

Welch, G. and Bishop, G. (2001). An introduction to the kalman filter. In *ACM SIGGRAPH, Los Angeles, California.*

Willett, P., Ruan, Y., and Streit, R. (2002). Pmht: problems and some solutions. *IEEE Trans. on Aerospace and Electronic Systems*, 38(3):738–754.

Williams, O., Blake, A., and Cipolla, R. (2005). Sparse bayesian learning for efficient visual tracking. *IEEE Trans. on PAMI*, 27(8):1292–1304.

Wren, C. R., Azarbayejani, A., Darrel, T., and Pentland, A. P. (1997). Pfinder: Real-time tracking of the human body. *IEEE Trans. on PAMI*, 19(7):780–785.

Wu, Y., Hua, G., and Yu, T. (2003). Tracking articulated body by dynamic markov network. In *ICCV, Nice, France.*

Wu, Y. and Huang, T. S. (2004). Robust visual tracking by integrating multiple cues based on co-inference learning. *IJCV*, 58:55–71.

Xu, C. and Prince, J. L. (1998). Snakes, shape, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369.

Yang, M. and Wu, Y. (2005). Tracking non-stationary appearances and dynamic feature selection. In *CVPR, San Diego, CA.*

Yang, M., Wu, Y., and Lao, S. (2006). Intelligent collaborative tracking by mining auxiliary objects. In *CVPR, NewYork.*

Yedidia, J. S., Freeman, W. T., and Weiss., Y. (2001). Understanding belief propagation and its generalizations. In *IJCAI, Seattle, Washington.*

Yilmaz, A. (2007). Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection. In *CVPR, Minneapolis, MN.*

Yilmaz, A., Li, X., and Shah, M. (2004). Object contour tracking using level sets. In *ACCV, Jeju Island, Korea.*

Zhou, X. S., Comaniciu, D., and Gupta, A. (2005). An information fusion framework for robust shape tracking. *IEEE Trans. on PAMI*, 27(1):115–129.

Zhu, S.-C. (2003). Statistical modeling and conceptualization of visual patterns. *IEEE Trans. on PAMI*, 25(6):1–22.