Chapter 6

Pixel-Wise Posterior Tracking

6.1 Introduction

Not only do we require fast and reliable visual tracking for the objective of this dissertation, but it is a prerequisite for a vast number of other applications in computer vision. Though it has been the subject of intense effort over the last two decades, it remains a difficult problem for a number of reasons. In particular, when tracking previously unseen objects, many of the constraints that give reliability to other tracking systems – such as strong prior information about shape, appearance or motion – are unavailable. For further reading on the vast quantities of work relating to visual tracking refer to [Yilmaz *et al.*, 2006] and [Moeslund *et al.*, 2006][1].

One of the earliest approaches to visual tracking, originally proposed by [Lucas & Kanade, 1981], is template tracking. It has been used extensively in one form or another for over two decades (see [Baker & Matthews, 2004] for an excellent summary). The idea behind template tracking is to extract a template (a rectangular region of pixels) in one image and to warp the template into another image so that the Sum-of-Squared Differences (SSD) between pixel values is minimised. The underlying optimisation uses image derivatives to drive the warp to a local minimum. One of the key features of using a template is that it encodes both the pixel locations and colours. This makes it possible to compute the tracking parameters to sub-pixel accuracy. A significant weakness of the approach is that an accurate template must be maintained over time, which is difficult for anything but planar objects.

[1] This survey covers the period of 2000-2006 in the application area of human motion tracking. It lists more than three hundred papers.

A significant step away from template tracking was made by [Kass *et al.*, 1988]. Rather than tracking a template corresponding to an object, they tracked the boundary between the object and the background. This was achieved using an active contour to represent the boundary. One typical use of active contours is to align them to edges in the image, by using a 1D search along the direction normal to the contour (similar to the 1D search used by [Harris, 1993]). A problem with this approach is that natural images typically contain many edges that do not belong to the object of interest, which act as distractors. This was tackled by [Isard & Blake, 1998a] by using a non-parametric method, known as particle filtering, to represent multi-modal distributions. The ability to model the distractors using multi-modal distributions added robustness to the tracker, making it possible to track objects that template tracking alone would fail on.

More recently, probability distributions have been used to model an object's appearance. This is referred to as region-based tracking and is considered more robust than templates or edges for many applications. [Comaniciu *et al.*, 2000] showed how mean-shift, a 'local' non-parametric mode finding technique, could be used to align a region to the image. The objective function used for this alignment measured the similarity between the modelled appearance distribution and the observed empirical distribution. In contrast to template tracking and active contours, an elliptical region is used and rather than taking a template or using edges, the pixel values are used to build a non-parametric distribution (a colour histogram). This non-parametric distribution is somewhat weaker than a template, in the sense that it does not contain spatial information and yet captures more information than edges alone.

A key problem with both template tracking and mean-shift is how to adapt the appearance models i.e. the template or colour histogram, over time. Both methods assume a very simplistic segmentation of the object from the background, either a rectangular or elliptical region. In order to track objects for long periods of time it is crucial that the appearance model can be adapted. In all but the simplest cases this requires a good segmentation of the object from the background, otherwise pixels that belong to the background are incorporated in an object's appearance model or vice-versa. The result is that the tracker 'drifts' away from the object over a period of time and eventually fails; this is commonly referred to as tracking drift. Although active contours have a much better notion of shape than template tracking or mean-shift, they lack a region-based representation.

One region-based technique that has shown considerable promise for its ability to per-

form tracking and segmentation within a unified framework is the use of an implicit contour (level-set) to represent the boundary of the object [Osher & Paragios, 2003; Paragios & Deriche, 2000; Goldenberg *et al.*, 2001]. As well as handling topological changes seamlessly, tracking using level-sets can be couched in a fairly standard probabilistic formulation [Cremers, 2006; Cremers *et al.*, 2007], and hence can leverage the power of Bayesian methods.

In this chapter, we present a novel probabilistic framework for combined tracking and segmentation, which, as well as capturing all of the desirable properties of level-set based tracking, is very robust and runs in a few milliseconds on standard hardware. We base our framework on a generative model of image formation that represents the image as a bag-of-pixels [Jebara, 2003]. The advantage of such a model – in common with other simpler density-based representations such as colour-histograms – is the degree of invariance to viewpoint this confers.

Like [Cremers, 2006], we derive a probabilistic, region-based, level-set framework, which comprises an optimal rigid registration, followed by a segmentation to re-segment the object and account for non-rigid deformations. Aside from issues of speed (which are not addressed in [Cremers, 2006]), there are a number of key differences between [Cremers, 2006] and our work, some of which stem from the generative model we use for image data (see Section 6.3). Firstly, our derivation gives a probabilistic interpretation to the Heaviside step function used in most region-based level-set methods [Chan & Vese, 2001; Cremers, 2006]. Secondly, given this interpretation we propose a pixel-wise posterior term, as opposed to a likelihood, which allows us to marginalise out model parameters at a pixel level. As we show in Section 6.3, this derives naturally from our generative model, and is a subtle but absolutely crucial difference between our method and others e.g. [Cremers, 2006; Paragios & Deriche, 2000; Goldenberg *et al.*, 2001], as our results show in Section 6.8. Thirdly, in contrast to [Chan & Vese, 2001; Cremers, 2006] and similar to [Freedman & Zhang, 2004; Zhang & Freedman, 2005], we assume a non-parametric distribution for image values as opposed to a single Gaussian (for an entire region). The superior performance of our method is particularly noticeable when using non-parametric distributions and much less noticeable when using smooth Gaussian distributions (as presented in [Cremers, 2006]). Finally, we introduce a prior on the embedding function which constrains it to be an approximate signed distance function. We show that this gives a clean probabilistic interpretation to the idea proposed by [Li *et al.*, 2005] and avoids the need for reinitialisation of the embedding function that is necessary in the majority

of level-set based approaches.

Our work also bears some similarity to [Yilmaz, 2007], who sought the rigid transformation that best aligns a fixed shape-kernel with image data using the Bhattacharyya coefficient. This work extended the pioneering work of this type [Comaniciu *et al.*, 2000; Collins, 2003] to handle translation+scale+rotation as opposed to translation only or translation+scale. In contrast to [Yilmaz, 2007], however, we allow the shape to change online using an implicit contour and propose a novel framework using pixel-wise posteriors, which removes the cost of building an empirical distribution and testing it with the Bhattacharyya coefficient. This has a second hidden benefit as it avoids the need to build a 'good' empirical distribution given limited data; we find in practice this gives a significant improvement over [Comaniciu *et al.*, 2000; Collins, 2003; Yilmaz, 2007].

Unlike [Cremers, 2006], [Freedman & Zhang, 2004; Zhang & Freedman, 2005] use a non-parametric distribution for image data. They derive contour flows based on both KL-divergence and the Bhattacharyya coefficient. Though they demonstrate that both are effective for tracking, they do not model rigid transformation parameters explicitly. They must recompute their non-parametric distributions at every iteration, and – as we show in Section 6.8 – objectives based on the Bhattacharyya coefficient are inferior to the one we propose.

Within our framework (and other similar work), because the segmentation is performed rapidly and reliably online, the appearance and shape models of the object can be updated over time without suffering from the significant problems of tracking drift that plague other algorithms. Our framework is general enough to be extended to various types of prior information and various imaging modalities, but in this dissertation we restrict ourselves to the problem of tracking the 2D projections of either 2D or 3D objects in ordinary colour video. In summary, the key benefits of our method are: (i) an extensible probabilistic framework; (ii) robustness - given by pixel-wise posteriors and marginalisation; (iii) real-time performance; (iv) excellent cost function characteristics; (v) no need to compute empirical distributions at every frame; (vi) online learning (i.e. adaption of appearance and shape characteristics); (vii) flexibility to track many different types of object and (viii) high invariance to view and appearance changes.

The remainder of this chapter is organised as follows: Section 6.2 describes the representation of the object being tracked; Section 6.3 derives a probabilistic framework from a simple generative model; Section 6.4 outlines the level-set segmentation; Sec-

tion 6.5 shows the registration process; Section 6.6 describes our method for dealing with tracking drift; Section 6.7 outlines the online learning process; Section 6.8 shows our results and Section 6.9 concludes with a summary and discussion.

6.2 The Representation

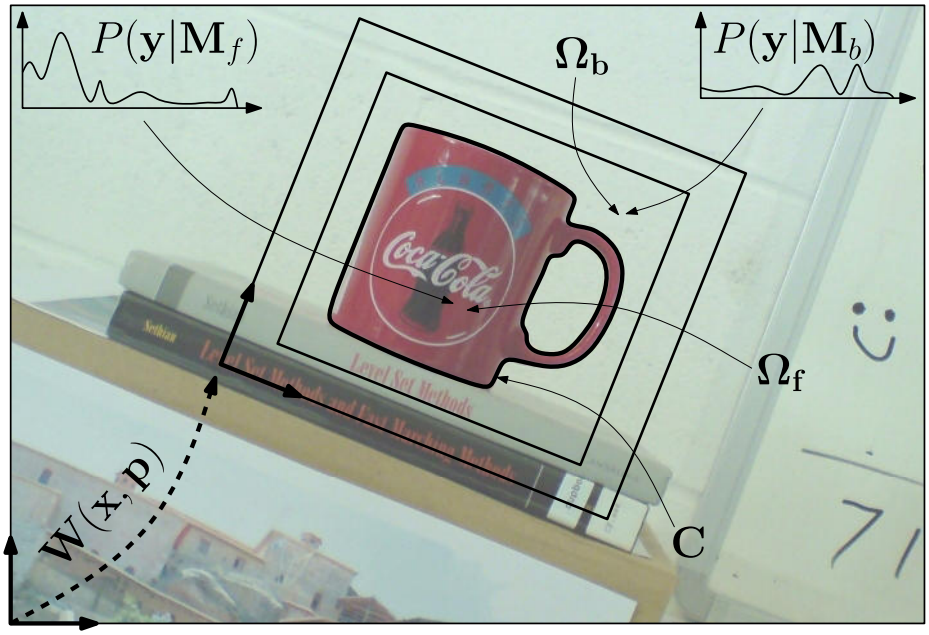

Figure 6.1: Representation of the object, showing: the contour $\mathbf{C}$, the set of foreground pixels $\mathbf{\Omega}_f$, the set of background pixels $\mathbf{\Omega}_b$, the foreground model $P(\mathbf{y}|M_f)$, the background model $P(\mathbf{y}|M_b)$ and the warp $\mathbf{W}(\mathbf{x},\mathbf{p})$.

We represent the object being tracked by its shape $\mathbf{C}$, its location in the image $\mathbf{W}(\mathbf{x},\mathbf{p})$ and two underlying appearance models: one for the foreground $P(\mathbf{y}|M_f)$ and one for the background $P(\mathbf{y}|M_b)$. Figure 6.1 illustrates this with a simple example.

Shape: is represented by the zero level-set $\mathbf{C} = \{\mathbf{x}|\mathbf{\Phi}(\mathbf{x}) = 0\}$ of an embedding function $\mathbf{\Phi}(\mathbf{x})$ [Osher & Paragios, 2003; Cremers *et al.*, 2007]. In our case, the shape $\mathbf{C}$ is a 2D contour and the embedding function $\mathbf{\Phi}(\mathbf{x})$ is a 3D function represented on a discrete grid of $\mathbf{x}$ values. Figure 6.2 shows examples of embedding functions and their corresponding level-sets for a variety of different shape contours. Given the shape contour $\mathbf{C}$, the pixels $\mathbf{\Omega}$ in the object frame are segmented into two regions: one for the foreground $\mathbf{\Omega}_f$ and one for the background $\mathbf{\Omega}_b$.

Location: is described by a warp $\mathbf{W}(\mathbf{x},\mathbf{p})$ that takes a pixel location $\mathbf{x}$ in the object frame and warps it into the image frame according to parameters $\mathbf{p}$.

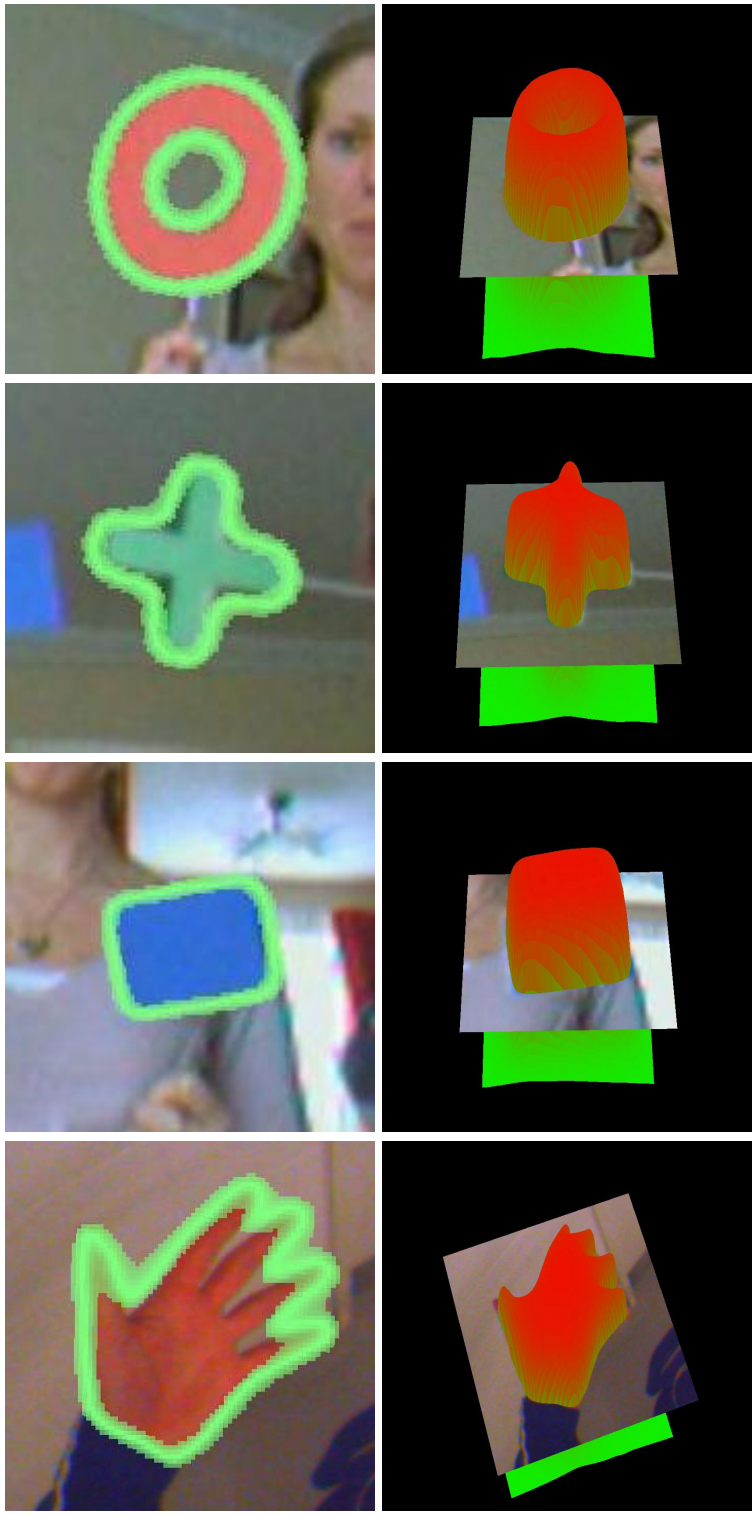

Figure 6.2: A selection of object shapes and their corresponding level-sets: (left column) shows the 2D shape contour $\mathbf{C}$ and (right column) shows the corresponding 3D embedding function $\mathbf{\Phi}(\mathbf{x})$ and the level-set $\mathbf{\Phi}(\mathbf{x}) = 0$.

Appearance models: $P(\mathbf{y}|M_f)$ and $P(\mathbf{y}|M_b)$ are represented with RGB histograms using 32 bins per channel. The histograms are initialised either from a detection module or a user inputted initial bounding box. The pixels inside the bounding box are used to build the foreground model and the pixels from an inflated bounding box are used to build the background model. The two initial distributions are then used to produce a tentative segmentation, which is in turn used to rebuild the model. This procedure is iterated until the shape converges (similar to [Rother *et al.*, 2004]). Once tracking commences, the appearance models and shape $\mathbf{C}$ are estimated (adapted) online, as described in Section 6.7.

In summary, we use the following notation:

- $\mathbf{x}$: A pixel location in the object's coordinate frame.
- $\mathbf{y}$: A pixel value (in our experiments this is a RGB value).
- $\mathbf{I}$: The image.
- $\mathbf{W}(\mathbf{x}, \mathbf{p})$: Warp with parameters $\mathbf{p}$.
- $M = \{M_f, M_b\}$: Model parameter either foreground or background.
- $P(\mathbf{y}|M_f)$: Foreground model over pixel values $\mathbf{y}$.
- $P(\mathbf{y}|M_b)$: Background model over pixel values $\mathbf{y}$.
- $\mathbf{C}$: The contour that segments the foreground from the background.
- $\mathbf{\Phi}(\mathbf{x})$: Shape kernel (in our case the level-set embedding function).
- $\mathbf{\Omega} = \{\mathbf{\Omega}_f, \mathbf{\Omega}_b\}$: Pixels in the object frame $[\{\mathbf{x}_0, \mathbf{y}_0\}, \ldots, \{\mathbf{x}_N, \mathbf{y}_N\}]$, which are partitioned into foreground pixels $\mathbf{\Omega}_f$ and background pixels $\mathbf{\Omega}_b$.
- $H_\epsilon(z)$: Smoothed Heaviside step function.
- $\delta_\epsilon(z)$: Smoothed Dirac delta function.

6.3 The Generative Model

Figure 6.3 illustrates the simple generative model we use to represent the image formation process. This model treats the image as a bag-of-pixels [Jebara, 2003] and

can, given the model M, the shape $\mathbf{\Phi}$ and the location $\mathbf{p}$, be used to sample pixels $\{\mathbf{x}, \mathbf{y}\}$.

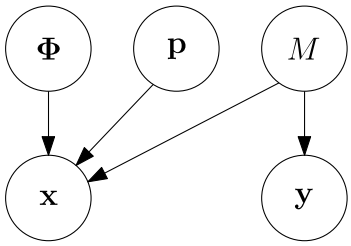

Figure 6.3: Generative model representing the image as a bag-of-pixels.

Although the resultant image would not look like the true foreground/background image to a human (the pixels would be jumbled up), the colour distributions corresponding to the foreground/background regions $\mathbf{\Omega}_f/\mathbf{\Omega}_b$ would match the models $P(\mathbf{y}|M_f)$ and $P(\mathbf{y}|M_b)$, see Figure 6.4 for an example. It is this simplicity that gives more invariance to viewpoint and allows 3D objects to be tracked robustly without having to model their specific 3D structure.

Figure 6.4: Showing the effect of using a bag-of-pixels model: (left) original image and (right) pixels sampled from the foreground and background colour distributions. Both of these images would have an equal chance of being sampled from our generative model.

The joint distribution for a single pixel given by the model in Figure 6.3 is:

$$P(\mathbf{x}, \mathbf{y}, \mathbf{\Phi}, \mathbf{p}, M) = P(\mathbf{x}|\mathbf{\Phi}, \mathbf{p}, M)P(\mathbf{y}|M)P(M)P(\mathbf{\Phi})P(\mathbf{p}). \tag{6.1}$$

We now divide (6.1) by $P(\mathbf{y}) = \sum_{i=\{f,b\}} P(\mathbf{y}|M_i)P(M_i)$ to give:

$$P(\mathbf{x}, \mathbf{\Phi}, \mathbf{p}, M|\mathbf{y}) = P(\mathbf{x}|\mathbf{\Phi}, \mathbf{p}, M)P(M|\mathbf{y})P(\mathbf{\Phi})P(\mathbf{p}), \tag{6.2}$$

where the term $P(M|\mathbf{y})$ is the pixel-wise posterior, of the model M, given a pixel value $\mathbf{y}$:

$$P(M_j|\mathbf{y}) = \frac{P(\mathbf{y}|M_j)P(M_j)}{\sum_{i=\{f,b\}} P(\mathbf{y}|M_i)P(M_i)} \qquad j = \{f, b\}. \tag{6.3}$$

Using this posterior is equivalent to applying Bayesian model-selection to each individual pixel[2]. We are interested in finding the pose $\mathbf{p}$ and the shape $\mathbf{\Phi}$, where the model parameter M can be considered a nuisance variable. The Bayesian way of dealing with nuisance variables is to marginalise them out using the sum rule. We take this approach to obtain the pixel-wise posterior of the shape $\mathbf{\Phi}$ and the location $\mathbf{p}$ given a pixel $\{\mathbf{x}, \mathbf{y}\}$:

$$P(\mathbf{\Phi}, \mathbf{p}|\mathbf{x}, \mathbf{y}) = \frac{1}{P(\mathbf{x})} \sum_{j=\{f,b\}} \{P(\mathbf{x}|\mathbf{\Phi}, \mathbf{p}, M_j)P(M_j|\mathbf{y})\} P(\mathbf{\Phi})P(\mathbf{p}). \tag{6.4}$$

Note that the pixel-wise posterior and marginalisation are the subtle but crucial differences to the work in [Cremers, 2006], which lacks the marginalisation step and uses a pixel-wise likelihood $P(\mathbf{y}|M)$. We show in Section 6.8 that our formulation yields a much better behaved objective. We consider two possible methods for fusing the pixel-wise posteriors: (i) a logarithmic opinion pool (LogOP):

$$P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega}) = \prod_{i=1}^{N} \left\{ \sum_{j=\{f,b\}} \{P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, M_j)P(M_j|\mathbf{y}_i)\} \right\} P(\mathbf{\Phi})P(\mathbf{p}) \tag{6.5}$$

and (ii) a linear opinion pool (LinOP):

[2]It is also the distribution that would be computed in the E-Step of an EM solution to the problem.

$$P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega}) = \sum_{i=1}^{N} \left\{ \sum_{j=\{f,b\}} \{P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, M_j)P(M_j|\mathbf{y}_i)\} \right\} P(\mathbf{\Phi})P(\mathbf{p}). \tag{6.6}$$

The logarithmic opinion pool is normally the preferred choice and is most similar to previous work [Cremers, 2006; Cremers *et al.*, 2007]. It assumes pixel-wise independence; whereas the linear opinion pool is equivalent to marginalising over the pixel locations – this is allowed as our bag-of-pixels generative model treats pixel locations as a random variable. We continue our derivation assuming a logarithmic opinion pool for clarity, but also include results using a linear opinion pool for completeness. For more information on opinion pools refer to [Manyika & Durrant-Whyte, 1994]. Note the term $\frac{1}{P(\mathbf{x})}$ has been dropped as it is constant for all pixel locations and we only seek to maximise $P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega})$.

6.4 Segmentation

The typical approach to region-based segmentation is to take a product of the pixel-wise likelihood functions $\prod_{i=1}^{N} P(\mathbf{I}(x_i)|M_i)$, over the pixel locations $\mathbf{x}_i$, to get the overall likelihood $P(\mathbf{I}|M)$. This can then be expressed as a summation by taking logs and optimised using variational level-sets [Osher & Paragios, 2003; Cremers *et al.*, 2007]. In contrast to these methods, our derivation leads to pixel-wise posteriors and marginalisation (6.5), a subtle but important difference.

For the remainder of this section, in order to simplify our expressions (and without loss of generality), we assume that the registration is correct and therefore $\mathbf{x}_i = \mathbf{W}(\mathbf{x}_i, \mathbf{p})$. We now specify the term $P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, M)$ in (6.5) and the term $P(M)$ in (6.3) :

$$P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, M_f) = \frac{H_\epsilon(\mathbf{\Phi}(\mathbf{x}_i))}{\eta_f} \qquad P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, M_b) = \frac{1 - H_\epsilon(\mathbf{\Phi}(\mathbf{x}_i))}{\eta_b} \tag{6.7}$$

$$P(M_f) = \frac{\eta_f}{\eta} \qquad P(M_b) = \frac{\eta_b}{\eta}, \tag{6.8}$$

where

$$\eta = \eta_f + \eta_b, \qquad \eta_f = \sum_{i=1}^{N} H_\epsilon(\mathbf{\Phi}(\mathbf{x}_i)), \qquad \eta_b = \sum_{i=1}^{N} 1 - H_\epsilon(\mathbf{\Phi}(\mathbf{x}_i)). \tag{6.9}$$

Equation (6.7) represents normalised versions of the blurred Heaviside step functions used in typical region-based level-set methods and can now be interpreted probabilistically as model specific spatial priors for a pixel location $\mathbf{x}$. Equation (6.8) represents the model priors, which are given by the ratio of the area of the model specific region to the total area of both models. Equation (6.9) contains the normalisation constants (note that $\eta = N$).

We now specify a geometric prior on $\mathbf{\Phi}$ that rewards a signed distance function:

$$P(\mathbf{\Phi}) = \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp - \frac{(|\nabla \mathbf{\Phi}(\mathbf{x}_i)| - 1)^2}{2\sigma^2}, \tag{6.10}$$

where σ specifies the relative weight of the prior. This gives a probabilistic interpretation to the work in [Li *et al.*, 2005]. The term $|\nabla \mathbf{\Phi}(\mathbf{x}_i)|$ is the magnitude of the gradient at point $\mathbf{x}_i$ and for a signed distance function should equal one everywhere. Maintaining a signed distance function is desirable as it has good properties when performing the registration step, which will be described in the next section. Substituting (6.7), (6.8), (6.9) and (6.10) into (6.5) and taking logs, gives the following expression for the log posterior:

$$\log(P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega})) \propto \sum_{i=1}^{N} \left\{ \log\left(P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, \mathbf{y}_i)\right) - \frac{(|\nabla \mathbf{\Phi}(\mathbf{x}_i)| - 1)^2}{2\sigma^2} \right\} + N \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \log(P(\mathbf{p})), \tag{6.11}$$

where

$$P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, \mathbf{y}_i) = \frac{P_f H_\epsilon(\mathbf{\Phi}) + P_b(1 - H_\epsilon(\mathbf{\Phi}))}{P_f \sum H_\epsilon(\mathbf{\Phi}) + P_b \sum (1 - H_\epsilon(\mathbf{\Phi}))}$$

and

$$P_f = P(\mathbf{y}_i|M_f), \qquad P_b = P(\mathbf{y}_i|M_b), \qquad \mathbf{\Phi} = \mathbf{\Phi}(\mathbf{x}_i).$$

Given that we are about to optimise with respect to $\mathbf{\Phi}$, we can drop the last two terms in (6.11) and by calculus of variations [Evans, 2002] express the first variation (Gateaux derivative) of the functional as (see Appendix C for details):

$$\frac{\partial \log(P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega}))}{\partial \mathbf{\Phi}} = \frac{\delta_\epsilon(\mathbf{\Phi})(P_f - P_b)}{P_f H_\epsilon(\mathbf{\Phi}) + P_b(1 - H_\epsilon(\mathbf{\Phi}))} - \frac{1}{\sigma^2}\left[\nabla^2 \mathbf{\Phi} - \text{div}\left(\frac{\nabla \mathbf{\Phi}}{|\nabla \mathbf{\Phi}|}\right)\right], \tag{6.12}$$

where ∇^2 is the Laplacian operator and $\delta_\epsilon(\mathbf{\Phi})$ is the derivative of the blurred Heaviside step function, i.e. a blurred Dirac delta function. Interestingly, $\delta_\epsilon(\mathbf{\Phi})$ can now be interpreted as a way of expressing uncertainty on the contour $\mathbf{C}$. If we were to use Gaussian uncertainty for the contour, then the region-based uncertainty would be expressed in terms of $\text{erf}(\mathbf{\Phi})$ instead of $H_\epsilon(\mathbf{\Phi})$ (we do not explore this any further in this dissertation). We seek $\frac{\partial \log(P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega}))}{\partial \mathbf{\Phi}} = 0$ by carrying out steepest-ascent using the following gradient flow:

$$\frac{\partial \mathbf{\Phi}}{\partial t} = \frac{\partial \log(P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega}))}{\partial \mathbf{\Phi}}. \tag{6.13}$$

In practice this is implemented using a simple numerical scheme on a discrete grid. All spatial derivatives are computed using central differences and the Laplacian uses a 3×3 spatial kernel. We use $\sigma = \sqrt{50}$ and a time-step $\Delta t = 1$ for all experiments. For stability $\frac{\Delta t}{\sigma^2} < 0.25$ must be satisfied (see [Li *et al.*, 2005] for details).

6.5 Registration

It is possible to pose the tracking problem directly in a segmentation framework [Freedman & Zhang, 2004]. Instead, like [Cremers, 2006] we model the frame-to-frame registration explicitly, by having the level-set in the object frame and introducing a warp $\mathbf{W}(\mathbf{x}, \mathbf{p})$ into (6.11). The main benefits of this approach are: (i) control

over the interaction between registration (tracking) and segmentation (local shape deformation); (ii) by registering the embedding function first, fewer iterations are required to take account of shape changes (in fact we find one per frame is adequate for our sequences). We now drop any terms in (6.11) that are not a function of $\mathbf{p}$ in preparation for differentiation:

$$\log(P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega})) \propto \sum_{i=1}^{N} \Big\{ \log\left(P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, \mathbf{y}_i)\right) \Big\} + \log(P(\mathbf{p})) + const. \tag{6.14}$$

If we now take $P(\mathbf{p})$ to be the uninformative uniform distribution (i.e. no motion model) and compute the MAP estimate by maximising with respect to $\mathbf{p}$, then we can write:

$$\mathbf{p} = \arg\max_{\mathbf{p}} \left\{ \sum_{i=1}^{N} \log P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, \mathbf{y}_i) \right\}. \tag{6.15}$$

Using the short-hand $P(\ldots) = P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, \mathbf{y}_i)$ we can differentiate once with respect to $\mathbf{p}$ to obtain the Jacobian:

$$\mathbf{G} = \frac{\partial \log P(\ldots)}{\partial \mathbf{p}} = \frac{1}{P(\ldots)} \frac{\partial P(\ldots)}{\partial \mathbf{p}} \tag{6.16}$$

and a second time to obtain the Hessian:

$$\frac{\partial \mathbf{G}}{\partial \mathbf{p}} = \frac{-1}{P(\ldots)^2} \frac{\partial P(\ldots)}{\partial \mathbf{p}}^T \frac{\partial P(\ldots)}{\partial \mathbf{p}} + \frac{1}{P(\ldots)} \frac{\partial^2 P(\ldots)}{\partial \mathbf{p}^2}. \tag{6.17}$$

Referring back to (6.4) we can break the term $\frac{\partial P(\ldots)}{\partial \mathbf{p}}$ into two components:

$$\frac{\partial P(\ldots)}{\partial \mathbf{p}} = \mathbf{J}_n B_n, \tag{6.18}$$

where

$$\mathbf{J}_n = \frac{\partial H_\epsilon}{\partial \mathbf{\Phi}} \frac{\partial \mathbf{\Phi}}{\partial \mathbf{W}} \frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \delta_\epsilon(\mathbf{\Phi}) \nabla \mathbf{\Phi}(\mathbf{x}_n) \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \tag{6.19}$$

and

$$B_n = \frac{P_f - P_b}{P_f H_\epsilon(\mathbf{\Phi}) + P_b(1 - H_\epsilon(\mathbf{\Phi}))}. \tag{6.20}$$

The second term in (6.17) will contain second-order derivatives of the level-set embedding function $\mathbf{\Phi}(\mathbf{x}_n)$. Given that this function is approximately a signed distance function due to (6.10) i.e. zero curvature near the contour, we can approximate (6.17) by its first term only. This enables us to use an approximate[3] version of the second-order Newton optimisation scheme:

$$\Delta \mathbf{p} = \left[\sum_{i=1}^{N} B_i^2 \mathbf{J}_i^T \mathbf{J}_i \right]^{-1} \sum_{i=1}^{N} \mathbf{J}_i^T B_i. \tag{6.21}$$

Equation (6.21) is then used to update the parameters $\mathbf{p}$ by composing $\mathbf{W}(\mathbf{x}_i, \mathbf{p})$ with $\mathbf{W}(\mathbf{x}_i, \Delta\mathbf{p})^{-1}$, analogous to inverse compositional tracking [Baker & Matthews, 2004]. For this reason the warp must form a group [Baker & Matthews, 2004]; however, this is acceptable as many common useful transformations in computer vision do form groups, for instance: translation, translation+scale, similarity transforms, affine transforms and homographies.

6.6 Spatial Drift Correction

Having the object represented by its location $\mathbf{p}$ and shape $\mathbf{\Phi}$ leaves an ambiguity where it is possible to explain rigid transformations of the shape either with $\mathbf{p}$ or $\mathbf{\Phi}$. Ideally, any rigid motion would be explained solely by $\mathbf{p}$; however, over time the shape $\mathbf{\Phi}$ slowly incorporates a rigid transformation. We tackle this problem by keeping the top, bottom, left and right borders[4] (B_t, B_b, B_l, B_r) balanced and the

[3] This would be exact if we perfectly maintained a signed distance function.

[4] Smallest distances between the contour and the corresponding side of the foreground box.

minimum border distance equal to four pixels. This keeps the object contour at an appropriate size and centred within the foreground box. We use the following proportional controllers:

$$\begin{aligned} T_x &= \max(-L_t, \min(K_t(B_l - B_r), L_t)) \\ T_y &= \max(-L_t, \min(K_t(B_t - B_b), L_t)) \\ S &= \max(-L_s, \min(K_s(B_{des} - B_{min}), L_s)) \end{aligned} \tag{6.22}$$

where

$$B_{min} = \min(B_l, B_r, B_t, B_b)$$

to build a drift correction warp:

$$\mathbf{W}_{dc} = \begin{bmatrix} 1+S & 0 & T_x \\ 0 & 1+S & T_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.23}$$

This warp is applied to the level-set $\mathbf{\Phi}$ and the pose parameters $\mathbf{p}$ to compensate for spatial drift. The parameters $L_t = 0.4$ and $L_s = 0.1$ are the saturation limits, $K_t = 1$ and $K_s = 0.005$ are the gains for the translation and scale controllers respectively and $B_{des} = 4$ is the desired minimum border distance. Figure 6.5 gives an example of how the drift correction works.

6.7 Online Learning

Once registration and segmentation are completed, both the foreground and background models are adapted online. This is achieved using linear opinion pools with variable learning rates α_i, $i = \{f, b\}$:

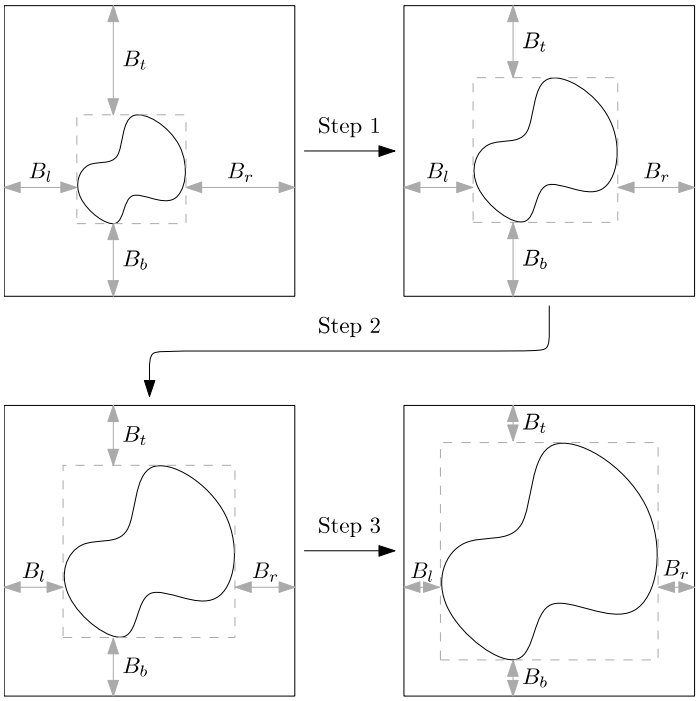

Figure 6.5: Diagram showing how the spatial drift correction works. Starting with the top-left arrangement, three warps are applied. This gradually equalises the borders and makes them equal to B_{des}.

$$P_t(\mathbf{y}|M_i) = (1-\alpha_i)P_{t-1}(\mathbf{y}|M_i) + \alpha_i P_t(\mathbf{y}|M_i), \qquad i = \{f, b\}. \tag{6.24}$$

In all experiments $\alpha_f = 0.02$ and $\alpha_b = 0.025$. For shape adaptation we control the evolution rate of the level-set using the time-step Δt.

6.8 Results

We have tested our system extensively on live video and on a variety of recorded sequences, which include objects that exhibit rapid and agile motion with significant motion blur, varying lighting, moving cameras, and cluttered and changing backgrounds. Figure 6.6 shows a qualitative evaluation of our method on three sequences. The first is a speedboat undergoing a 180° out-of-plane rotation – note how the shape is adapted online. The second is a person jumping around – note the motion blur and shape adaptation. Finally, the third is a hand being tracked from a head mounted camera past a challenging background that has a similar appearance to the object.

Figures 6.10-6.14 show a selection of different types of objects that we have successfully tracked.

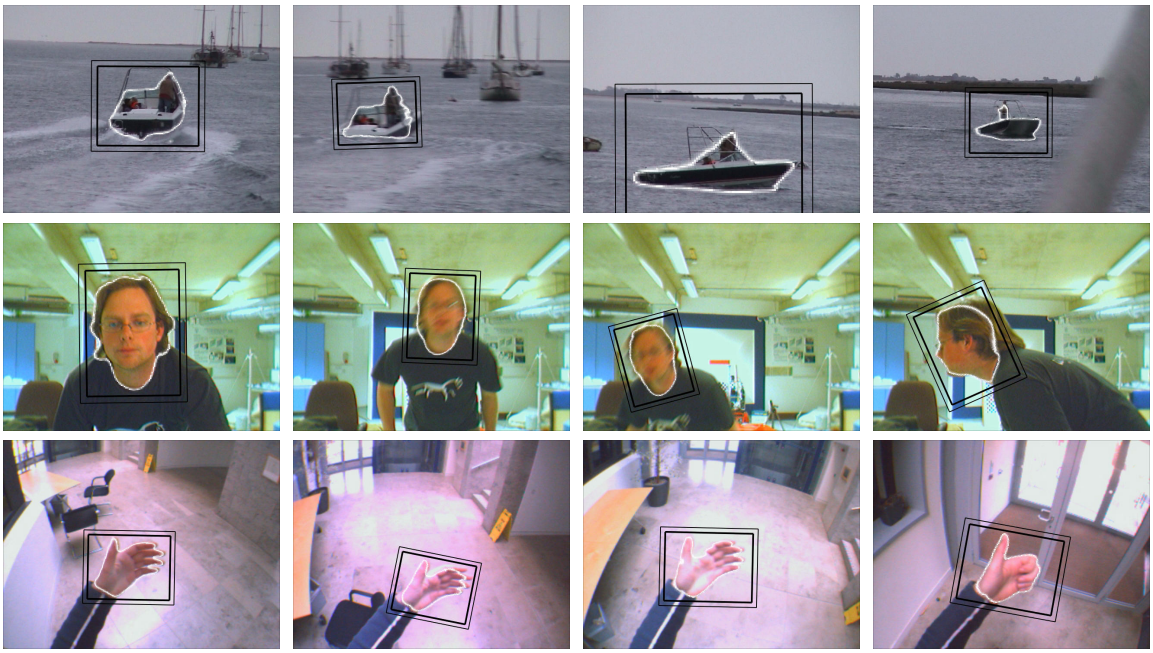

Figure 6.6: Qualitative evaluation: (top) a speedboat undergoing a 180° out-of-plane rotation illustrating shape adaptation; (middle) a person jumping around with significant motion blur and (bottom) a hand being tracked in front of a challenging background.

To perform a quantitative evaluation we have analysed the characteristics of the underlying cost function for our technique and compared this with competing alternatives on a set of pre-recorded video sequences. Figure 6.7 shows still images taken mid-sequence from a subset of these sequences; the minimum length is 400 frames and the total number of frames is over 20,000. To facilitate visualisation of the results we use a 2D rigid transformation + scale, considering each of the four dimensions separately. The competing cost functions considered correspond to the following alternative methods of tracking: level-set methods based on likelihoods [Cremers, 2006; Paragios & Deriche, 2000], mean-shift [Comaniciu *et al.*, 2000; Collins, 2003; Yilmaz, 2007], inverse compositional [Baker & Matthews, 2004] and distribution based tracking [Freedman & Zhang, 2004; Zhang & Freedman, 2005].

A good cost function has a single extremum at the true location. A poor one has multiple extrema and any local optimisation technique is liable to fall into one of these, which in practice is often the start of tracking failure. For each video frame and each dimension (translation in x and y, rotation and scale) we compute the objectives for

Figure 6.7: A selection of video frames from the data sets: (1st row) lifeboat, Coca-Cola mug, a face and a hand filmed from a head mounted camera and (2nd row) a hand using a mouse, a speedboat, a person from the caviar data set [Fisher, 2004] and a tractor. The white contour indicates the current segmentation and the two black boxes indicate the object's coordinate frame.

the competing cost functions at 40 evenly spaced points over an interval centred at the true state. We then extract all local extrema from these objectives and examine how they are distributed across the interval. To summarise this information we compute a distribution for each dimension and each cost function, using our collection of over 20,000 frames. Figure 6.8 shows a diagram of how the distribution of extrema is generated. The ideal distribution would be a delta function centred on the true state i.e. no chance of a local extremum away from the true optimum; whereas a good distribution would be peaky around the true state and have low probability of local extrema within the region it will be required to converge from. A bad distribution would be relatively flat with high probability of local extrema over the entire space, such as the one illustrated in Figure 6.8. The particular cost functions we consider are:

- **LogPWP:** Pixel-wise posteriors fused using a logarithmic opinion pool.
- **LinPWP:** Pixel-wise posteriors fused using a linear opinion pool.
- **LogLike:** Log likelihood, used in most level-set work [Cremers *et al.*, 2007; Cremers, 2006; Paragios & Deriche, 2000].
- **BhattF:** Bhattacharyya coefficient:
  $B(\boldsymbol{\Omega}_f) = \sum_{j=1}^{V} \sqrt{P(\mathbf{y}_j|M_f)P(\mathbf{y}_j|\boldsymbol{\Omega}_f)}$, used by [Comaniciu *et al.*, 2000; Collins, 2003; Yilmaz, 2007].

- **BhattFB:** Bhattacharyya coefficient with a background model: $B(\mathbf{\Omega}_f, \mathbf{\Omega}_b) = \sum_{j=1}^{V} \sqrt{P(\mathbf{y}_j|M_f)p(\mathbf{y}_j|\mathbf{\Omega}_f)} + \sum_{j=1}^{V} \sqrt{P(\mathbf{y}_j|M_b)P(\mathbf{y}_j|\mathbf{\Omega}_b)}$.

- **BhattFBM:** Bhattacharyya coefficient with a background mismatch: $B(\mathbf{\Omega}_f, \mathbf{\Omega}_b) = \sum_{j=1}^{V} \sqrt{P(\mathbf{y}|M_f)p(\mathbf{y};\mathbf{\Omega}_f)} - \sum_{j=1}^{V} \sqrt{P(\mathbf{y}_j|M_f)P(\mathbf{y}_j|\mathbf{\Omega}_b)}$, suggested by [Zhang & Freedman, 2005].

- **Ideal SSD:** Sum of squared pixel differences using the ideal template i.e. the template extracted at the current location $\mathbf{p}$. This is essentially what you would get if you had the perfect generative model giving the true pixel value at each pixel location, including the noise. This of course is *never* going to be achievable, but has been included as a useful benchmark and gives an indication of what effect incorporating texture may have.

Note:- V is the number of pixel values i.e. $32 \times 32 \times 32$; $P(\mathbf{y}|\mathbf{\Omega}_i) \quad i = \{f, b\}$ is the empirical density built from the pixels $\mathbf{\Omega}_i$ and when computing Bhattacharyya coefficients we weight the contribution of each pixel according to our shape kernel, which is identical to Yilmaz's work [Yilmaz, 2007].

Figure 6.9 shows distributions generated from over 20,000 real video frames for: translation in x, translation in y, scale and rotation.

- **Translation in x and y:** Our method has narrower distributions near the true state than all methods apart from ideal SSD and is significantly better than the log likelihood used by [Cremers, 2006]. Unlike the other methods, it also exhibits virtually no extrema outside a ± 5 pixel region – this means that our method will converge to within ± 5 pixels of the true state from anywhere within the ± 20 pixel space we have evaluated.

- **Scale:** The Bhattacharyya method and Bhattacharyya with background mismatch both have poor localisation in scale, which is in agreement with the findings of many authors. The log likelihood also poorly localises scale compared with our pixel-wise posterior based methods.

- **Rotation:** All Bhattacharyya methods and the log likelihood are poor at correctly localising the rotation. The straight Bhattacharyya coefficient for example has more than a 1% chance of exhibiting extrema anywhere in the rotation space, at a 30Hz frame rate this corresponds to approximately 1 frame in every 3 seconds of video. It is worth noting that the side lobes (at approximately

25°) exhibited by our methods and ideal SSD, are due to the self similarity corresponding to fingers in the hand sequences.

Experimentally we were unable to make the log likelihood successfully track several of our sequences, which is confirmed by its poor performance in Figure 6.9. One possible explanation is that in other work [Cremers, 2006; Cremers *et al.*, 2007; Paragios & Deriche, 2000], a single Gaussian parametric model is used. This implicitly enforces a smooth, unimodal distribution for the joint likelihood. Non-parametric representations do not exhibit these properties; however, they are better at describing complicated distributions and therefore desirable. The reason that our method can deal with these distributions is because of the normalising denominator in (6.3) and the marginalisation step in (6.4). These two steps prevent individual pixels from dominating the cost function, hence making it smoother and more well-behaved.

The work of [Freedman & Zhang, 2004] and its subsequent improvement [Zhang & Freedman, 2005] use distribution matching techniques to incorporate non-parametric distributions into a level-set framework. These methods, similar to the Bhattacharyya based methods, involve computing the empirical densities at every iteration of the optimisation, whereas our method avoids this extra cost. Not only is our method superior to these approaches in terms of cost functions (see Figure 6.9), but it is computationally cheaper to evaluate as it does not require empirical distributions. This is a significant benefit because it not only reduces the cost per iteration, but avoids the issue of having to build 'good' distributions. One explanation for the difference between the performance of these methods and ours, is that it is hard to build 'good' empirical distributions in real-time and most methods rely on simple histograms. Although this could be improved with Parzen or NP windowing techniques [Kadir & Brady, 2005], it would almost certainly sacrifice real-time performance.

Timing

All terms in (6.21) include $\delta_\epsilon(\mathbf{\Phi}(\mathbf{x}_i))$ (blurred Dirac delta function). This means that an individual pixel's contribution to the optimisation diminishes the further from the contour it is. An efficient implementation, therefore, recognises this. Our implementation ignores pixels outside a narrow band and for an object size of 180×180 runs in $500\mu s$ on a P4 3.6GHz machine. On average the system runs at a frame rate of 85Hz for the complete algorithm and if shape and appearance learning are turned off (i.e. rigid registration only) it averages 230Hz.

6.9 Conclusions

We have proposed a novel probabilistic framework for robust, real-time, visual tracking of previously unseen objects from a moving camera. The key contribution of our method and reason for its superior performance compared with others, is the use of pixel-wise posteriors as opposed to a product over pixel-wise likelihoods. In contrast to other methods [Cremers, 2006; Cremers *et al.*, 2007], we solve the registration using Gauss Newton, which has significant practical benefits, namely: (i) the difficulty associated with step size selection is removed and (ii) reliable and fast convergence. We have demonstrated the benefits of our method both qualitatively and quantitatively, with a thorough analysis of pixel-wise posteriors versus competing alternatives using over 20,000 video frames. Our results demonstrate that using pixel-wise posteriors provides excellent performance when incorporating non-parametric distributions into region based level-sets. It not only offers superior cost functions, but avoids the need for computing empirical distributions [Comaniciu *et al.*, 2000; Freedman & Zhang, 2004; Zhang & Freedman, 2005; Yilmaz, 2007] and is therefore faster.

One of the failure modes of this tracker is if an object with similar appearance occludes/interacts with the object being tracked. The result is the tracker will often be seduced away from the true object and either fail completely or end up tracking the wrong object. The next chapter describes how a more complicated generative model can be used to deal with multiple interacting objects that occlude each other.

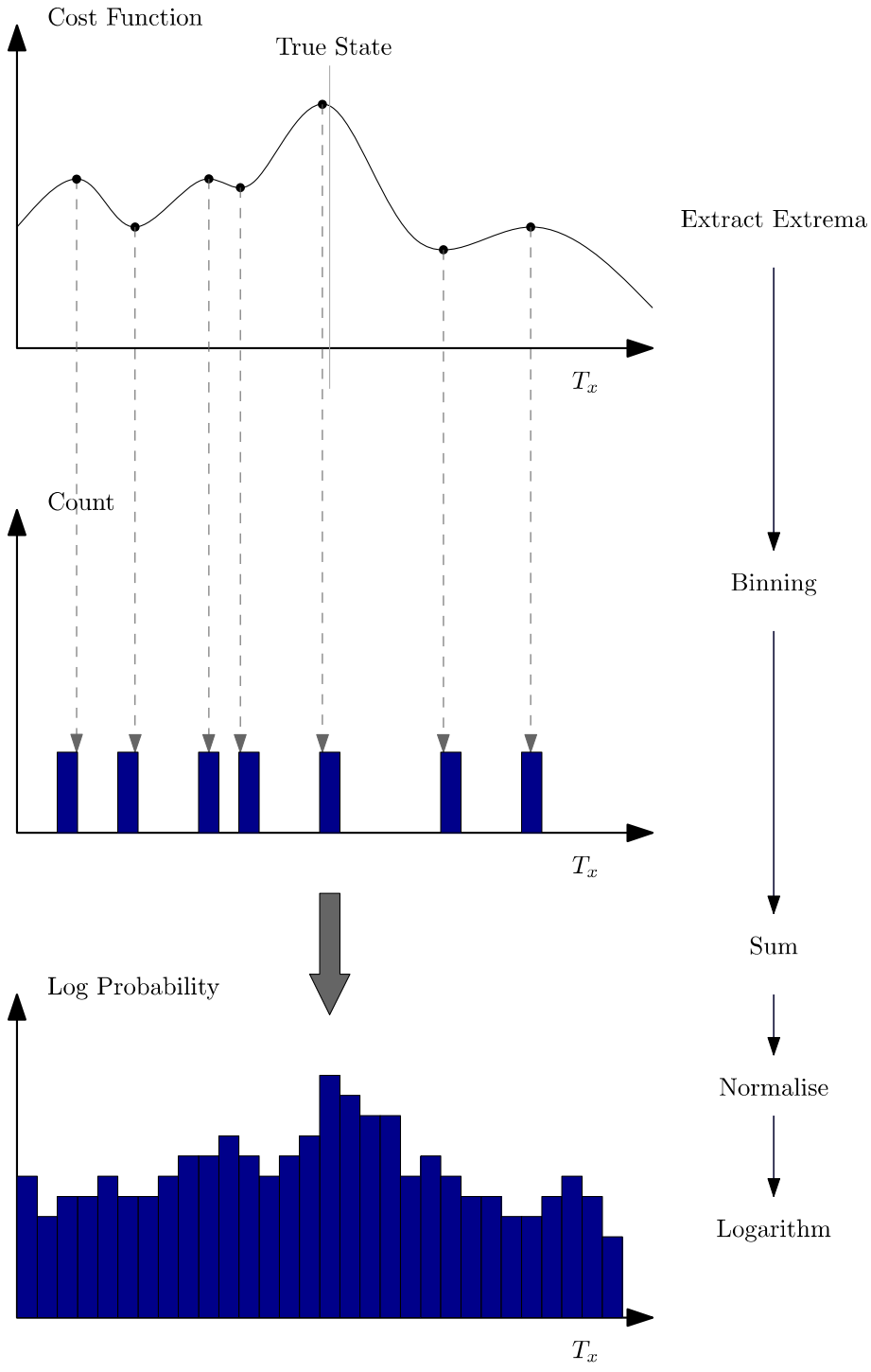

Figure 6.8: A diagram showing how the distribution of extrema is generated. For each video frame all extrema are extracted from the cost function and binned into 40 evenly spaced bins over an interval centred at the true state. These are then accumulated over all video frames into a normalised distribution. Finally, we take the log of this distribution to make visualisation easier.

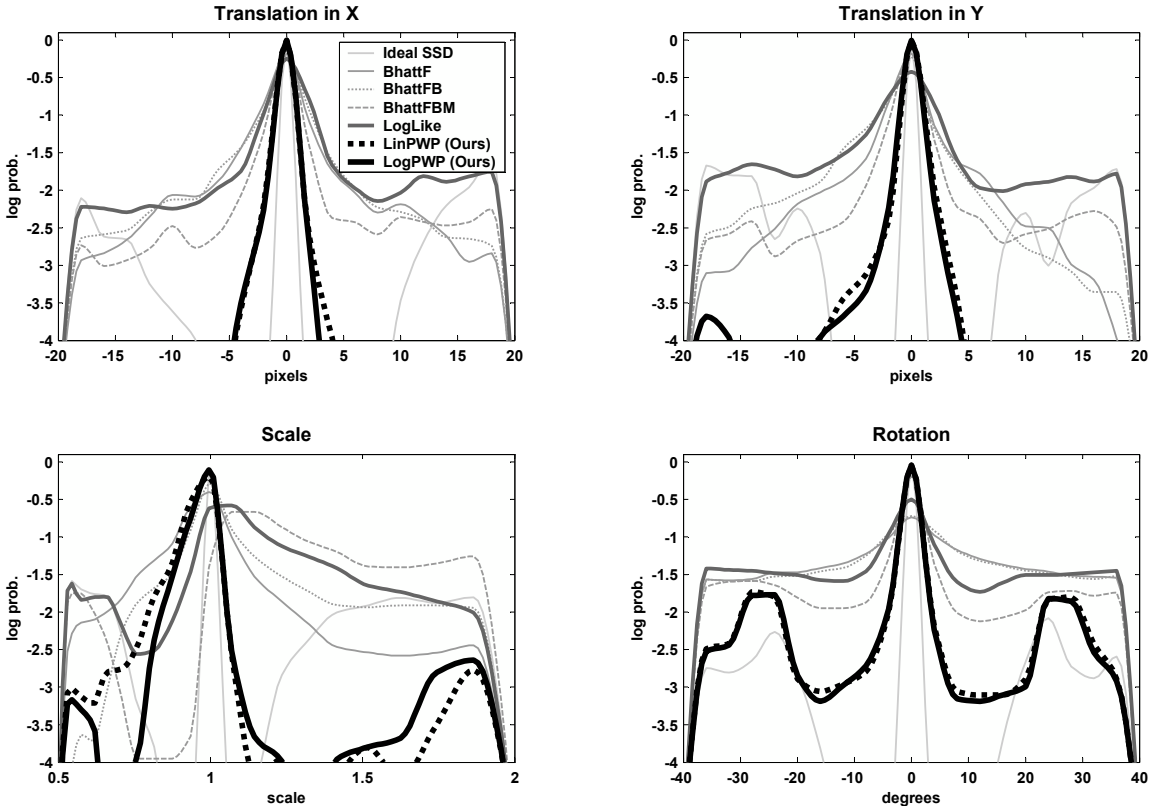

Figure 6.9: Quantitative Analysis: log probability distribution of extrema in the cost functions generated from 20,000 frames of real video data.

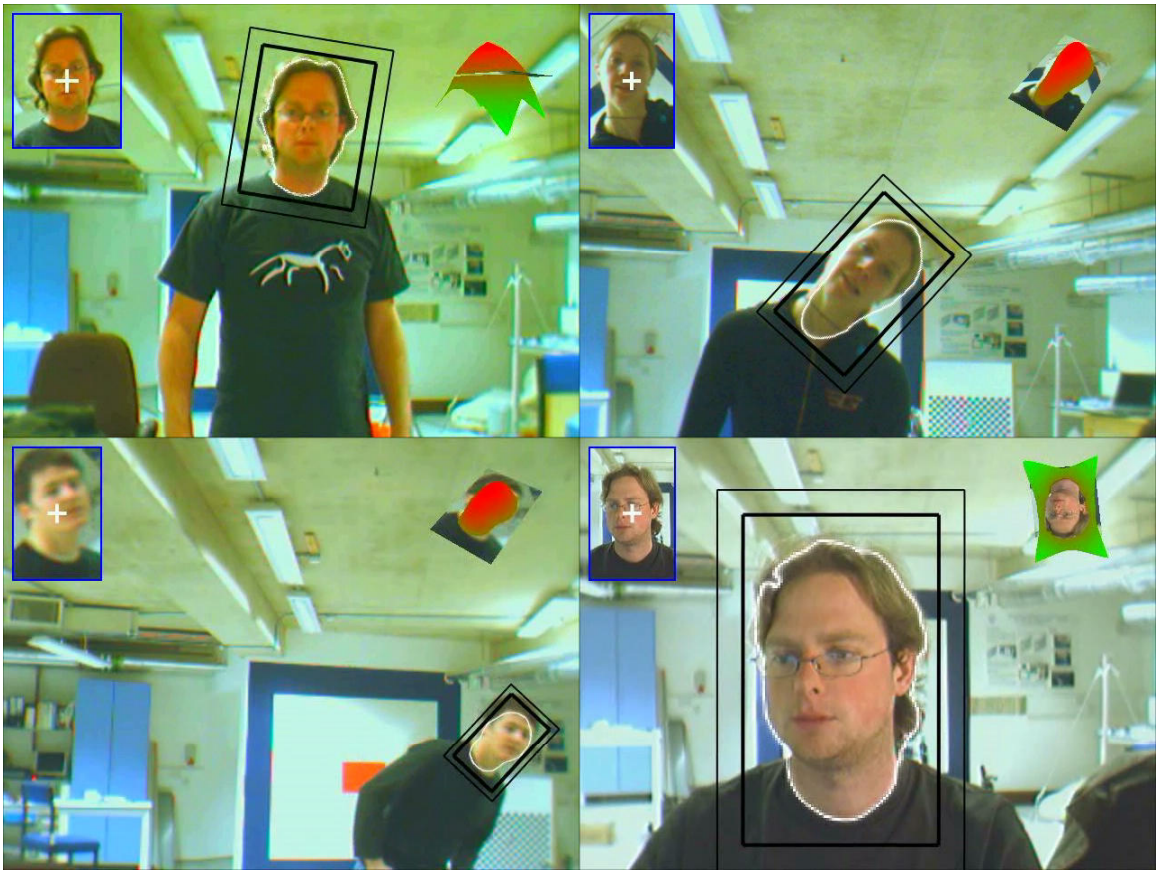

Figure 6.10: Tracking faces montage.

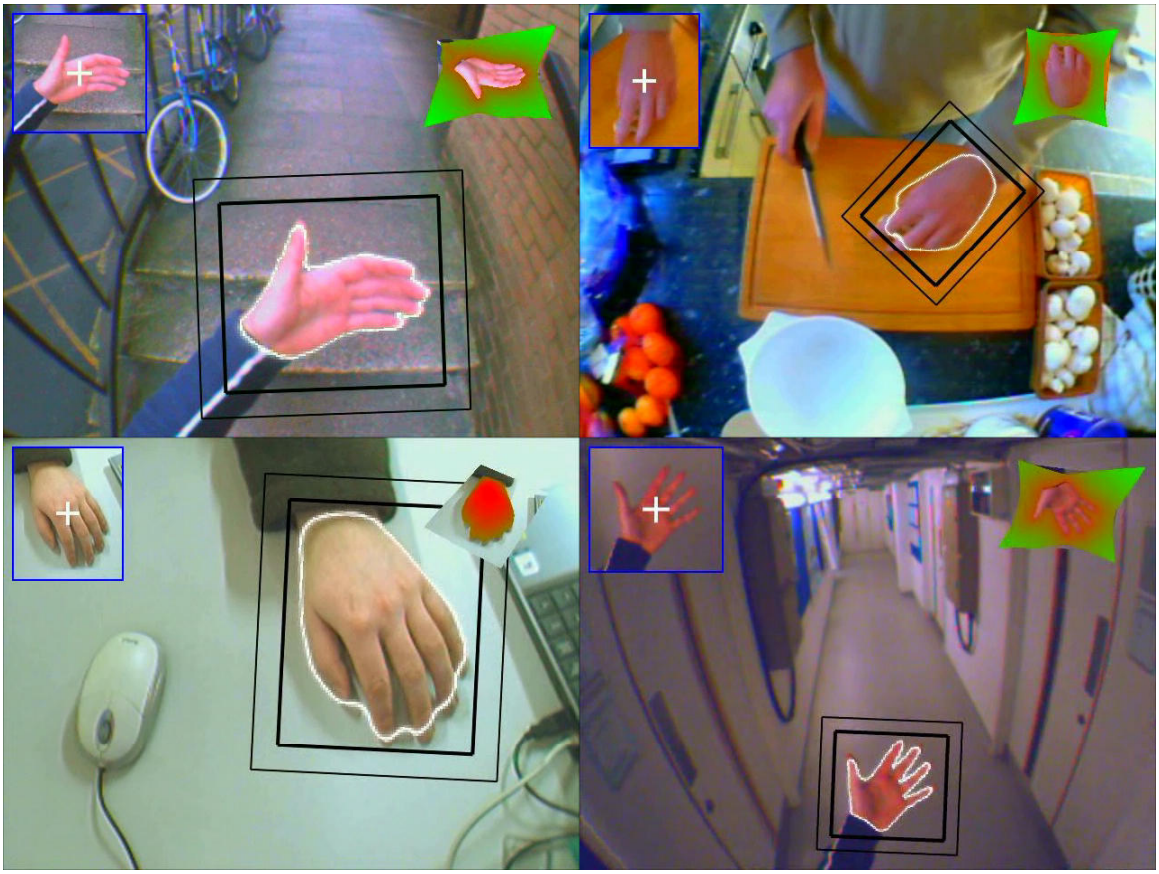

Figure 6.11: Tracking hands montage.

Figure 6.12: Tracking people montage.

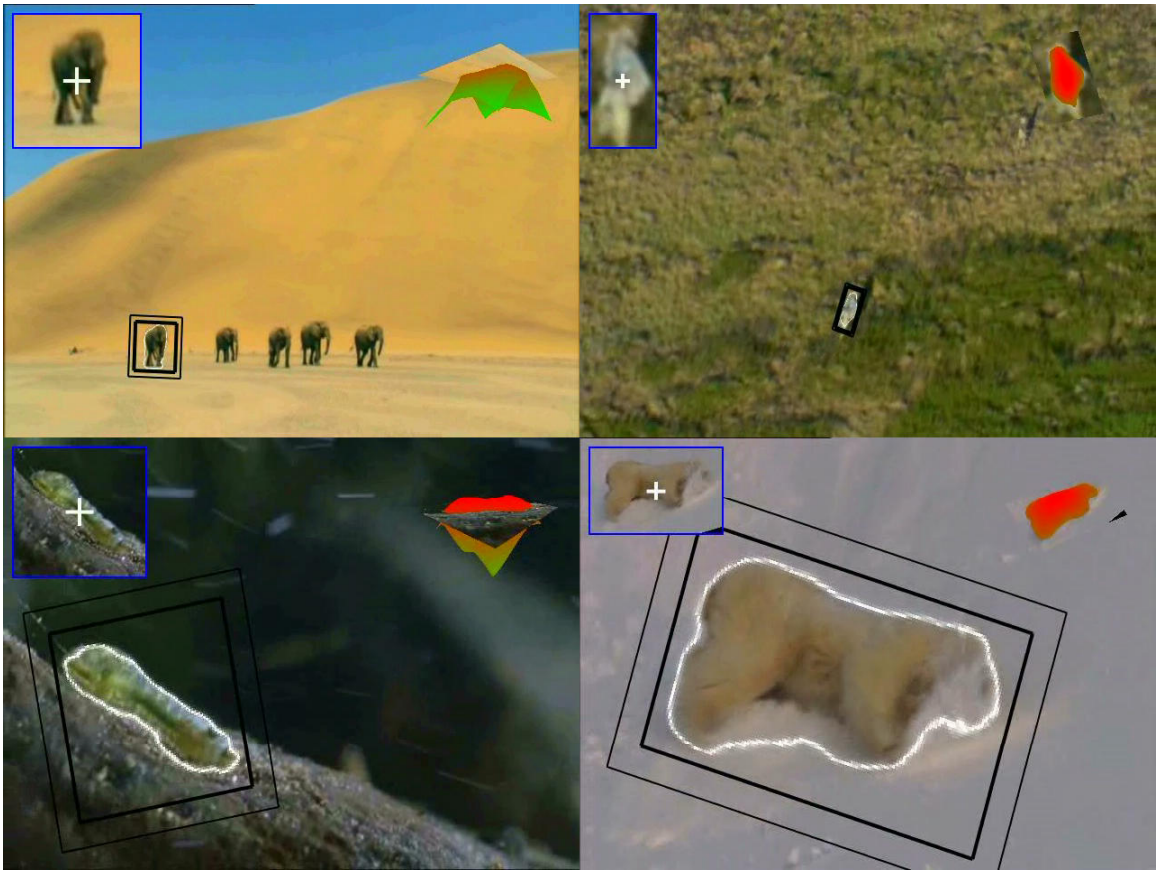

Figure 6.13: Tracking animals montage.

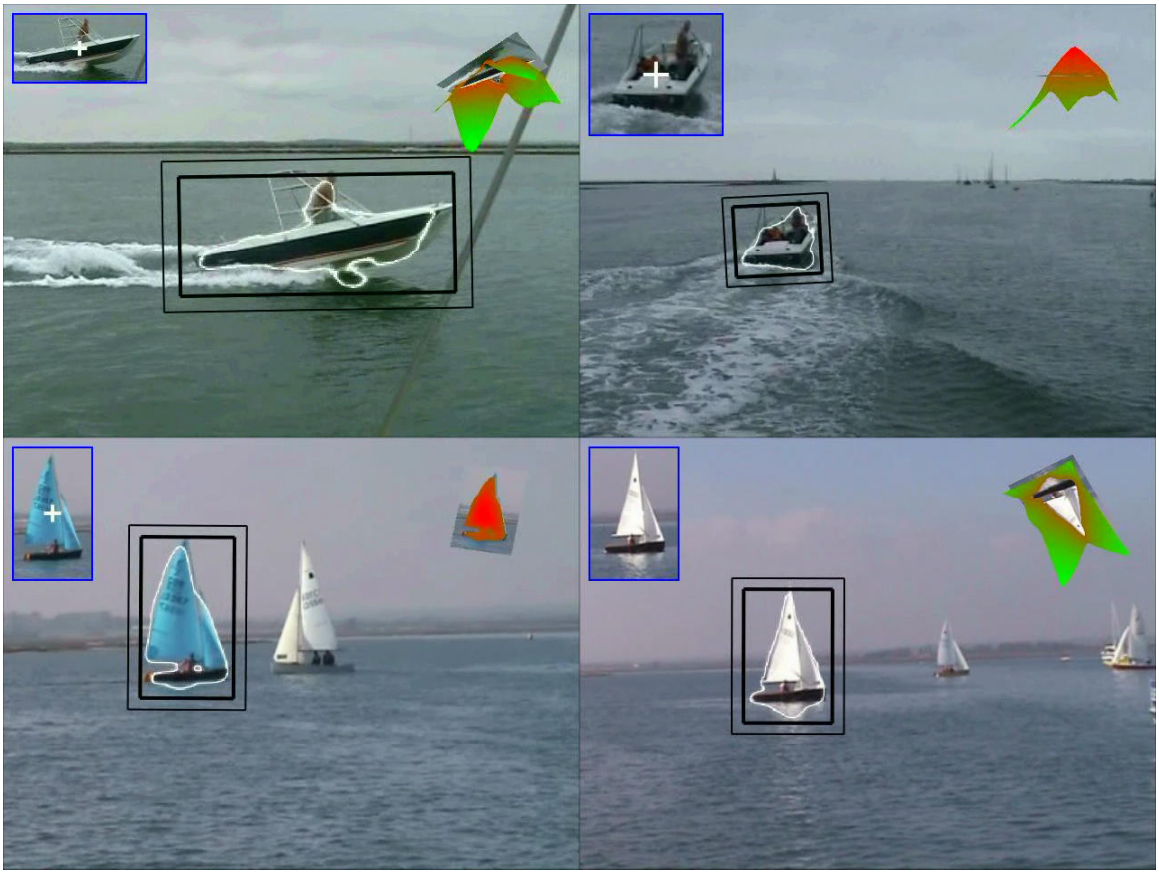

Figure 6.14: Tracking boats montage.

Chapter 7

Tracking Multiple Interacting/Occluding Objects using Pixel-Wise Posteriors

7.1 Introduction

While the previous chapter dealt with tracking a single object, this chapter will introduce a generalised model that allows us to track *multiple objects* in real-time. One approach that has demonstrated significant success in representing the presence of multiple objects, in an image or video, is that of a layered or 2.5D representation [Nitzberg & Mumford, 1990; Wang & Adelson, 1993; Jepson *et al.*, 2002; Tao *et al.*, 2000; Jojic & Frey, 2001; Reid & Connor, 2005]. The image formation process is modelled by multiple layers at different depths, with the background layer being furthest away and then a number of different object layers organised according to their relative depths in the scene. This representation is able to directly model the occlusion process, assuming that objects do not interlock with each other.

This chapter presents a fully probabilistic, generative model for the image formation process, which captures the essence of a layered representation and adds the power, speed and resilience of pixel-wise posteriors. Following the previous chapter, we use implicit contours (level-sets) to represent the boundaries of the objects being tracked [Osher & Paragios, 2003; Paragios & Deriche, 2000; Goldenberg *et al.*, 2001; Chan & Vese, 2001; Cremers, 2006] and pixel-wise posteriors to give better behaved objective functions and hence resilience to noise. The result is, to our knowledge, the first

system that can simultaneously estimate the position, scale, rotation (and potentially other parameters), depth-ordering, figure-figure and figure-ground segmentation for up to twelve occluding/interacting objects, in *real-time*, using standard PC hardware.

It is not obvious from the previous chapter how multiple occluding objects should be handled and so the key contribution of this chapter is: solving the non-trivial task of tracking multiple occluding objects using pixel-wise posteriors. In particular: (i) we begin with a more sophisticated generative model, which directly models the process of inter-object occlusion; (ii) we take the Bayesian approach and marginalise out all nuisance parameters when inferring the best configuration over the multi-object space; (iii) we show how to introduce motion models; (iv) we compute a posterior over the depth-ordering of multiple objects and (v) we demonstrate the ability to track through complete occlusions in challenging situations.

Given our generative model, we are able to perform inference and an optimisation at each frame to find the MAP estimate for the configuration of the multiple objects. An alternative to this approach, which has been used successfully for multi-object tracking, is the use of a non-parametric representation, in particular, the various methods based on particle filtering [Isard & MacCormick, 2001; Vermaak & Doucet, 2003; Khan *et al.*, 2004]. A significant advantage of these methods is that you only need to sample a particle distribution and weight each particle by the likelihood function, which in practice is often easier to implement than a direct optimisation. The downsides are that often a very large number of particles is required to achieve a good approximation of the true posterior and that given this approximation it is not always obvious how to interpret the particle set.

An important aspect to any multi-object tracking system is probabilistic exclusion, which was first demonstrated in a particle filtering context by [MacCormick & Blake, 1999]. Probabilistic exclusion enforces that a single measurement should not be allowed to explain multiple objects, which in practice ensures that several objects do not incorrectly get assigned to the same data. The generative model we propose enforces this directly by only allowing a pixel to be generated from a single object.

Our method only needs an initial bounding box to start tracking an object and therefore has to estimate everything about the objects and their interactions from the incoming video data. In contrast, if 3D models for the objects are available before tracking commences, then it is possible to estimate the full 3D poses of the objects and hence compute occlusion/visibility directly from their 3D structure [Drummond & Cipolla, 2000; Schmaltz *et al.*, 2007]. This can offer superior performance and

accuracy; however, there are many real-world scenarios where this information is not available before tracking commences and so a method such as ours is required that can be initialised without prior knowledge of 3D structure.

The remainder of this chapter is organised as follows: Section 7.2 describes the representation of the objects being tracked; Section 7.3 describes our generative models and their corresponding probability distributions; Section 7.4 outlines our method for tracking; Section 7.5 shows our results and Section 7.6 concludes with a summary and discussion.

7.2 The Representation

Figure 7.1 illustrates how we represent each of the K objects being tracked by their shape $\mathbf{C}_j$, their location in the image $\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)$ and two underlying appearance models: one for the foreground $P(\mathbf{y}|M = m_j^f)$ and one for the background $P(\mathbf{y}|M = m_j^b)$.

Shape: This is represented by the zero level-sets $\mathbf{C}_j = \{\mathbf{x}|\mathbf{\Phi}_j(\mathbf{x}) = 0\}$ of the embedding functions $\mathbf{\Phi}_j(\mathbf{x})$ [Osher & Paragios, 2003; Cremers, 2006]. This is identical to the approach in the previous chapter except there is now a level-set for each object being tracked.

Location: This is described by a warp $\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)$ that takes a pixel location $\mathbf{x}_n$ in object j's coordinate frame and warps it into the image frame according to parameters $\mathbf{p}_j$. Again this is identical to the approach in the previous chapter except there is now a set of pose parameters for each object being tracked.

Appearance models: Each object j has a pair of colour models $P(\mathbf{y}|M = m_j^f)$ and $P(\mathbf{y}|M = m_j^b)$, one for its foreground pixels and one for the nearby background pixels. We use a local background model for each object being tracked, as opposed to a single model shared between objects. We have tried both approaches and we have found that a local background model per object is crucial to the success of the algorithm. This is because the background can vary significantly over the image and successful tracking requires that each object is segmented from the local background i.e. nearby background pixels.

Regions: This is the most significant difference from the previous chapter. The area in each object's coordinate frame is broken into multiple regions R. For a single object there would simply be two regions: foreground and background. Extending

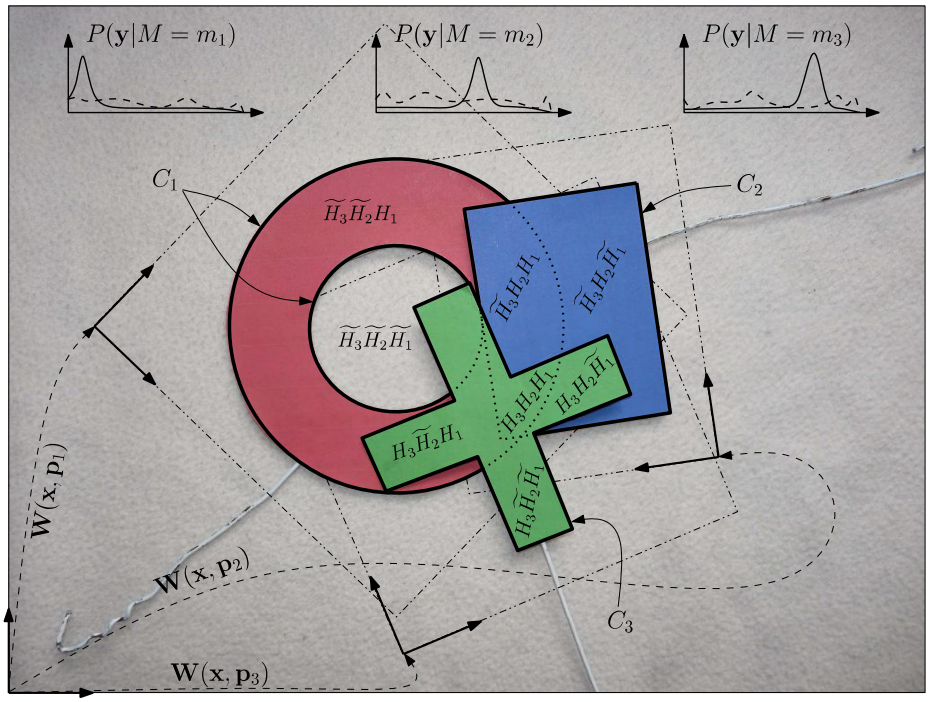

Figure 7.1: Representation for three objects showing: the three pairs of colour models $P(\mathbf{y}|M = m_j)$ where $j = \{1, 2, 3\}$; the three warps into the objects' coordinate frames $\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)$; the three contours $\mathbf{C}_j$ corresponding to the shapes of the three objects and the eight potential regions: $\widetilde{H_3}\widetilde{H_2}\widetilde{H_1}$, $\widetilde{H_3}\widetilde{H_2}H_1$, $\widetilde{H_3}H_2\widetilde{H_1}$, $\widetilde{H_3}H_2H_1$, $H_3\widetilde{H_2}\widetilde{H_1}$, $H_3\widetilde{H_2}H_1$, $H_3H_2\widetilde{H_1}$ and $H_3H_2H_1$, which correspond to different types of overlap for example $\widetilde{H_3}\widetilde{H_2}\widetilde{H_1}$ is the background region and $H_3\widetilde{H_2}H_1$ is the region where object 1 and object 3 overlap in the image.

to two objects there are four regions: background, object 1, object 2 and the overlap between object 1 and 2. In general, for K objects there are 2^K regions required to cover all possible types of interaction (occlusion). Figure 7.1 shows an example of three interacting objects with their eight potential regions. The notation H_j and $\widetilde{H_j}$ can be interpreted as: foreground of object j and not foreground of object j respectively, for example $H_3H_2H_1$ is the foreground of objects 1,2 and 3, in other words the region where these three objects overlap. In summary, we use the following notation:

- N: The number of pixels.
- K: The number of objects.
- $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$: The set of pixel locations in the object's coordinate frame.

- $\mathbf{y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$: The set of pixel values (in our experiments this is a RGB value).
- $\mathbf{I} = \{\{\mathbf{x}_1, \mathbf{y}_1\}, \ldots, \{\mathbf{x}_N, \mathbf{y}_N\}\}$: Image within the objects' coordinate frames.
- $j = \{1, \ldots, K\}$: Object index where K is the number of objects being tracked.
- $\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)$: Warp with parameters $\mathbf{p}_j$ corresponding to object j.
- $M_n = \{m_1^{\{b,f\}}, \ldots, m_K^{\{b,f\}}\}$: Model parameter either background or foreground for one of the K objects, there are $2K$ models.
- D: The discrete depth-ordering of the foreground objects, there are $K!$ possible orderings.
- R_n: The region that the pixel $\mathbf{x}_n$ has been generated from. With K objects there are 2^K possible regions.
- $\mathbf{C}_j$: The contour that segments the foreground object j from the background.
- $\mathbf{\Phi}, \mathbf{p} = \{\{\mathbf{\Phi}_1, \mathbf{p}_1\}, \ldots, \{\mathbf{\Phi}_K, \mathbf{p}_K\}\}$: Shape kernels (level-set embedding functions) and pose parameters.
- $H_\epsilon(z)$: Smoothed Heaviside step function, where ϵ is the smoothing parameter.
- $\delta_\epsilon(z)$: Smoothed Dirac delta function, where ϵ is the smoothing parameter.
- $H_j = H_\epsilon(\mathbf{\Phi}_j(\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)))$: Shorthand for the smoothed Heaviside step function applied to object j's shape kernel.
- $\widetilde{H_j} = 1 - H_\epsilon(\mathbf{\Phi}_j(\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)))$: Shorthand for one minus the smoothed Heaviside step function applied to object j's shape kernel.

7.3 The Generative Models

We will now consider two different approaches for modeling the relative depth-ordering of the objects. The first assumes that *each pixel* belonging to an object carries its own relative depth. This is a very general model and allows objects to interact in complicated ways, for example meshing hands. The second approach is more restrictive and assumes that *each frame* carries a relative depth. This is more in keeping

with the traditional layered representations and means that the pixels belonging to an object all share a common relative depth.

Figure 7.2 shows generative models that correspond to the two approaches and have the following variables: $\mathbf{\Phi}$ is the shape kernels, one per object; $\mathbf{p}$ is the set of parameters describing the rigid transformations from each object's coordinate frame to the image frame; D is the discrete depth-ordering of the objects (there are $K!$ orderings); M_n is the appearance model (there are $2K$ appearance models); R_n is the region the pixel is generated from (there are 2^K regions); $\mathbf{y}_n$ is the pixel colour and $\mathbf{x}_n$ is the pixel location. The difference between the two models is that the one on the left includes D_n within the plate, meaning that depth is treated as a local (pixel-wise) parameter, whereas the other model treats D as a global (frame-wise) parameter.

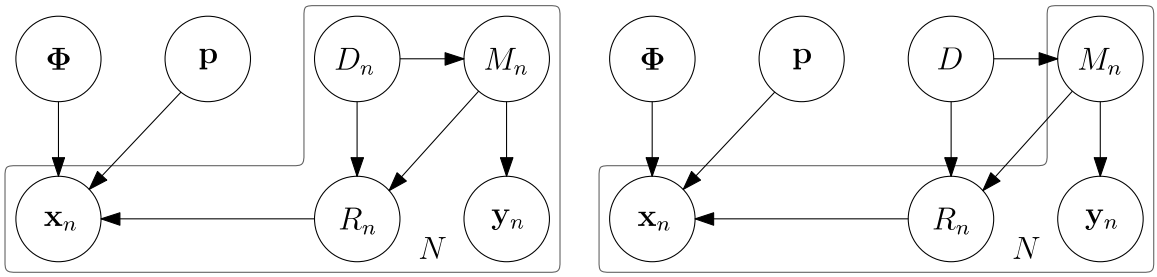

Figure 7.2: Generative models representing the image as a bag-of-pixels: (left) the discrete depth-ordering is a local parameter and (right) the discrete depth-ordering is a global parameter (note:- the gray polygon denotes a plate over the N pixels in the objects' coordinate frames).

The intuition behind these graphical models is that given the shape $\mathbf{\Phi}$, the pose $\mathbf{p}$ and the depth-ordering D, you can first sample a particular appearance model M_n, then sample a region R_n where the appearance model is present and finally sample a $\{\mathbf{x}_n, \mathbf{y}_n\}$ pixel pair, which tells you where the pixel is $\mathbf{x}_n$ and what colour it has $\mathbf{y}_n$. If you were to sample from this generative model N times, then you would end up with an image that is representative of the original apart from the fact that pixels within an object's contour are jumbled up, which is because we are modeling the colour distributions $P(\mathbf{y}_n|M_n)$ rather than the specific spatial arrangement of pixels. The joint distribution for a single pixel given the left-hand generative model is:

$$P(\mathbf{x}_n, \mathbf{y}_n, \mathbf{\Phi}, \mathbf{p}, R_n, M_n, D_n) = P(\mathbf{\Phi})P(\mathbf{p})\times$$
$$P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)P(R_n|D_n, M_n)\times$$
$$P(\mathbf{y}_n|M_n)P(M_n|D_n)P(D_n). \quad (7.1)$$

By performing inference we can obtain the MAP estimate of $P(\mathbf{\Phi}, \mathbf{p}, D_n|\mathbf{I})$ and compute the best shape, pose and discrete depth-ordering to explain the observed image data. The way that we achieve this is to first condition on $\mathbf{x}_n$ and $\mathbf{y}_n$, where $P(\mathbf{x}_n)$ is constant and

$$P(\mathbf{y}_n) = \sum_{M_n, D_n} P(\mathbf{y}_n|M_n)P(M_n|D_n)P(D_n). \quad (7.2)$$

Then second, marginalise over R_n and M_n to give us the pixel-wise posterior:

$$P(\mathbf{\Phi}, \mathbf{p}, D_n|\mathbf{x}_n, \mathbf{y}_n) = \frac{1}{P(\mathbf{x}_n)P(\mathbf{y}_n)}P(\mathbf{\Phi})P(\mathbf{p})\times$$
$$\sum_{R_n, M_n} P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)P(R_n|D_n, M_n)\times$$
$$P(\mathbf{y}_n|M_n)P(M_n|D_n)P(D_n). \quad (7.3)$$

Finally, we take the product over the pixel sites to get the posterior given the image $\mathbf{I}$:

$$P(\mathbf{\Phi}, \mathbf{p}, D|\mathbf{I}) = P(\mathbf{\Phi})P(\mathbf{p})\prod_{n=1}^{N}\frac{1}{P(\mathbf{y}_n)}\times$$
$$\sum_{R_n, M_n} P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)P(R_n|D_n, M_n)\times$$
$$P(\mathbf{y}_n|M_n)P(M_n|D_n)P(D_n). \quad (7.4)$$

The corresponding posterior taking D as a global parameter is:

$$P(\mathbf{\Phi}, \mathbf{p}, D|\mathbf{I}) = P(\mathbf{\Phi})P(\mathbf{p})P(D)\prod_{n=1}^{N}\frac{1}{P(\mathbf{y}_n)}\times \sum_{R_n, M_n} P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)P(R_n|D, M_n)\times P(\mathbf{y}_n|M_n)P(M_n|D), \quad (7.5)$$

where there is now a single D for all pixels in the current frame (see the right-hand generative model in Figure 7.2).

The Probability Distributions

We will now explain each of the distribution terms which comprise (7.4) and (7.5) in detail:

- $P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)$: is the probability of the pixel location $\mathbf{x}_n$ given the shape $\mathbf{\Phi}$, the pose $\mathbf{p}$ and the region R_n, which can be written in terms of the blurred Heaviside step function H_j and one minus the blurred Heaviside step function $\widetilde{H_j}$. For K objects there are 2^K regions, so for example with 3 objects there are 8 regions, which take the form:

$$\begin{aligned} P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n = 0) &= \widetilde{H_3}\widetilde{H_2}\widetilde{H_1}/\eta_0 \\ P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n = 1) &= \widetilde{H_3}\widetilde{H_2}H_1/\eta_1 \\ P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n = 2) &= \widetilde{H_3}H_2\widetilde{H_1}/\eta_2 \\ &\cdots \\ P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n = 7) &= H_3H_2H_1/\eta_7, \end{aligned} \quad (7.6)$$

where the terms on the right-hand side follow a binary sequence (this is related to the work on multi-phase level-sets [Vese & Chan, 2002]) and are normalised by η_r so that the distributions sum to one.

- $P(R_n|D_n, M_n)$: is the probability of a region R_n given a discrete depth-ordering D_n and a model M_n. This is computed using a ratio of region areas, an example using the object arrangement in Figure 7.1 would be:

$$P(R_n = 7|D_n, M_n = m_3^f) = \eta_7/(\eta_4 + \eta_5 + \eta_6 + \eta_7). \tag{7.7}$$

- $P(\mathbf{y}_n|M_n)$: is the probability of the colour $\mathbf{y}_n$ given the model M_n (represented using a colour histogram).

- $P(M_n|D_n)$: is the probability of a model M_n given a discrete depth-ordering D_n. This is computed using a ratio of region areas, an example using the object arrangement (depth-ordering) in Figure 7.1 would be:

$$P(M_n = m_3^f|D_n) = (\eta_4 + \eta_5 + \eta_6 + \eta_7)/\eta, \tag{7.8}$$

where η is the sum of η_r over all regions, which equals the number of pixels N.

- $P(D_n)$ and $P(D)$: are the prior distributions on the discrete depth-orderings. They are taken to be the uninformative uniform distribution.

- $P(\mathbf{\Phi})$: is the prior on the shape $\mathbf{\Phi}$, which we take to be:

$$P(\mathbf{\Phi}) = \prod_{n=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp -\frac{(|\nabla \mathbf{\Phi}(\mathbf{x}_n)| - 1)^2}{2\sigma^2}, \tag{7.9}$$

where σ specifies the relative weight of the prior. This is identical to the prior term used in the previous chapter and automatically maintains approximate signed distance functions for the level-set embedding functions.

- $P(\mathbf{p})$: is the prior on the pose $\mathbf{p}$, which is either taken to be the uninformative uniform distribution or a motion model $P(\mathbf{p}^t|\mathbf{p}^{t-1})$, which will be described in Section 7.4.

Note:- It should be pointed out that when substituting these distributions into Equations (7.4) and (7.5) several expressions cancel, simplifying the final implementation (exact details skipped for brevity).

7.4 Tracking

Given the posterior distributions (7.4) and (7.5) our objective is to compute the MAP estimate for the pose $\mathbf{p}$, the shape $\mathbf{\Phi}$ and the discrete depth-ordering D. Ideally these would be maximised jointly, but the high dimensionality of the joint space makes this prohibitively expensive and so as an approximation we break the problem into five steps: (i) a rigid registration to account for any rigid motion between frames; (ii) a segmentation to account for any local shape deformation; (iii) a posterior over depth-ordering, holding the pose and the shape constant; (iv) updating the appearance models and (v) drift correction. We will now explain each of these five steps in greater detail.

Rigid Registration and Motion Modeling

If we begin by taking $P(\mathbf{p}^t|\mathbf{p}^{t-1})$ to be the uninformative uniform distribution (i.e. no motion model), then the MAP estimate for $\mathbf{p}$:

$$\mathbf{p} = \arg\max_{\mathbf{p}} \left\{ \sum_{n=1}^{N} \log P(\mathbf{\Phi}, \mathbf{p}, D|\mathbf{x}_n, \mathbf{y}_n) \right\} \tag{7.10}$$

can be solved (similar to Section 6.5) with an approximate version of the second-order Newton optimisation scheme:

$$\Delta\mathbf{p} = \left[\sum_{n=1}^{N} B_n^2 \mathbf{J}_n^T \mathbf{J}_n \right]^{-1} \sum_{n=1}^{N} \mathbf{J}_n^T B_n, \tag{7.11}$$

where

$$\mathbf{J}_n = \frac{\partial H_\epsilon}{\partial \mathbf{\Phi}} \frac{\partial \mathbf{\Phi}}{\partial \mathbf{W}} \frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \delta_\epsilon(\mathbf{\Phi}) \nabla \mathbf{\Phi}(\mathbf{x}_n) \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \tag{7.12}$$

and B_n is scalar value, which is obtained by differentiating (7.3) with respect to $\mathbf{p}$. Although the exact derivation of B_n is skipped for brevity it should be pointed

out that the differentiation of (7.3) leads to a sum of positive and negative weights, where the weights are related to the likelihoods $P(\mathbf{y}_n|M_n)$ and the region probabilities $P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)$.

Let us now introduce a constant velocity motion model $P(\mathbf{p}^t|\mathbf{p}^{t-1})$ with Gaussian noise, which will add an extra term to (7.10) and modify (7.11) to include prior terms:

$$\Delta\mathbf{p} = \left[\sum_{n=1}^{N} B_n^2 \mathbf{J}_n^T \mathbf{J}_n + \mathbf{J}_p^T \mathbf{J}_p\right]^{-1} (\sum_{n=1}^{N} \mathbf{J}_n^T B_n + \mathbf{J}_p^T \mathbf{B}_p), \tag{7.13}$$

where $\mathbf{J}_p$ represents the Jacobian due to the prior information and $\mathbf{B}_p$ is the error vector for the prior. This equation has the property that as an object becomes gradually occluded, the B_n terms will reduce, increasing the relative weight of the prior. At the point an object becomes completely occluded the B_n terms will be zero and (7.13) will reduce to a motion model prediction equation. Note:- The terms $\mathbf{J}_n$ and $\mathbf{J}_p$ remain constant with respect to the shape and can therefore be precomputed for efficiency during the registration step (refer to [Baker & Matthews, 2004] for details).

Depth-Ordering

We have two methods of dealing with the depth parameter depending on whether we treat it as a local (pixel-wise) or a global (frame-wise) parameter. Treating it as a local parameter means we can maximise for D_n within the registration step (7.10) by performing a pixel-wise maximisation over D_n:

$$\mathbf{p} = \arg\max_{\mathbf{p}} \left\{ \sum_{n=1}^{N} \max_{D_n} \{\log P(\mathbf{\Phi}, \mathbf{p}, D_n|\mathbf{x}_n, \mathbf{y}_n)\} \right\}. \tag{7.14}$$

This method picks the best D_n out of the $K!$ choices at each pixel and therefore has no consistency at the frame level. The benefit of this approach is that it can deal with objects interacting in complicated ways, for example: a pair of hands meshing fingers. The downside of this approach is that it is often easier to choose an incorrect depth-ordering than it is to explain the image data correctly, which results in increased

sensitivity to noise and a tendency to assume interacting pixels are occluded, when they are not. The alternative to this is to treat D as a global parameter and assume that objects do not interlock (we show in Section 7.5 that this increases resilience to noise). We first compute the following posterior over the $K!$ depth-orderings:

$$P(D|\mathbf{\Phi}, \mathbf{p}, \mathbf{I}) = P(D) \prod_{n=1}^{N} \frac{1}{P(\mathbf{y}_n)} \sum_{R_n, M_n} P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n) P(R_n|D, M_n) P(\mathbf{y}_n|M_n) P(M_n|D) \quad (7.15)$$

and then use the MAP estimate for D when optimising for the pose $\mathbf{p}$ and shape $\mathbf{\Phi}$ in the next frame.

Grouping

An efficient implementation must address the problem of exponential growth in $K!$. We deal with this by forming groups of objects, so that the amount of inter-object overlap within a group is maximised, subject to a maximum group size of three. The depth posterior is then computed on a per group basis. The result is that depth posteriors are only calculated where they are most necessary. Figure 7.9 shows a typical example of how objects are grouped, with the white lines representing the object groups. By limiting the maximum group size to three, we are able to track twelve or more objects comfortably in real-time.

Segmentation, Appearance Learning and Drift Correction

The methods for segmentation, appearance learning and computing drift correction are the same as the approach taken in the previous chapter, except that the level-set evolution in the segmentation step and appearance learning are modulated by a parameter λ_n, which effectively turns off learning in areas where objects overlap. The parameter λ_n is defined as the probability that objects do not overlap at a given pixel, which is computed directly using the $P(R_n|\mathbf{\Phi}, \mathbf{p}, \mathbf{x}_n)$ terms.

7.5 Results

We will first quantitatively evaluate the performance of the local vs. global depth parameter as described in Section 7.4. To test their performances against each other, we have generated a simple simulation containing three objects that repeatedly cross each other in different ways, we then add varying amounts of Gaussian noise to the pixel colours (see Figure 7.3). Given that we have the ground truth for the sequence, we are able to compute the normalised pose error $\sum_{i=1}^{F}\sum_{j=1}^{K}(\mathbf{p}_j^i - \hat{\mathbf{p}}_j^i)^T\mathbf{R}^{-1}(\mathbf{p}_j^i - \hat{\mathbf{p}}_j^i)/K/F$ i.e. a normalised error between the estimated pose and the true pose, where $\mathbf{R}^{-1} = diag[1, 1, 100, 180/\pi]$ and F is the number of frames[1]. Figure 7.4 demonstrates that superior performance is achieved in the presence of noise using the global parameter. This is no surprise given that the global parameter is a simpler model than the local parameter, yet both can explain the data exactly. It is also worth noting that using a motion model (MM) confers a measurable improvement in performance.

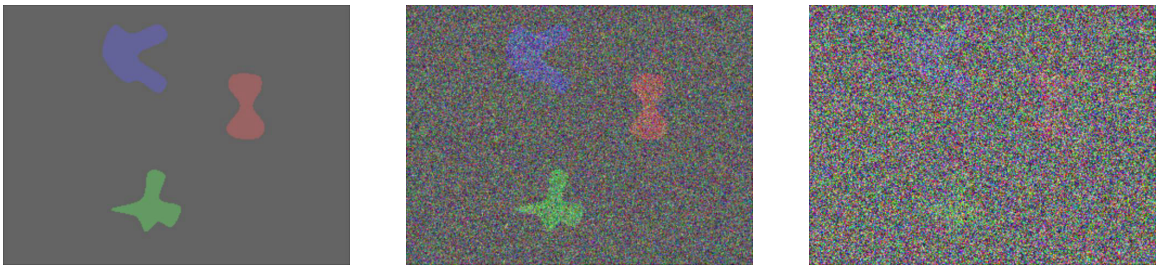

Figure 7.3: The simulation: these objects move around (translation, scale and rotation), overlapping each other in different ways. This shows varying amounts of noise, corresponding to standard deviations of 0, 40, and 70 pixels (from left to right).

Figure 7.5 illustrates that marginalising out nuisance variables to obtain the pixel-wise posterior (7.3) achieves superior resilience to noise compared with optimising the joint (7.1) directly. This graph was generated using a similar simulation to Figure 7.3 but with only a single object. The difference in performance is significant.

We will now qualitatively look at three sequences, the first is 1600 frames long and consists of three distinctive shapes that are translating, rotating and scaling, as well as occluding each other in different ways. Figure 7.6 highlights a small section of this sequence where all three objects interact. In particular, the green object passes behind the blue object (frame 1147), which then passes behind the red object (frame 1151), resulting in the green object being completely occluded and the blue object

[1]The purpose of $\mathbf{R}^{-1}$ is to normalise the pose dimensions so that an angle measured in radians can be compared against a scaling factor or a translation measured in pixels.

only being visible through the hole in the red object (frame 1157). Finally, by the end of the sequence (frame 1172) all objects have been successfully tracked through the occlusion.

The second sequence is 800 frames long and shows a typical security video taken in a shopping centre [Fisher, 2004] (see Figure 7.7). Our method successfully tracks four people as they walk from one end of the corridor to the other, occluding each other in different ways and crossing over completely between frames 0 and 207. In frame 369, a man in a striped shirt can be seen approaching from the right, then in frame 401 (zoomed in) he crosses behind the people we are tracking. At this point the green object shrinks to accommodate the unmodelled occlusion event, if we had been tracking the person in the striped top this would not have been the case, because the occlusion would have been modelled within the system.

Finally, the third sequence is 2000 frames long and shows a person carrying out actions that may be similar to those required for a human machine interface, see Figure 7.8. We successfully track both hands and the face for the duration of the sequence. As an example application, the tracker has been used to anonomise the face using the estimate for the shape and the pose. Frame 609 shows both hands occluding the face, sweeping from top to bottom. This is the sort of behaviour that would often break a tracking system. Frames 1231 to 1258 show the person rolling their hands one in front of the other, again behaviour that would often break a tracking system.

The depth-ordering is shown in the results by drawing the objects according to the MAP depth-order, which means that the coloured contour for an object will be hidden if it is behind another object. Generally the system gets the depth-ordering correct,

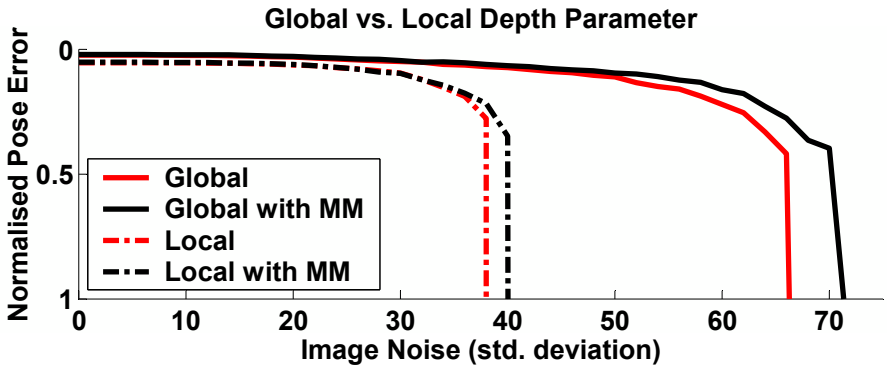

Figure 7.4: Comparison of global vs. local depth parameters, with and without a motion model (MM). The vertical regions in the plots at standard deviations of 40 and 70 correspond to catastrophic failures in tracking.

but occasionally makes mistakes if the appearance of the objects being tracked are similar. For an example consider frame 912 in Figure 7.8, where the person's hands overlap and the system makes a mistake about the correct depth-ordering. Although these mistakes are sometimes made, it is not necessarily detrimental to tracking because if the objects have similar appearances then the incorrectly classified pixels will have a weak influence during the registration stage.

Figures 7.10-7.12 show a selection of applications of the multi-object tracker to real-world examples. Figure 7.10 shows an application of the tracker to a marine environment where we track a selection of dinghys in a race and their safety boat. Figure 7.11 shows the tracker being used to track ice hockey players. Figure 7.12 shows the tracker being used to track every football player on a pitch, which is impressive for the following reasons: (i) no prior on human shape or appearance is used; (ii) no geometric information regarding the pitch is used and (iii) a detector is only used to initiate new tracks and not to correct tracking mistakes. Throughout the sequence the frame rate is always greater than 25 frames per second and averages 40 frames per second. Each of these figures display a stabilised video of the objects being tracked. These are shown as small tiles, which are overlaid on the video with a blue border.

7.6 Conclusions

We have presented a tracking method that represents multiple interacting/occluding objects with a probabilistic generative model. This model includes the discrete depth-ordering for the objects, their locations in the image represented by rigid transformations and their shapes represented with implicit contours. We show how to efficiently

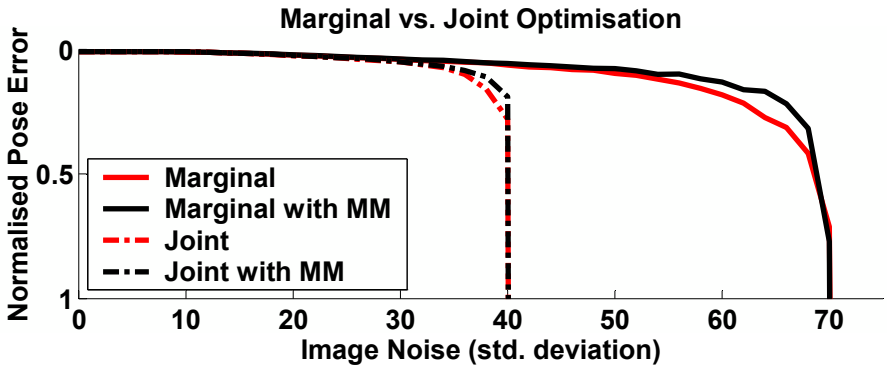

Figure 7.5: Comparison of optimising the marginal (7.5) vs. the joint distribution (7.1), with and without a motion model (MM).

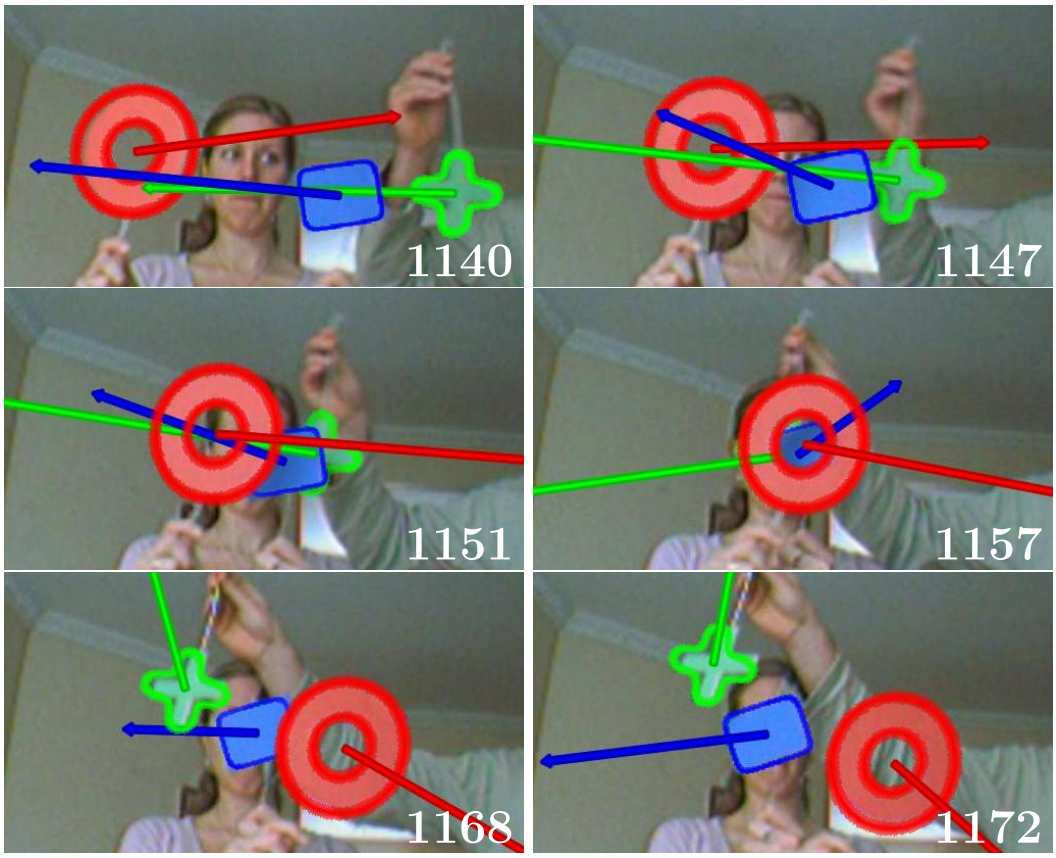

Figure 7.6: Tracking three distinctive shapes. The frame number is in the bottom right, the coloured contours show the objects' pose and shape estimates and the coloured arrows show the objects' estimated velocities.

perform inference on this generative model and compute the MAP estimate for the poses, shapes and discrete depth-ordering of the objects. By using implicit contours and a novel second-order registration scheme, we are able to compute this inference in real-time i.e. 30Hz. A key aspect to the success of the system is the fact that we marginalise out all nuisance parameters analytically, leading to pixel-wise posterior terms as opposed to pixel-wise likelihoods. We demonstrate using quantitative results that this provides superior resilience to noise in the image. We have explored two possible methods for representing the discrete depth-ordering, one based on a local (pixel-wise) parameter and the other on a global (frame-wise) parameter. We show quantitatively that the global parameter provides greater performance in difficult conditions. We have also shown how motion models can be included within our second-order registration step and how this enables the system to track complete occlusions. The system has been tested on a variety of challenging video sequences.

This concludes the second part of the dissertation, which has proposed two novel

methods for visual tracking based on pixel-wise posteriors as opposed to pixel-wise likelihoods. These methods can be used to acquire stabilised video streams of objects of interest with minimal prior information about the object itself (just a bounding box). The next part of the dissertation will describe how these methods can be combined with our HSLAMIDE to obtain an information rich estimate of the environment surrounding a marine vessel.

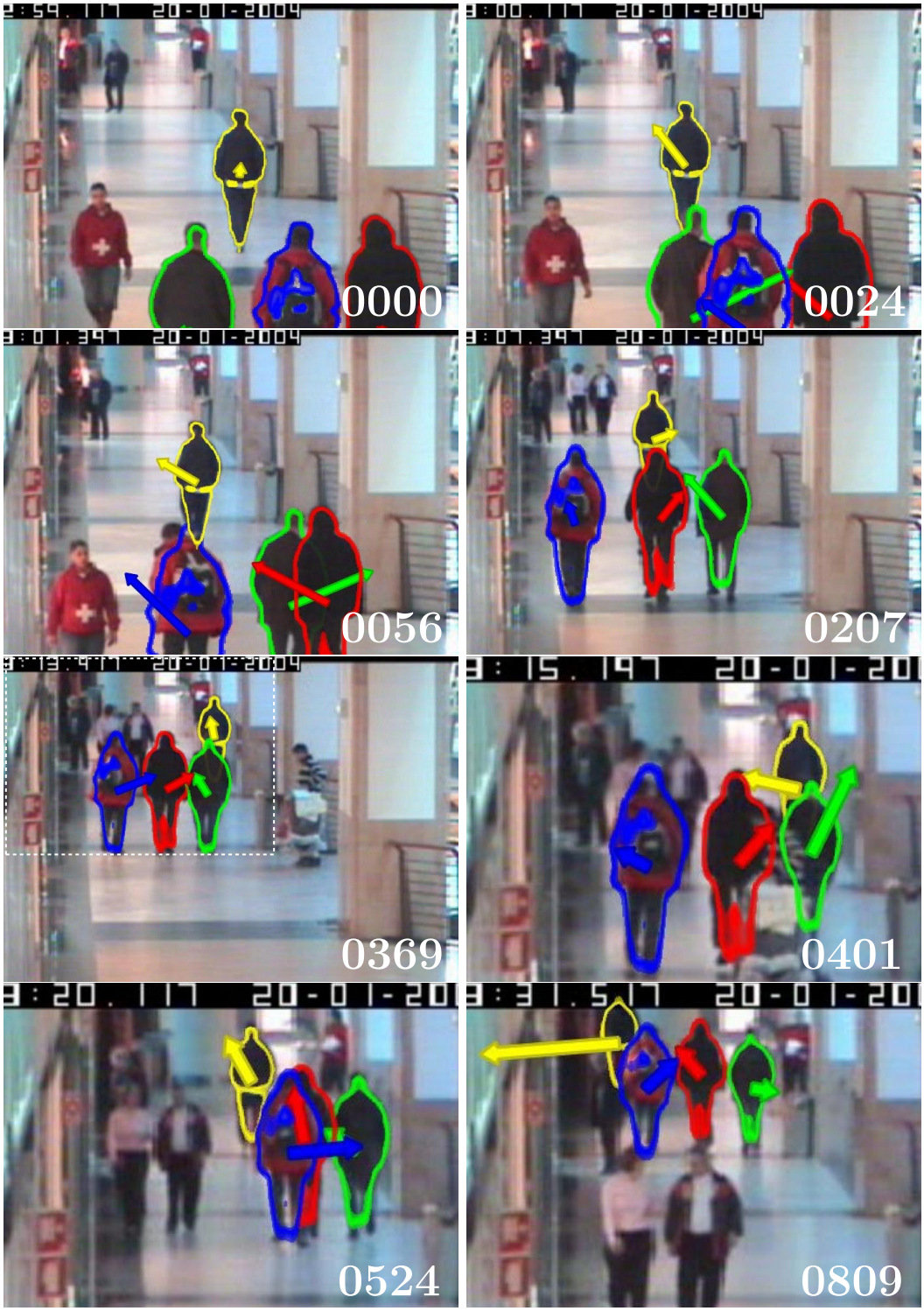

Figure 7.7: Tracking four people in a corridor [Fisher, 2004]. The frame number is in the bottom right, the coloured contours show the objects' pose and shape estimates and the coloured arrows show the objects' estimated velocities. The white dashed box in frame 369 shows the zoomed in area that is used for subsequent frames.

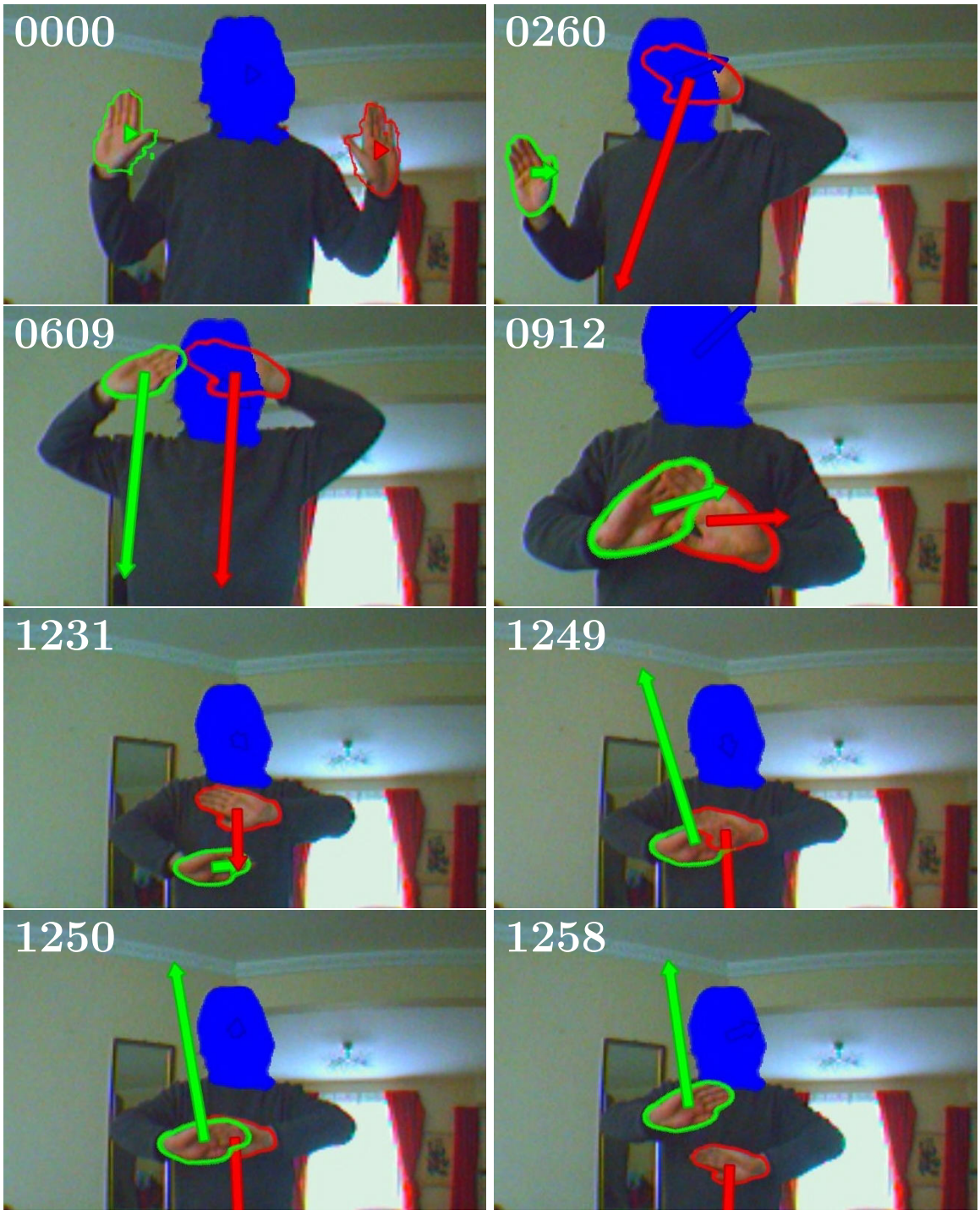

Figure 7.8: Tracking face and hands sequence. The frame number is in the top left, the coloured contours show the objects' pose and shape estimates and the coloured arrows show the objects' estimated velocities. The face has been anonomised using the output from the tracker.

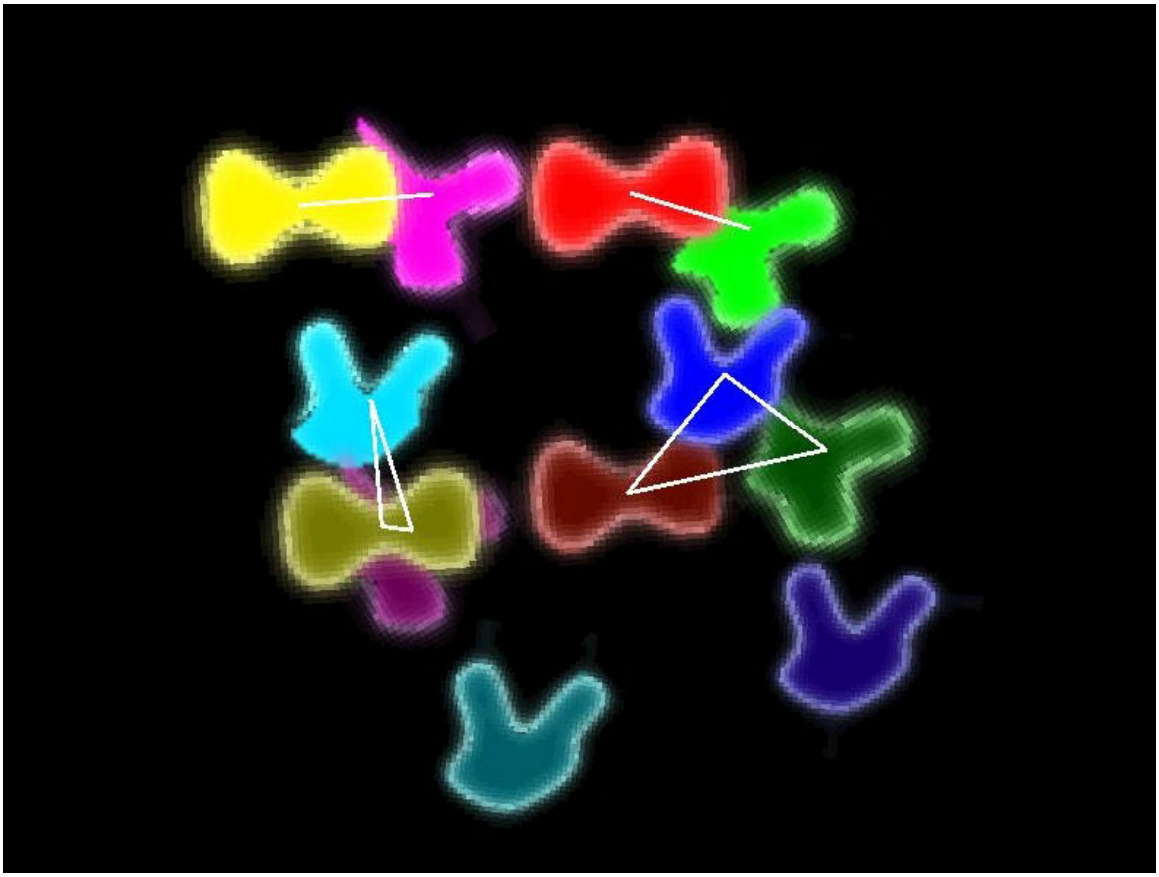

Figure 7.9: Shows the system tracking twelve interacting/occluding objects in real-time. The white lines show the current grouping of objects, which is used to limit the exponential growth when computing depth-orderings.

Figure 7.10: Tracking boats.

Figure 7.11: Tracking ice hockey players.

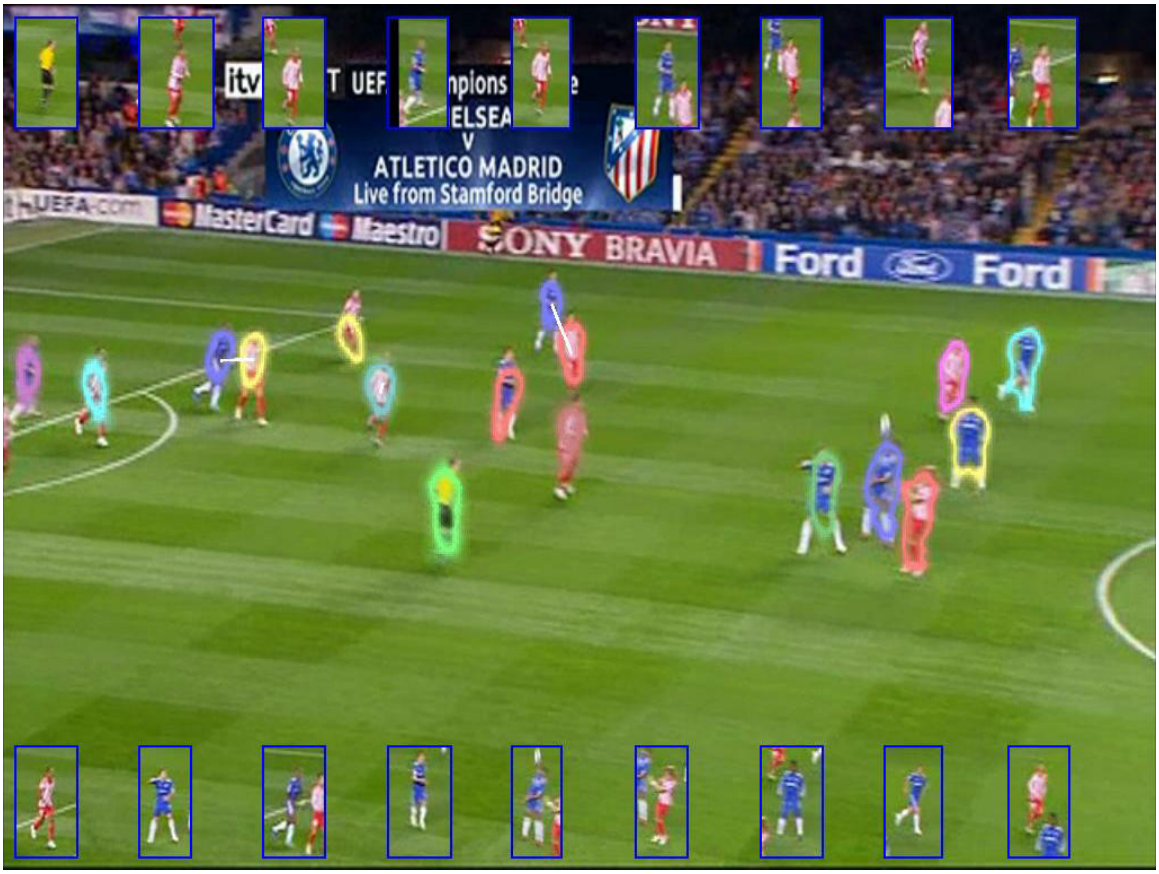

Figure 7.12: Tracking football players.