Ownership Masks, Evolving Views and Cooperative Templates in Template Tracking

by

Alan Stuart Angold

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, May 2003

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Alan Stuart Angold

Abstract

A template tracker is a tracker based on matching a pre-initialised view of an object with the object's view in an image sequence. Using an error function, the intensity difference between the template view and the templated region in the image is measured. This error measure is used as the basis for a template alignment algorithm that will adjust the template's pose to more accurately register the template view with the view of the object in the image.

Some significant problems present themselves with this simple tracker. Extraneous, or non-object, pixels within the template boundaries can cause bias in the registration of the template. Partial occlusions of the object's view in the image can also cause serious bias in the template's pose. Beyond simple occlusions there are transits of occlusions across an object. Occlusion transits are significant because over time they can occlude the entire object in an incremental fashion. If initially the template view is not completely known this kind of occlusion can easily cause a total tracking failure for an object.

In this thesis three enhancements of the basic template tracker are proposed: Ownership Masks, Cooperative Templates, and Evolving Views. Ownership Masks are aimed at eliminating the extraneous pixels from the template view. Cooperative templates are used to separate the intensity probabilities when more than one template covers a pixel. Building upon both Ownership Masks and Cooperative Templates, Evolving Views update the template views when occlusion transits are a problem.

With these enhancements we have been able to increase the accuracy of tracking objects where large portions of a template contain background pixels. Also occlusions and some types of unocclusions can be detected and discriminated. Finally, some failures in the basic tracker due to occlusion transits have been overcome.

Acknowledgements

First thanks first to my supervisor, Richard Mann, for the many times that he helped and directed me to my goals, and the times he put up with my grumpy behaviour when things were getting rough. Thank you as well for the financial support.

Thanks to my readers, Richard, Dale Schuurmans and Peter van Beek for taking time out of their hectic schedules to read and comment on my thesis.

Thanks to my wife and children for tolerating me throughout this process and for distracting me when I got overwhelmed. Thanks to Cari, my wife, for supporting me and making sure I knew I could complete this even when I had my doubts. Thanks to my children, Jordan, Brittany, Hayley, and Zachary for making me realize that I should put my education where my mouth was and demonstrate what I believed: education is a valuable commodity.

Finally I would like to thank my parents who, with their generosity, thoroughly stopped up that last loophole that would have let me give up when I shouldn't have. Thank you as well for being supportive.

Almost forgot ...

Thank goodness I am finally through with this so I can get on with all those things that were distracting me along the way and ... perhaps a PhD next term.

Contents

List of Figures

Chapter 1

Introduction

Tracking is used in many applications and fields from security systems to robotics to gesture recognition and air traffic control systems. Accuracy and robustness in tracking are important to allow for accurate interpretation of the given data and to prevent misconstruing one tracked object for another. Without accuracy a robot may run into objects or people, or fall down stairs; aircraft may come too close to others on their flight paths, and security systems may not register an intrusion by a criminal. Systems must be robust to occlusions and other signal noises such as background clutter to prevent inaccurate tracking, loss of tracking or tracking the wrong object. With the tracker described in this thesis we attempt to increase the accuracy of a simple template tracker to cope with some typical problems encountered in visually tracking objects as well as increase its robustness in the presence of some types of occlusion.

Tracking is the process of identifying and locating an object within a chronological sequence of sensor readings. Readings can be derived from a variety of sensors including: cameras, sonars, radars and other devices that return periodic signals representing some characteristic or aspect of the object of interest.

Various characteristics of an object may be used to track. In vision-based systems a predefined view of the object may be used to search for that object in successive frames of an image sequence. Corners, edges, colour blobs and textures, among other characteristics have also been used successfully to track.

1.1 Template Tracking

Template tracking, which this thesis deals with, is based on a sequence of camera images in the visible spectrum. It is based on the registering of a previously defined view of the object to be tracked within each frame of an image sequence. The registration in the basic tracker is based on finding the displacement that minimizes the squared error of the differences between itself and the displacement estimates from each pixel in the template view. The displacement estimates are derived from the image gradient and the difference between intensities of the template pixel and the corresponding image pixel, at each pixel location.

Template trackers work well in a variety of circumstances but are sensitive to a number of typical visual problems. Both occlusions by other objects and background pixels within the area of the template view can cause biasing of the template pose away from its correct position.

In this thesis we modify the simple template tracker to deal with these partial occlusions and background pixels. The remainder of this chapter provides an introduction to the problems of the basic tracker, the outline of the proposed modifications that solve these problems and some details about previous and related work.

1.1.1 Tracking Problems

To simplify "templating" an object to be tracked, a rectangle or other low degree polygon is often used to define the templated region. This simplification has its drawbacks, however, if the region exposes background pixels in its interior. If the proportion of background pixels within the template is large or an occlusion covers the tracked object, a basic template tracker can fail (Fig. 1.1). In the top row, the number of extraneous (or non-object) pixels within the hand template and the strongly textured background cause a tracking bias. The Coke-Put example (middle row) fails as the hand transits over the pop can because it obscures the portion of the pop can being tracked (the top of the can). The tracking on the telephone fails (bottom) due to the incoming occlusion of the hand and the lack of significant texture on the bottom of the phone.

1.2 Enhancements to Template Tracker

To solve these problems three modifications to the basic tracker were developed. These we call: Ownership Masks, Evolving Views, and Cooperative Templates. These new techniques are described briefly below and developed further in the following chapters.

1.2.1 Ownership Masks

In a template tracker the algorithm uses a partial or complete view of the tracked object to register the template within the image. Foreground and background pixels within this view may have different velocities as the tracked object moves. This difference in velocities tends to bias the tracker away from the pose of the tracked object. To reduce this effect, a template should be initialised to be 'tight' around the object to minimise the number of background pixels within the template boundaries.

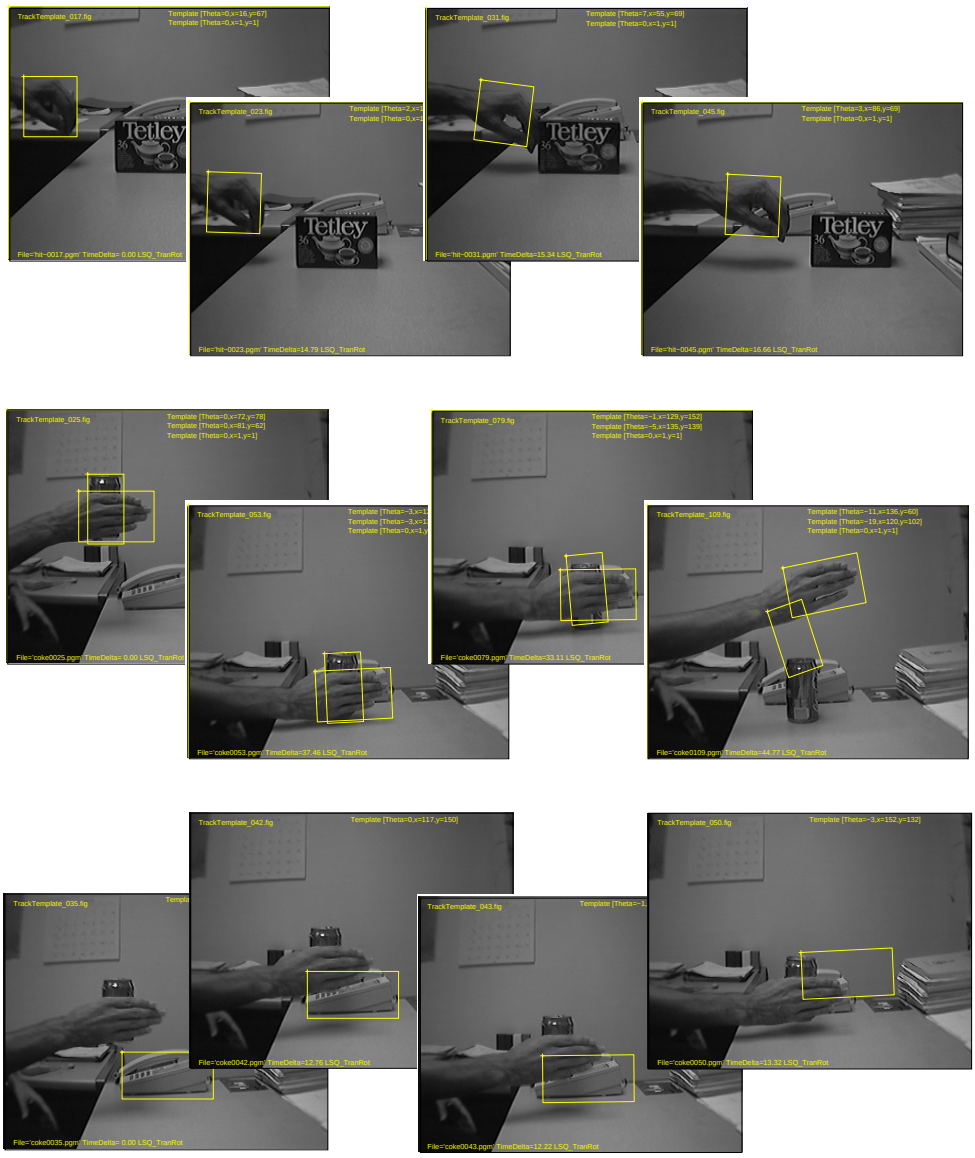


Figure 1.1: Problems with basic template tracker. Box-Hit example (top) fails because of extraneous (non-object) pixels around the hand. A transit of the hand over the tracked part of the pop can causes problems in the Coke-Pick example (middle). The occlusion by the hand causes tracking failure on the phone template.

Simple rectangular bounding boxes around an object are easier to initialize and maintain than complex polygons and are therefore preferable. However, in Fig 1.2 we can see that this can lead to a significant number of background or extraneous pixels within a template's bounding box. Thus, simple boundaries present a problem with irregularly shaped objects: how to eliminate the background pixels and keep the foreground pixels when making our pose estimate.

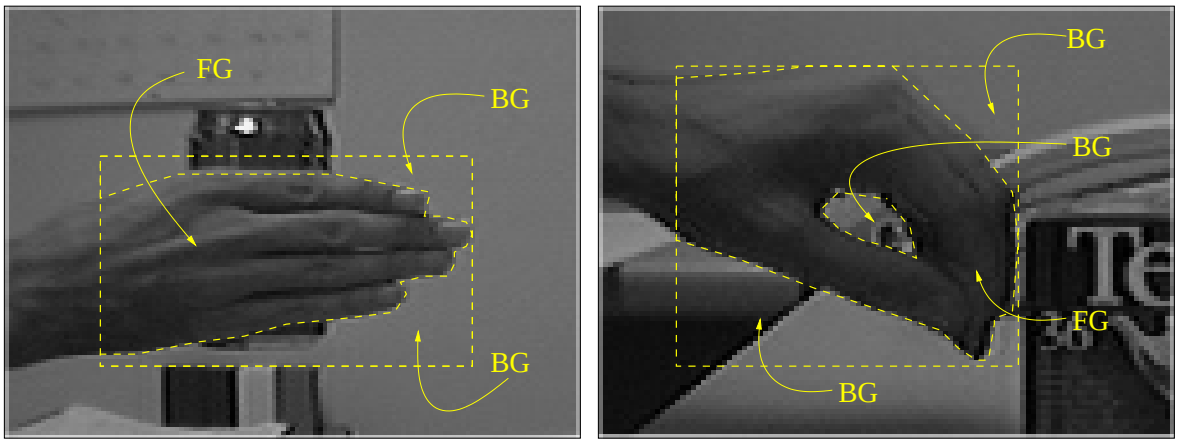


Figure 1.2: Ownership Mask and object extent. The template's rectangular bounding box and the irregular boundary of the tracked object introduce extraneous background pixels into the pose estimation.

When a template is initially placed on an image sequence there is no indication what is to be tracked (other than the rectangle of pixels bounded by the template). The boundaries of the tracked object may be difficult to find but even if this were not so deciding which of the objects in the region to be tracked based on the pixel values within that region is not a simple job for a low level algorithm. People can see the object and distinguish its extent and boundaries but with a low level tracker (such as a template tracker) there is no high level knowledge of what constitutes an object.

Without this high level knowledge we must look only at the pixels themselves and perhaps the initial locations and extents of the templates. But even using this low level information we can, as suggested in Fig 1.2, derive the extent of the tracked object. Simply having a template encompass an area of the image implies some information about the extent of the object to be tracked.

When the tracked object starts moving, the template follows and the object pixels will be static in the view. The background pixels, assuming the background has some texture, should be varying as the template moves over them. Therefore, the variance of the background pixels within the template should be large and that of the tracked object's pixels small.

A Gaussian distribution was chosen as a good model of pixel intensity noise. Using a Gaussian to represent the distribution around the mean value of each pixel's intensity we can determine which pixels are background. We use the error in pixel intensity [1] to derive the probability that the two pixel intensities represent the same object point.

Here we assume that the object to be tracked is within the template boundary, and that it covers a significantly larger portion of the template than the background. Otherwise the tracker, which is initially based on a consensus of all the template pixels, may be biased away from the correct location. If initially we have accurate tracking, variance within the template will come from the change in the background and not from the poor registering of the template. However, even with initially poor tracking the ownership masks will evolve properly but will maintain an initial bias.

Ownership Masks are discrete masks that cover the same region as the template view. They are used to down-weight the pixels that are not part of the tracked object and fully

[1] The pixel intensity error is the difference between the template view's pixel intensity and the corresponding pixel in the image sequence

weight the object pixels when doing the registration calculations. Each of the elements in the mask is associated with one template pixel and for this reason will be referred to as "mask pixels". The mask pixels are set to one wherever we are certain the pixel belongs to the tracked object in the template view, zero where we are certain it does not belong and values between these two extremes to suggest our degree of certainty.

In Figure 1.2, ideally the FG pixels would be set to one and the BG pixels to zero.

When the template tracking software is registering the tracked object it uses these ownership masks to mask out the pixels in the template image that do not correspond to the object (see Figure 1.3). Here Frame 30 of the Coke-Put sequence is displayed (a) along with the corresponding ownership probabilities (b) and masks (c) from the three templates that cover the hand, pop can and background (whole image). Notice in (b) the background regions are black where the hand and can regions are white.

The ownership probabilities shown for the hand demonstrate how the probabilities are higher for the finger tips and wrist because the variance in the intensity between the template and the image is low. The background in the hand template shows as grey because the background is varying as the hand moves. The middle portion of the hand and lower portion of the pop can are grey because probability is assigned to each object equally (more about this in the next section).

Because the non-object pixels are down-weighted or totally eliminated from the tracking equations their influence in biasing the tracker away from the correct pose should also be reduced or eliminated. Thus the Ownership Mask should remove the contribution from these other groups of pixels of different velocities from consideration in the tracking algorithm.

When a tracked object starts out unoccluded and is then occluded in the course of the image sequence, we would like the tracker to be able to distinguish which parts of

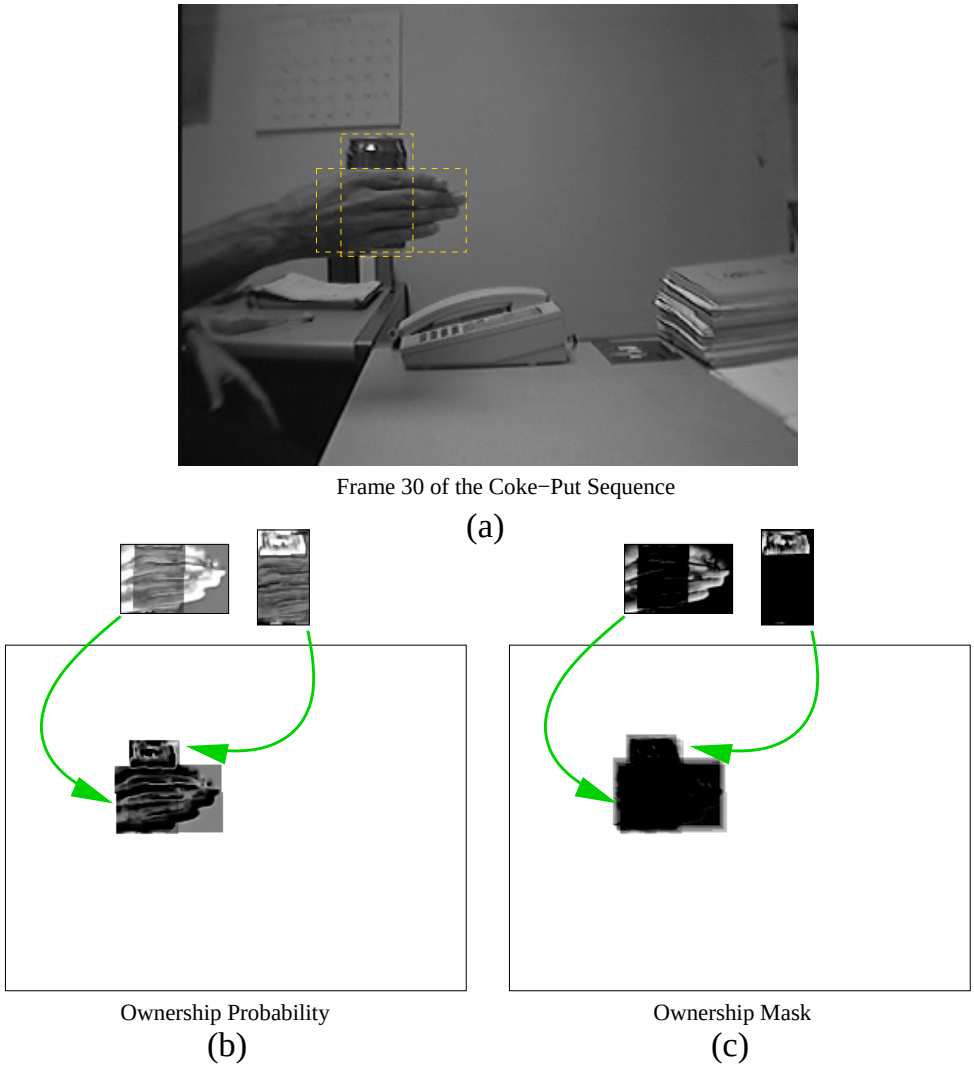


Figure 1.3: Ownership probability and corresponding ownership mask. Frame 30 of the Coke-put sequence. In (b) the ownership probabilities for each of the three templates are shown. The corresponding three ownership masks (c) are derived by thresholding the probabilities. The arrows indicate the corresponding region in the background template where the smaller templated regions would occur.

the image correspond to the tracked object and which belong to the occluding object(s) and the background. The ownership masks should have the same effect in this case: they should eliminate the occluding and background pixels from the tracking equations that are causing a tracking bias. Therefore, the tracker should track more accurately even through partial occlusion events.

1.2.2 Cooperating Templates

Another problem arises when we consider the example given: the hand putting the pop can down. In this case the pop can is initially occluded and since the two objects (hand and can) travel together for the first half of the sequence there will be no evidence from the pixel variance to distinguish the two objects. For this reason another idea was developed: *Cooperative Templates*.

Cooperative templates will allow us to take advantage of extra information in the areas of the image where templates are overlapping. We will use Bayes law to convert the prior probabilities of each template and the probability that the template pixel intensity matches the image pixel intensity to derive a segmentation of the image pixel to one of the covering templates. Cooperative templates will also allow us to distinguish occlusions and different types of unocclusions as they happen.

In this example, if we assume that initially the occluding objects were templated then the templates will be overlapping (see Fig 1.4). This overlapping can be used to help segment the templates. In the areas that are not occluded the templates 'see' their tracked object. Thus the pixels in those areas belong to those templates. Where the objects occlude, the templates overlap. With no prior knowledge of the segmentation of the pixels it seems reasonable that the probability for each of the pixels in the overlapping section should initially be evenly distributed between the templates: the pixels are equally likely

to have come from any of the covering templates. Where the hand occludes the pop can the tracker has, initially, no evidence that the pixel has come from the hand rather than the pop can. Note that this does not attempt to model depth layers, occlusions or spatial correlation.

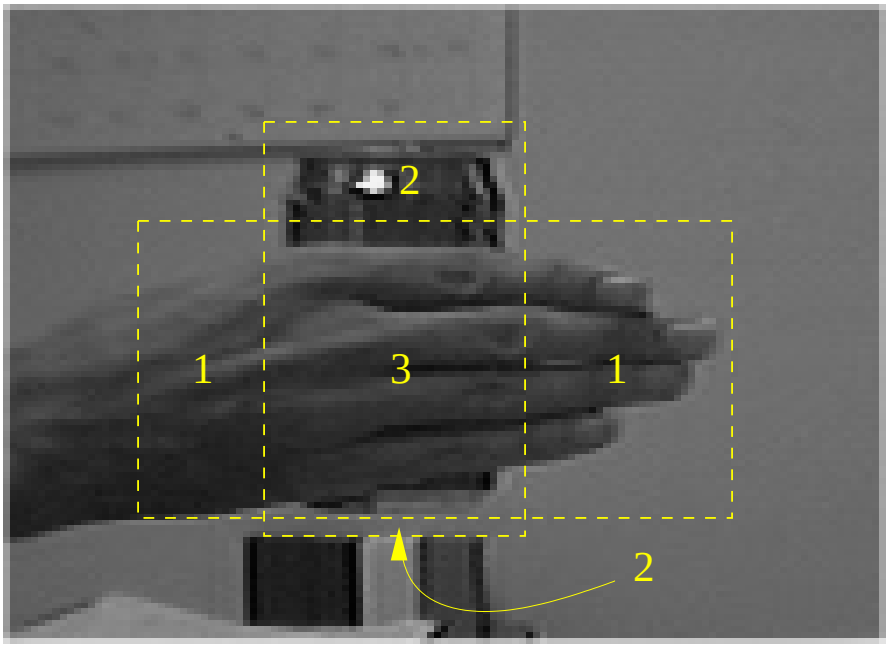


Figure 1.4: Cooperating templates. When templates overlap they must cooperate when assigning ownership for the pixels in the common areas. Implicitly the hand template will track the (1) areas, the can template the (2) areas because no other template covers these portions of the image. However, the (3) area is covered by both templates. The 'ownership' will be shared between the templates until there is evidence that these pixels belong to one template or the other.

We assume that each template covering an occluded pixel has, as a first approximation, equal prior probabilities. Using these prior probabilities and the per-pixel Gaussian

probabilities in Bayes' theorem we come up with an ownership probability for each pixel in each frame. This ownership probability is then accumulated in the template's cumulative ownership probability array as the image sequence proceeds. The ownership masks are a thresholded version of the cumulative ownership probabilities.

If a pixel is covered by more than one template then the cooperative template method will keep the ownership probability low for each template until the pixel variances (as the template moves) assign more or less probability to that template.

The cooperative template and ownership mask methods also allow us to update our template views in the case of occlusion and some unocclusion events. When a tracked object becomes unoccluded the probability of the newly unoccluded pixels will increase [2] for the templates still covering the pixel. Thus the probability of the pixel being owned by those templates goes up. If this ownership probability goes up enough then we can conclude that the pixel belongs to the tracked object and hence the template. When we have determined that the pixel belongs to the template, then the ownership mask for that template can be updated to reflect this change. The increased area unmasked on the object will allow for more accurate tracking.

Similarly when the ownership probability drops significantly this is an indication that an occlusion is in progress. When this happens we refrain from updating the template view because otherwise the template view would be updated with the image of the occlusion.

This process is displayed in Figure 1.5.

[2] If the template view was previously known at this pixel. Template view pixel intensities may not be known if the object has always been occluded.

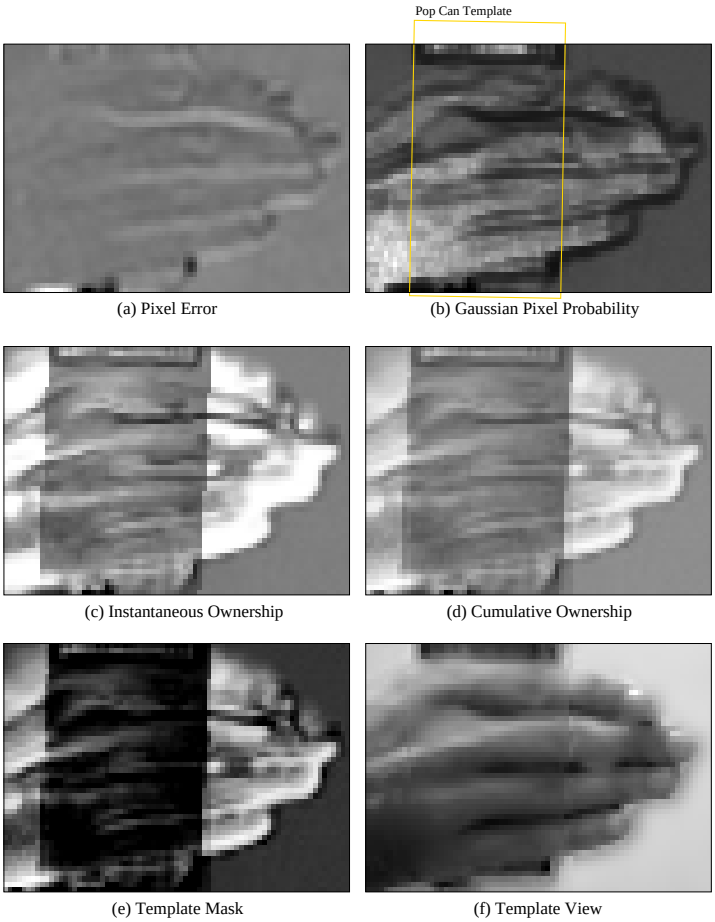


Figure 1.5: Evolution of probabilities in the hand template. The per pixel error (a) is used in a Gaussian distribution to calculate the per pixel probabilities (b). These probabilities are used in Bayes' formula to derive the instantaneous ownership probabilities (c) which are compounded over time into the cumulative ownership (d). A thresholding of the cumulative ownership gives us the template mask (e). Finally the template mask and cumulative ownership are used to update the template views (f). This template view is then used to determine the pixel probabilities at the next time step. Note the approximate position of the pop can template is shown in (b).

1.2.3 Evolving Views

As alluded to above there are situations where a complete template view cannot be found initially. Through unocclusion events [3] we can find the portions of the template view for the previously occluded object that we did not have access to initially. The Cumulative Ownership probability and Cooperative Template method can be used to our advantage here. What we have done is used changes in the Cumulative Ownership probability derived from the Cooperative Templates method to trigger the acquisition of the newly exposed regions of a previously occluded object.

In the case that the newly exposed area of the object was previously unknown (because historically it has always been occluded) the pixel probabilities may decrease. This is due to the newly exposed pixels being part of the tracked object and the template view pixels being part of the occluding object. An example of this is the "Coke-Put" sequence where the templates were initialized when the hand occluded the pop can and later in the sequence a portion of the can not previously seen was exposed. However, since the tracked objects no longer overlap the templates tracking them may not be overlapping [4]. Since the distribution of probability in the cooperative templates method is inversely proportional to the number of templates covering these pixels, and this number has decreased, the probability of that pixel belonging to the newly exposed template increases. This increase in probability produces a positive change in the cumulative ownership probability and this is then used to trigger an update to the template view.

The enhanced template tracker has thus 'learned' the view of the tracked object in the

[3] An unocclusion event is when a previously hidden portion of an object is exposed when the occluding object moves away.

[4] Due to the margins of background pixels in the template region surrounding a tracked object the unocclusion of the templates may not occur at the same time as that of the occluding objects.

initially occluded portion of the object.

With *Evolving Views* we have attempted to use the change in the cumulative ownership probability to indicate where the occlusion or unocclusion is occurring in the template and use this occlusion information to update the template view (see Fig 1.6). When we know where the unocclusion is occurring we can use the current view of the object in the image sequence to update the template view. The changes in the ownership mask also allow us to update the list of pixels which we want the tracker to use to register the template.

1.3 Previous Work

There has been extensive work done on tracking. Much of this work has incorporated tracking through occlusions implicitly. The following documents some of the more common tracking methods and highlights those that deal with occlusions.

In Lucas&Kanade's 1981 paper [20] they describe an image registration method that has become known as template tracking. They laid out all the basic details including the error metric that the registration strategy is based upon, the course-fine search strategy (see also Rosenfeld [27]), and the Newton-Raphson iteration in the intensity gradient space. This defined the basic template tracker that we use in this paper. However, their tracker had no means of detecting or dealing with occlusions. It would fail to a greater or lesser degree depending upon the amount of occlusion.

Earlier Vanderbrug and Rosenfeld [30] had done work to deal with the problem of articulated objects. They used a two-stage template matching method that first matches subsections of the object that are less prone to distortion than the whole object (for instance arm and leg segments when tracking a whole person).

Brunelli and Poggio [6] attacked the pattern deformations problem with matched spatial

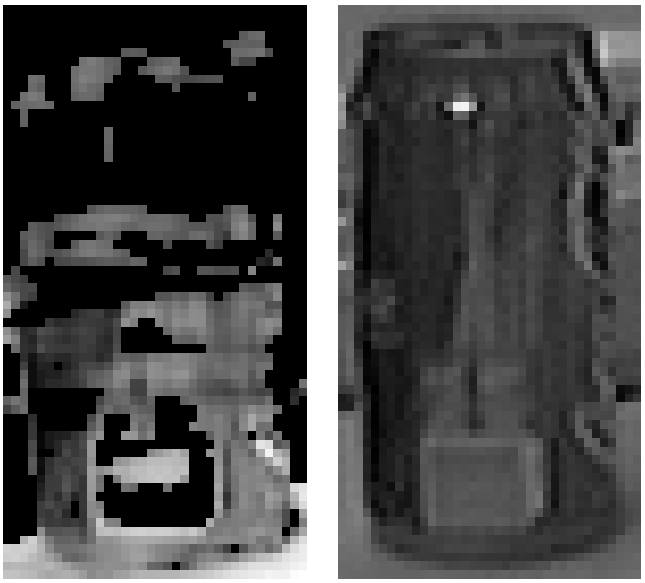


Figure 1.6: Updated template views. When a tracked object (pop can) is initially occluded we must use motion information to tell us when to update the template view. In the Coke-Put sequence (see middle row in Fig. 1.1), the hand has been raised so that it only obscures the upper half of the pop can. In image (a) the pop can template view's originally occluded lower portion is starting to appear. Image (b) shows how the pop can appears in this orientation.

filters and achieved some success. They also reviewed and compared a number of other approaches to template matching. Berger [1] continued to develop Lucas&Kanade's LSQ tracker with extensions to error propagation and self diagnosis. Brown [5] has done a survey of image registration techniques that may be useful. Tian and Huhns [28] continue this with a survey and analysis of a number of subpixel registration techniques. Meer et. al. [22] delve more deeply into the problem of making the registration robust in terms of fitting a model to noisy data.

Black and Jepson [3] use a technique using views to allow them to track rigid and articulated objects. Using an eigenspace representation of an object's view in various poses they are able to track objects that change appearance in a limited fashion. Using weighted amalgamations of the eigenspace basis they can reconstruct an approximation to the current view of the tracked object. The subspace consistency assumption and corresponding method extends the eigenspace concept to allow for occlusions, background clutter and noise. This work generates a mask separating inliers from outliers by thresholding the error between the image and an eigenspace representation of the tracked object. This mask, however, assumes a standard variance in the per-pixel error over the entire image. This paper also makes an implicit assumption of a background model as was done in the Ownership Mask chapter and found to be problematic. The Black&Jepson paper also requires a pre-initialized eigenspace which was seen as a problem because our distant goal is a fully automated tracker without any sort of bootstrapping.

Jepson et. al. [18] developed a method to deal with occlusions and natural appearance changes such as facial expressions and changes in 3D pose. Their method was based on a mixture of a stable model, a wandering model, and an outlier model. The EM-algorithm was used to adapt the appearance model parameters over time. This system seemed to deal well with partial occlusions and evolving appearance models. We go beyond the goals

of this paper in one direction. For our Evolving Views technique we assume that initially we do not have a complete view of the tracked object. This can happen when one object, partially obscuring another object, carries it into the image sequence. Thus we not only track the object but also extract an appearance model which will be useful in further work when we initialize the tracker from motion clues alone.

Wren et. al. [32] develop a multi-class statistical model using region-based features of colour and shape to obtain a 2D representation of a human's hands, arms and body in motion.

Segmentation has been addressed by a number of people. Collins et. al. [7] developed a tracking system called VSAM (Video Surveillance and Monitoring). This system uses three-frame differencing to detect motion and initialise tracking and adaptive background subtraction.

The Kalman filter has been used extensively in the past to track objects. A good introduction to the Kalman filter is Welch&Bishop's introduction [31].

Based upon the Kalman filter and Reid's algorithm for tracking multiple objects [26] is the Multiple Hypothesis Tracking (MHT) algorithm of Cox and Hingorani [8] and [10]. This algorithm tracks a number of image features (such as corners) from one frame to another and uses a hypothesis tree and Reid's algorithm to take care of the uncertainty in associating points with tracks. The MHT algorithm takes care of track initialisation, termination and occlusion of tracks for a limited period. The MHT algorithm is further developed in [9], [10], [23] and [11].

Working from the basis of particle filtering, Isard&Blake developed the Condensation ([16],[15]) method of tracking. This method uses a cluster of points to represent a non-standard density distribution. This method has been useful where keeping simultaneous alternate hypotheses of the object's position is necessary (in situations where there is

dense visual clutter in the background). This method is robust in the case of temporary occlusion but does not identify objects with their tracks when multiple indistinguishable objects are being tracked and may drift with time. MacCormick and Blake [21] attack this problem using a probabilistic exclusion principle which prevents a single piece of image from independently contributing to similar hypothesis for different targets. Isard later extended Condensation with the use of automatic model-switching [17] to satisfy the problem of abrupt changes in dynamic model. For instance: the cusp in the motion of a bouncing ball or a three state drawing model where hand motion distinguishes the task being performed. Further development using the Condensation algorithm was done in [12] and [4].

Elgammal and Davis [13] use a person model that is based on segmenting the body into regions in order that colour regions can be spatially localised. Modeling these colour regions depends upon modeling the colour distributions as well as their spatial distribution with respect the the position of the body. Initialisation of the segmentation is based on training data. The people being tracked must first have their segmentation initialised before occlusion occurs. Occlusion reasoning is done by labelling each pixel, after segmentation, with their model. The elliptical regions corresponding to the people are then compared, using an error measure, to these labels and the occlusion model corresponding to the smallest error is selected.

There are many other papers dealing with tracking including the following papers that present some interesting branches.

Birchfield developed a method [2] to track peoples' heads modelled by an ellipse. The ellipse's pose was continually updated using two global statistic measures: intensity gradients around the ellipse's perimeter and colour histograms within the ellipse. Global statistics measures like this have difficulty in the accuracy of the registration of the model to the image. They also tend to fail when nearby regions have similar statistics (colour or

grey scale distributions) as the model. However, Birchfield's method did seem to deal well with severe but brief occlusions and was tested by Birchfield and appeared to be resistant to the similar background patterns.

Ju, Black and Yacoob's [19] paper on Cardboard People demonstrated a method for tracking parts of an articulated object using connected planar patches. The motion of these patches was constrained by a distance measure between corresponding points on the connected planar patches.

Chapter 2

Basic Template Tracker

Tracking with templates is a method used to register a view of a tracked object with its corresponding view in each frame of a sequence of images. This is accomplished by comparing pixel intensities between the template view and the view in the image sequence. These pixel comparisons are used in an error function which is both a measure of the registration accuracy and the basis for a iterative registration method that will move the pose of the template closer to the correct pose.

In this chapter we will go through the basic operation and mathematics of a template tracker. We will see where the basic tracker performs well and where it fails and why. A more detailed mathematical treatment can be found in Appendix A.

2.1 The Template

A template consists of a number of basic elements. The template view is the image of the object to be tracked. In this thesis we initialize the template view automatically using a template manually placed in the initial image in the video sequence. The template view can

also be manually initialised off-line with an independent view of the object to be tracked. Since part of the long term goal is to fully automate this tracking process with methods to initialise the template view, the automated method was preferred.

The pose is the vector of parameters that define the transformation from the pixel's template coordinates to the image's coordinate system. When the basic tracker is enhanced in later chapters additional information about the template will be retained.

2.1.1 Pose Transformation

The template's pixels in the template coordinate system are from $(0, 0)$ in the upper left to $(H-1, W-1)$ in the lower right where H and W are the height and width of the template in pixels. These coordinates, $\vec{x}$, are transformed into the image coordinate system using the template pose transformation, $m(\vec{x}; \vec{a}_T)$, and the pose parameters $\vec{a}_T$ (see Figure 2.1)[1].

Since this transformation normally takes the template pixel coordinates into non-discrete locations within the image, an interpolation method must be used to derive an approximation to the intensity of the template pixel. Bilinear interpolation (see Appendix B.3) was chosen to interpolate the pixel locations because of its simplicity and low computational cost.

2.1.2 Error Equation

The basis of a template tracker is the pixel intensity error equation (see Equation 2.1).

[1] After the template is correctly registered, the template's transformational parameters define the pose of the template within the image. Later, in the Cooperative Templates chapter (Chapter 4), the template's pose transformation will be used to further develop the Ownership Masks.

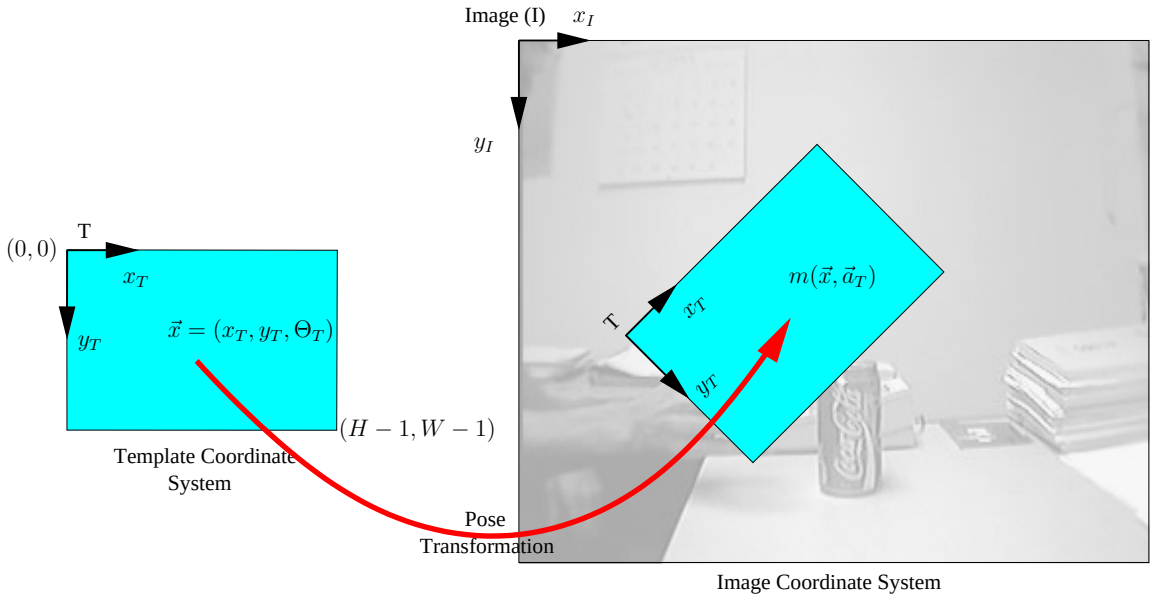


Figure 2.1: Pose transformation from the template's coordinate system to that of the image. The pose transformation transforms the pixel positions from the template's own coordinate system to that of the image.

$$\mathcal{E}(\vec{a}) = \sum_{\vec{x} \in T} \rho(I(m(\vec{x}; \vec{a})) - J(\vec{x})) \tag{2.1}$$

The total error $\mathcal{E}(\vec{a})$, given the current pose $\vec{a}$, is the sum of the individual pixel errors adjusted by an estimator function ρ, where $J(\vec{x})$ is the pixel intensity in the template view and $I(m(\vec{x}; \vec{a}))$ is the corresponding intensity in the image after the template, T, is transformed to its position, $m(\vec{x}; \vec{a})$, in the image. In our case we consider translation and rotation only, so the template pose is described by the vector (x, y, Θ). Note that eqn. 2.1 is a nonlinear function of the pose $\vec{a}$ and no closed form solution exists. We will use an iterative registration technique to solve for $\vec{a}$.

This equation is the total error between the intensity of the pixels in the image and those in the template view. A smaller error between the pixels in the template view and those in the image suggests a better registration. Generally the total pixel error varies in proportion to the accuracy of the template's registration. The exception being when subsections of the template view match other areas of the image and misleadingly reduce the total pixel error [2]. Since we use an iterative registration method local minima result and can be a problem. Typically there are numerous minima in which the tracker can get trapped (see Fig. 2.2). This problem is partially alleviated by the use of a Course-to-Fine algorithm which is touched upon briefly in a later section.

2.1.3 Estimator

The error equation uses a metric (or estimator) ρ to define a distance measure for the error in the pixel intensities. Various estimators ρ can be used depending on the criterion

[2] Such as when tracking a polar bear in a snow storm.

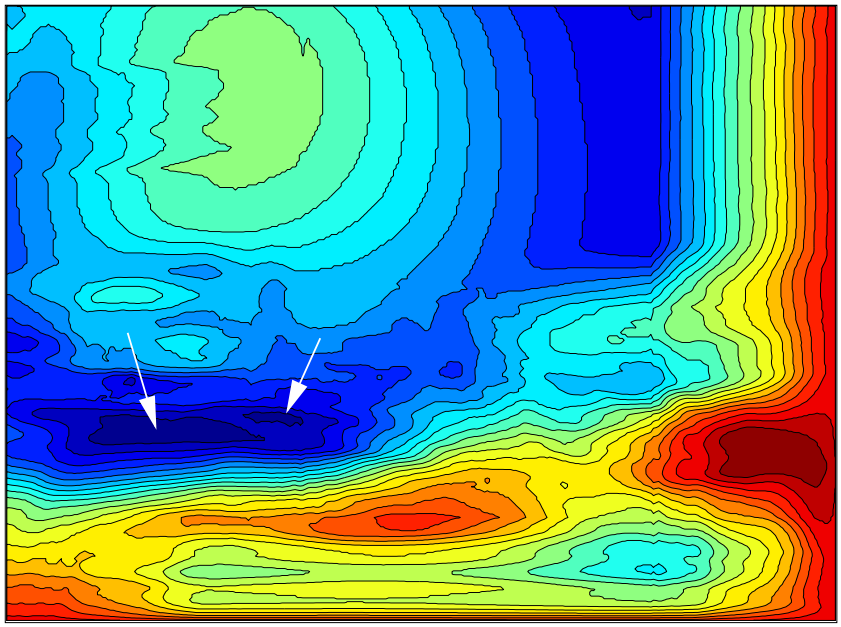


Figure 2.2: A contour plot of Frame 50 in the Coke-Put sequence showing the variety of minima. Notice the global minima (left) and a local minima (right) at nearly the same depth and location (indicated by the white arrows). These can occur since there is little vertical structure on the hand and its template can slip horizontally without changing the registration error much.

of the problem. The basic Least Squares(LSQ) estimator (see Eqn. 2.2) is both simple mathematically and simple to implement. However, the LSQ estimator has a problem. It will treat greater pixel intensity errors with a disproportionately larger significance (see Fig. A.1).

The robust estimator (see Eqn. 2.3) will reduce asymptotically the significance of larger values of pixel intensity differences. This could be used to reduce the effect of non-object pixels in the pose calculations if low and high pixel intensity differences could be consistently associated with object and non-object pixels. However, we use the LSQ estimator for its simplicity.

The LSQ estimator is:

$$\rho(z) = z^2 \tag{2.2}$$

The Robust estimator is:

$$\rho(z) = \frac{z^2}{\sigma^2 + z^2} \tag{2.3}$$

where z is the pixel intensity error, and σ is a constant that defines the x-axis spread of the function.

2.2 Template Alignment

To find the correct pose for the template we need an algorithm which will find minima in the surface defined by the error equation (Eqn. 2.1), where the global minimum corresponds to the correct registration of the template.

A naive approach to template tracking uses a global search method based on this error equation. However, given a good initial guess to the templates next position it is possible

to move the template to follow the object. The initial guess can prevent the alignment algorithm getting caught in a local minima.

The iterative registration procedure is implemented by first deriving a linear approximation to the image intensity at each pixel using the image gradient and a first degree Taylor's polynomial:

$$\begin{aligned} I(m(\vec{x};\vec{a}')) &= I(m(\vec{x};\vec{a}+\delta\vec{a})) && (2.4)\\ &= I(m(\vec{x};\vec{a})) + \nabla I(m(\vec{x};\vec{a})) \times \frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}}\delta\vec{a} + O(\|\delta\vec{a}\|^2) && (2.5) \end{aligned}$$

Where $\vec{a}' = \vec{a}+\delta\vec{a}$, $I(m(\vec{x};\vec{a}'))$ is the image intensity after one iteration of the algorithm, $I(m(\vec{x};\vec{a}))$ is the intensity of the pixel before the update to the initial pose, $\nabla I(m(\vec{x};\vec{a}))$ is the image gradient, $\frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}}$ is the derivative of the transformation with respect to the initial pose, $\vec{a}$, and $\delta\vec{a}$ is the iteration in the pose parameters that we are after.

This linearization is then substituted into the error equation (Eqn. 2.1) and the derivative with respect to the change in pose parameters, $\delta\vec{a}$, is equated to zero, converted to matrix form and solved for $\delta\vec{a}$ giving:

$$\delta\vec{a} = -(Q^T Q)^{-1} Q^T \times (I(m(\vec{x};\vec{a})) - J(\vec{x})) \quad (2.6)$$

Where

$$Q = \left(\nabla I(m(\vec{x};\vec{a})) \frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}} \right) \quad (2.7)$$

The solution $\delta\vec{a}$ is added to the current pose $\vec{a}$ and the procedure is repeated either a fixed number of times or until the change in pose $\delta\vec{a}$ falls below some threshold. Here we simply iterate a fixed number of times to avoid the threshold problems induced by the course to fine algorithm (see Section 2.5).

2.3 The Basic Tracker In Action

The basic tracker works very well when the number of background pixels is minimal and where no occlusions occur, or if they do occur the background is mostly uniform (see Figure 2.3). In the Box-Tip example (top) the box is tracked accurately. The hand also tracks accurately, even with a large number of background pixels, because the strength of the background texture is minimal. The bottom example shows that even with the occlusion of the hand, the pop can template can track reasonably well because the can is well textured and the background is contrasting.

2.4 The Error Surface

The error surface defined by equation 2.1 can be quite complex with many local minima in which the tracker can get stuck (see Fig. 2.2). This complexity comes, in part, from the numerous areas in the image that match subsections of the template view. A solution to this problem is the Course-to-Fine algorithm (see [20, 24]).

This algorithm's strategy is to obscure the local minima with a heavy blur of the image and template view which allows the iterative registration method to 'jump' these minima. As the pose estimate becomes more accurate the blurring can be reduced until no blur is used and the global minima is reached.

2.5 Course-to-Fine Algorithm

The Course-to-Fine algorithm is a simple method to smooth the image and template view (which smoothes the error surface) to prevent our iterative registration method from getting caught in a local minima.

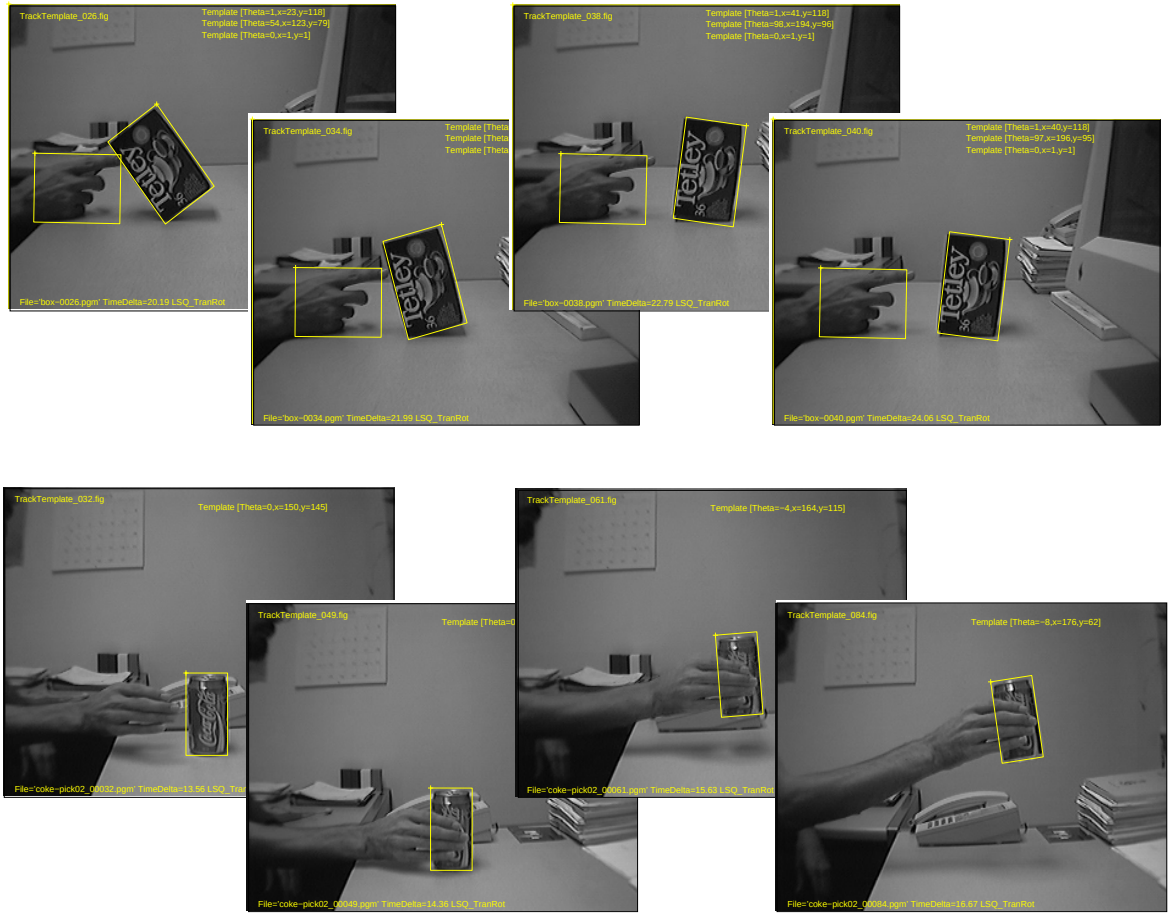


Figure 2.3: Examples of basic tracker in operation. Box-Tip example; top. Coke-Pick example; bottom.

We use a course to fine algorithm (see Fig. 2.4) that starts with a heavy blur on both the image and template and reduces it at successive stages until no blur exists. At each stage the registration technique finds a closer approximation to the global minima. If we used a pose threshold to stop the iterative registration process we could run into problems. A pose threshold that is too small during the large blur stage could cause the registration process to oscillate around the minima. One that was too large during the last blur stage may cause the registration to halt prematurely. To avoid these problems we just iterate a fixed number of times.

More about the Course-to-Fine method can be found in Lucas&Kanade's paper [20].

2.6 Summary

As we have seen in the introduction, the basic tracker does a good job but there are circumstances where it fails by various degrees. In the following chapters problems with the basic tracker will be described and examples shown. Three methods will be developed and incorporated into the basic template tracker to make it more robust in the face of these problems: Ownership Masks, Cooperative Templates and, based on these, Evolving Views.

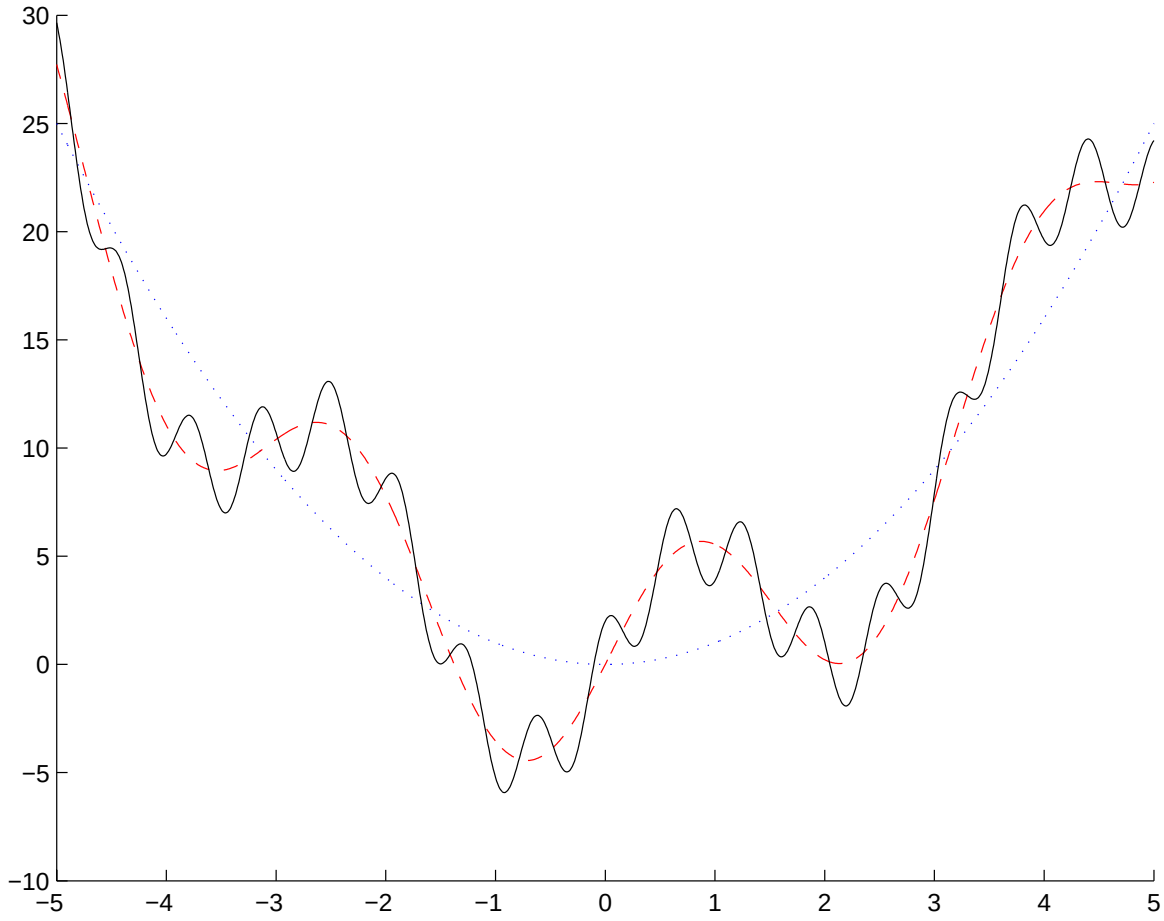


Figure 2.4: Course to Fine algorithm smoothes local variance to aid registration. In this synthetic example the image intensity variation (solid black line) is blurred heavily (dotted blue line) to allow the registration technique to find the minimum at zero. The blur is reduced (dashed red line) allowing the registration method to find a closer approximation (approx. -0.7) to the global minimum at -1.0. Finally, with no blur, the registration method will find the true global minima.

Chapter 3

Ownership Masks

As mentioned previously there are a number of sources of error in template tracking: extraneous pixels, occlusions, lighting/shadows and deformation being some of the more common. One of the most significant sources is the inclusion of extraneous or outlying pixels in the template region when registering the template. In this chapter we look at this source of error and how ownership masks can be used to reduce or eliminate it altogether.

3.1 Motivation

Extraneous, or outlying, pixels are the pixels within the template region which are associated with the background or other moving objects and not with the tracked object. In the comparison of the template view and the current image both the number of outlying pixels and the magnitude of the individual pixel intensity differences can produce a significant error in the pose calculations. Elimination of these outlying pixels from the tracking calculation should produce a noticeable improvement in the robustness of the template registration.

But how much do the outlying pixels bias the positioning of the template? Potentially outlying pixel error may have little influence in a real world tracking example [1]. However, in general the bounding polygon(bordering the template region) can expose a significant number of outlying pixels and these result in a noticeable amount of bias (see Fig 3.1).

Modifying the bounding polygon so that it covers only pixels belonging to the tracked object can be difficult when the object's shape is complex.

Figure 3.1 shows a case where the template incorporates too many outlier pixels in the bounding polygon due to the irregular boundaries of the tracked object (hand). Pixels from the background are exposed around the periphery of the hand and in the hole between the thumb and fingers (see Fig 1.2). A manually placed bounding box is placed in a more appropriate pose to indicate the magnitude of the error as the tracking progresses.

The bounding polygon around the hand includes about 20-30% of the template image as background pixels. The result is that the template tracker is trying to track both the background and the hand and thus tries to accommodate both velocities by averaging them. This effectively biases the tracking towards the background and away from the tracked object. In more severe cases, the tracker will appear to track the background and it will be the pixels associated with the tracked object that will bias the tracking.

The quantity of extraneous pixels affects the registration process but we also have to consider the pixel intensity differences between the template view and image. This tracking method uses the pixel's image intensity gradient and the pixel's intensity difference to estimate the change in position. The intensity difference and image gradient effectively weight the significance of the contribution of each pixel to the pose estimation. The estimated change in pose $\delta\vec{a}$ is directly proportional to the image intensity difference $(I(\vec{x}') - J(\vec{x}))$ at each pixel. Where the template view is the view of the object to be tracked and is

[1] When the tracked object is rectangular for instance.

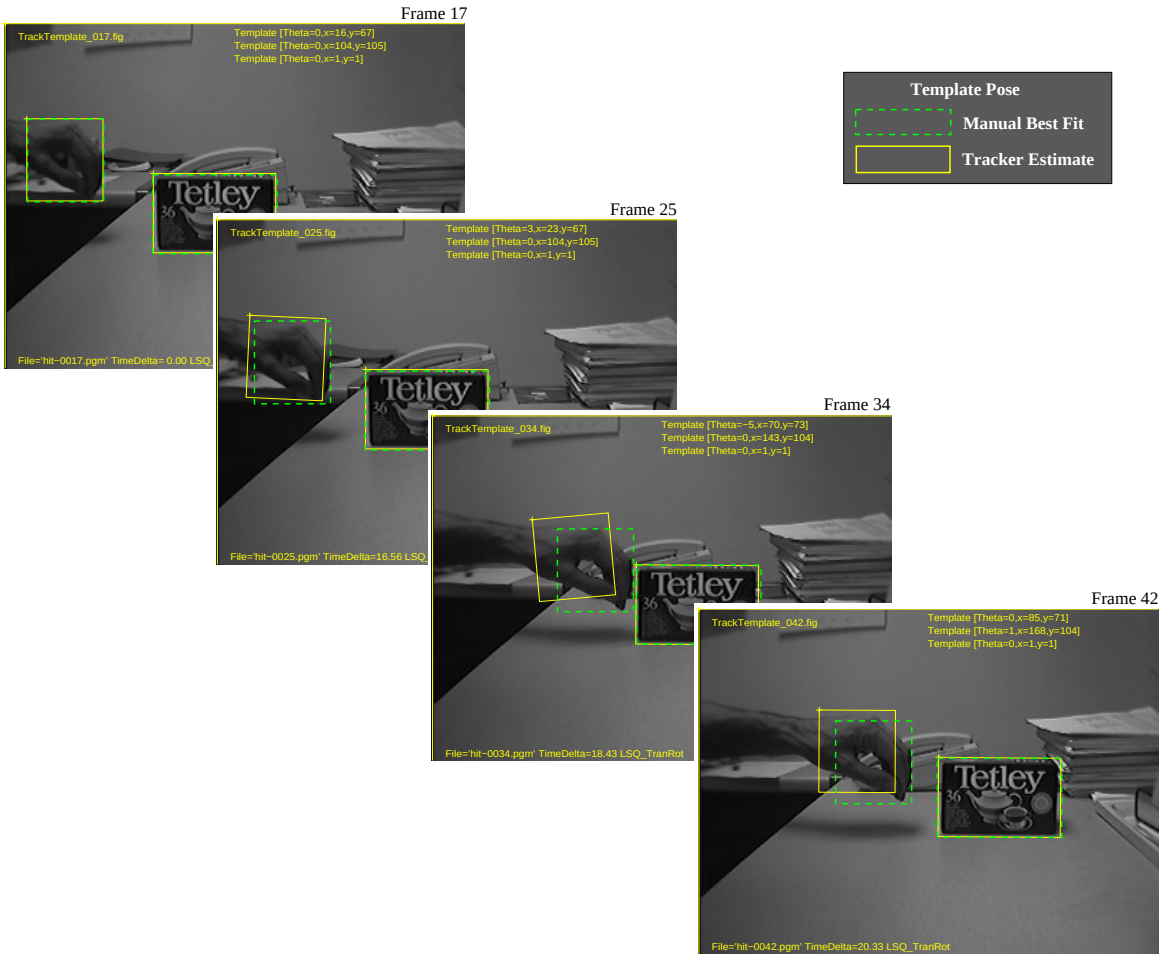


Figure 3.1: Example of template sliding off the tracked object due to the outlier pixels within the template's region. biasing the result. The solid (yellow) bounding box shows the location estimate by the basic tracker. The dashed (green) bounding box is a manually placed template that follows the tracked object's position. Because there are more background pixels within the template region of the hand the tracking of the hand, is biased more heavily.

often defined as a subsection of the initial image, $J(= I_0)$. In the template view the pixel intensities are defined as $J(m(\vec{x};\vec{a}))$ where $m(\vec{x};\vec{a})$ is the transformation of the template coordinates $\vec{x}$ from the template coordinate system into the image coordinate system using the transformational parameters $\vec{a}$ (see Fig 2.1).

The smaller the gradient for a pixel, the smaller the influence of that pixel on the calculation of the pseudo inverse and thus the estimated pose. When the gradient is zero there is no influence from that point at all. This is why tracking with a loose template on a uniform background is not a problem.

The effect of pixel intensity can be seen in Figure 3.2 where we have two different sequences. Both templates include roughly the same percentage of background pixels. The first sequence (a) shows a hand crossing in front of a 'noisy' background where there is a great deal of image variation information and thus large gradients. This noisy background produces a large intensity error for the pixels that are not part of the hand within the template region. The result is the template slips off the hand and is biased towards the stationary background. In the second sequence (b) the hand has a 'quiet', or uniform, background and there is little or no error in the positioning of the template. This lack of bias is due to the background pixels being roughly uniform across the sequence which minimises the gradient for these pixels and thus the influence of these pixels.

3.2 Spatially Weighted Templates

A rudimentary LSQ iterative registration tracker is used in conjunction with a course-to-fine iterative registration technique. This type of tracker normally operates fairly well in a well defined but limited environment.

For instance, template tracking works well when the tracked object does not vary in

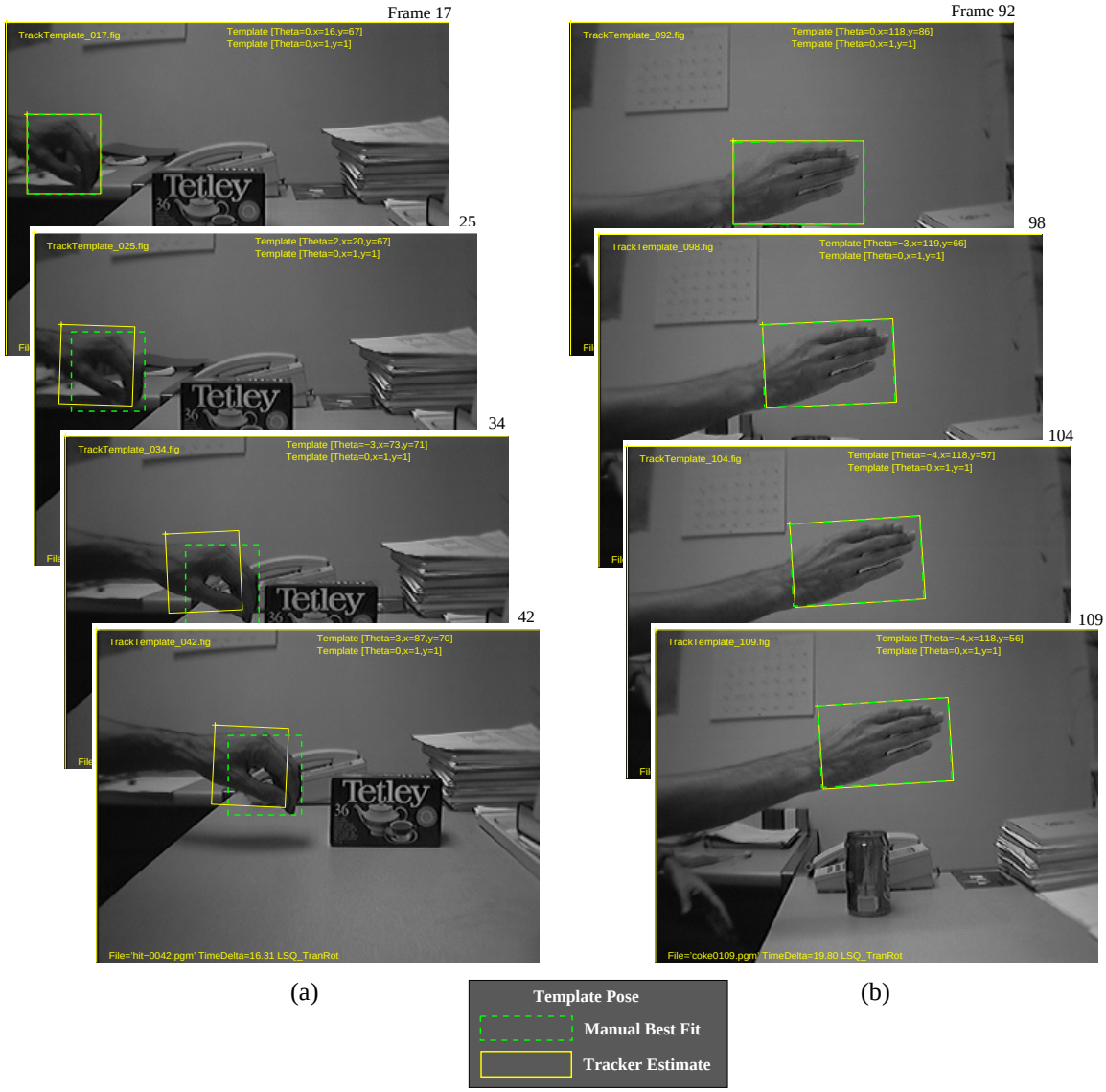


Figure 3.2: Comparing the tracking accuracy of two sequences: one with a 'noisy' and the other with a 'quiet' background. Running the basic template tracker on these two sequences shows the effect of the extraneous-pixel error-magnitude on the bias of the tracker. When the background is textured the pixel error-magnitude is large (a) and the resulting bias is large. Similarly when the background is uniform (b) the error is minimal as is the bias.

appearance other than the normal rigid affine transformations. Difficulties arise for this type of tracker when the object is occluded either partially or fully and when the template region includes a sizeable portion of non-template, or extraneous, pixels.

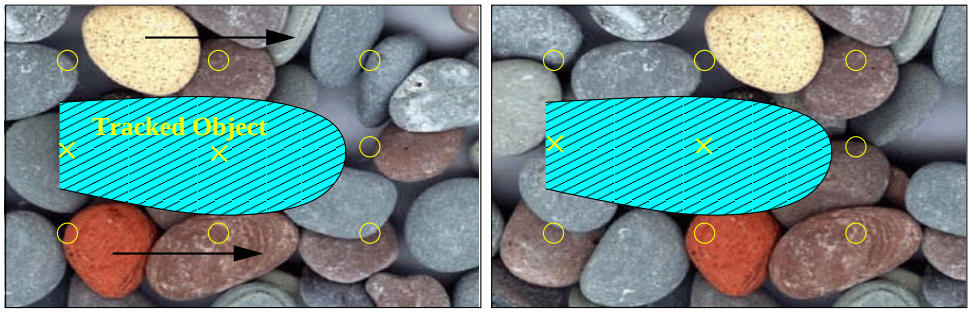
Spatially weighted templates or *ownership masks* are proposed as an enhancement for this simple template based tracking method. The template will weight each pixel within the template region with a measure of the certainty that the pixel belongs to the tracked object.

To develop these masks we need to deal with a number of problems. The first problem that presents itself is that we need some method to distinguish object from non-object pixels. Assuming perfect tracking for a moment, it seems reasonable to assume that the object pixels will vary far less than the transient pixels in the background as the object is tracked. If we start with a sufficiently tight variance measure based on the image from which the tempate view was derived, the transient pixels in the background should appear with very low probability as the template pixels cross contrasting parts of the image. For instance, if our initial background is uniformly black our initial template view will show black in the background areas. As the template follows the tracked object to an area with a white background the difference in pixel intensities is large and the background pixels in the template should show a low probability. This low probability can then be used to isolate these pixels from the registration calculations thereby reducing the tracking bias.

This, however, assumes perfect tracking. As the positional error of our tracker varies, the number of pixels that are eliminated from the mask by their intensity variance will increase. If the tracking becomes poor enough this effect will produce an ill-conditioned tracker. To circumvent this we will accumulate evidence as we track. Thus a transient effect will have little impact on the registration process.

When a template travels over a uniform background the extraneous pixels will have

a high probability of being misinterpreted as part of the tracked object. In Figure 3.3 the background of the tracked blob object (light blue) should give us evidence that the background pixels are not part of the object of interest.



Frame t Frame $t + \delta$

Figure 3.3: Changes in the intensity of pixels in the background of the image indicate the ownership of these pixels is not by the template. Here the background details transit past the tracked object. The circles represent the pixels that change intensity and the crosses those that do not.

To develop this mechanism we need to derive pixel statistics from a key image. We assume that any noise in the pixel intensity will come from small (e.g. a single pixel width) error in position of the template or camera. The camera is assumed to have a static pose.

3.2.1 Pixel Statistics

To model the ownership of template pixels, a pixel noise model was needed. A Gaussian model of noise was chosen as a good approximation to the discrete distribution that represents a combination of the camera and real-world noise in the image. It does have its limitations when the variance measure becomes too small or when the mean value of the distribution is near to the extremes of the intensity range $[0 \ldots 255]$. However, the

Gaussian model is simple mathematically and well understood.

Previously all pixles were given an identical variance measure but this was found inappropriate as simple camera shake and sharp discontinuities in the image intensity (such as on the edge of the stack of papers) caused spurious ownership indications.

A single pixel error in positioning was assumed (and detected with a background subtraction test) so the pixel variance was based on the eight surrounding pixels (the pixel neighbourhood). This will define a mean and variance for each template pixel.

These pixel statistics provide a per pixel mean $\mu(\vec{x})^t = \frac{1}{9}\sum_{i=0}^{8} I_i^t(\vec{x})$ and variance $\sigma(\vec{x})^t = \frac{1}{8}\sum_{i=0}^{8}(I_i^t(\vec{x}) - \mu(\vec{x})^t)^2$ where $I_i^t(\vec{x})$ is defined by the neighbourhood around the central pixel (see Figure 3.4).

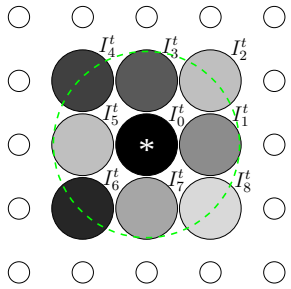


Figure 3.4: The statistics of the 3×3 *neighbourhood of pixels around the central pixel* I_0^t *defines the mean and variance. The green circle represents the area a pixel sweeps out as its position varies by one pixel width.*

3.2.2 Ownership Probability

The Ownership Probability is derived using a Gaussian model (Eqn B.1) of the pixel noise along with the variance and mean measures derived for each pixel in the previous section.

If we assume that the pixel intensity of the pixels on the object will stay constant during

the transit of the object through the image sequence and that the template is properly positioned, the pixels in the template corresponding to the object should have zero intensity error. That is, the pixels of the tracked object match from template view to image frame for the entire sequence. This is normally an approximation because lighting variations and shadow effects introduce error. Any interpolation method will also induce a process error into the system.

The pixels that are not on the tracked object, however, should vary a great deal in intensity due to the changing background [2]. Applying a Gaussian noise model to the pixel intensity we derive a measure of the probability that a image pixel matches the corresponding template view pixel:

$$P(I_i(\vec{x})) = \frac{1}{\sigma_i(\vec{x})\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{I_i(\vec{x})-\mu_i(\vec{x})}{\sigma_i(\vec{x})})^2} \tag{3.1}$$

Where:

$I_i(\vec{x})$	is the Image pixel intensity corresponding to the point $\vec{x}$ in the template T_i ,
$\sigma_i(\vec{x})$	is the estimated pixel intensity variance at this pixel position,
$\mu_i(\vec{x})$	is the estimated pixel mean intensity at this pixel position, and
$P(I_i(\vec{x}))$	is the probability of intensity $I_i(\vec{x})$ at this pixel position.

3.2.3 Accumulation of Evidence

We accumulate the ownership probability over time in a template array called the cumulative ownership using Eqn. 3.2. This is done to solve the problem of transient effects in

[2]This assumes that the object is moving against a background that has some texture; even a slight gradient will suffice

our image sequence [3] Cumulative Ownership $\hat{O}(\vec{x})$ is a temporally averaged function of the ownership probability $O(\vec{x}) = P(I_i(\vec{x}))$.

$$\hat{O}(\vec{x})^t = \lambda \hat{O}(\vec{x})^{t-1} + (1 - \lambda) O(\vec{x})^t \tag{3.2}$$

Where:

$\hat{O}(\vec{x})$	is the Cumulative Ownership,
$O(\vec{x})$	is the estimated instantaneous ownership,
λ	is the migration constant,
t	is the current frame or time step, and
$\vec{x}$	is the pixel position in template T.

Experiments show that this eliminates some of the error due to occasional matches for the background pixels in the template region. It improves the accuracy of the template positioning problem.

3.2.4 The Mask

Now that we have a measure of the probability that the intensities of the pixels match between the image and the template we can define a mask. The intent of this mask is to eliminate non-object pixels from the iterative registration process.

Our first attempt to derive a mask is to use a rudimentary threshold for all pixels. The idea is to segment the pixel errors into two groups: inliers and outliers by thresholding the ownership. Inliers are those pixels that have a high probability of being the same pixel from template to image, outliers are those pixels that have a low probability.

[3] Transient effects such as minor shadowing.

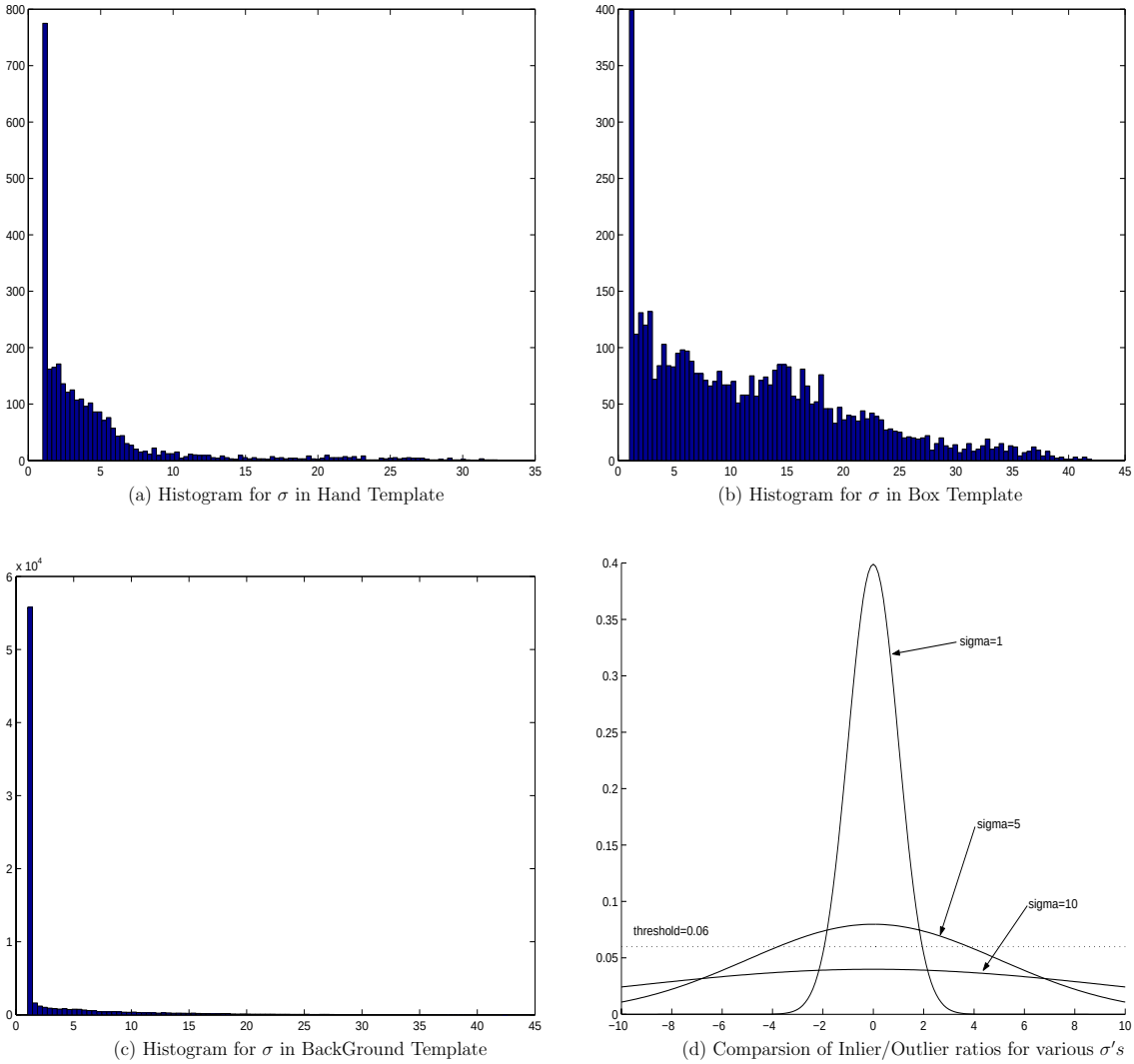
The mask is defined as $M(\vec{x}) = max(0, \hat{O}(\vec{x}) - \tau)$ where τ is an arbitrary threshold. The mask is then normalized to the $[0 \ldots 1]$ range to provide a standard range of values for the iterative registration process.

This method has a few problems. First, not all pixels in the template have the same variance. These variances were determined for each pixel and depended on the neighbourhood of the pixel in the key frame from which the template view was derived.

Furthermore, as the histograms of the pixel standard deviation in Figure 3.5(a,b and c) show, the distribution of pixel standard deviations varies from template to template. The problem with this simple threshold method is even more evident in Figure 3.5(d) where the $N(0, \sigma)$ distributions are plotted for $\sigma = 1, 5$ and 10 with the threshold(=0.06) displayed. Here the tallest distribution ($\sigma = 1$) is divided ($x \in [-1.95, 1.95]$) by the threshold into 95% inliers and 5% outliers. In the $\sigma = 5$ distribution the threshold divides ($x \in [-3.77, 3.77]$) the pixel errors into 55% inliers and 45% outliers. When $\sigma = 10$ all points are outliers. The maximum σ that will allow any inliers (where the peak of the $N(0, \sigma)$ distribution touches the threshold) is $\sigma = 6.65$.

Thus the threshold has eliminated more pixels in the highly textured areas of the template than those in the less textured areas. Since tracking of higher texture areas should lead to more accurate tracking this seems counterproductive. Clearly this is not the best way to exclude the outliers but it did have some success and showed that eliminating the non-object pixels could improve the accuracy of the tracking.

The problems alluded to above will be remedied to a degree in the next chapter.



(a) Histogram for σ in Hand Template

(b) Histogram for σ in Box Template

(c) Histogram for σ in BackGround Template

(d) Comparsion of Inlier/Outlier ratios for various $\sigma's$

Figure 3.5: A common threshold does not work well when the distributions being thresholded have different parameters. In (a,b,c) the difference in distribution of pixel standard deviations between templates is displayed. The problem with using a rudimentary threshold to segment pixel errors into inliers and outliers is suggested in part (d) (see text for further details).

3.3 Results of using Ownership Masks

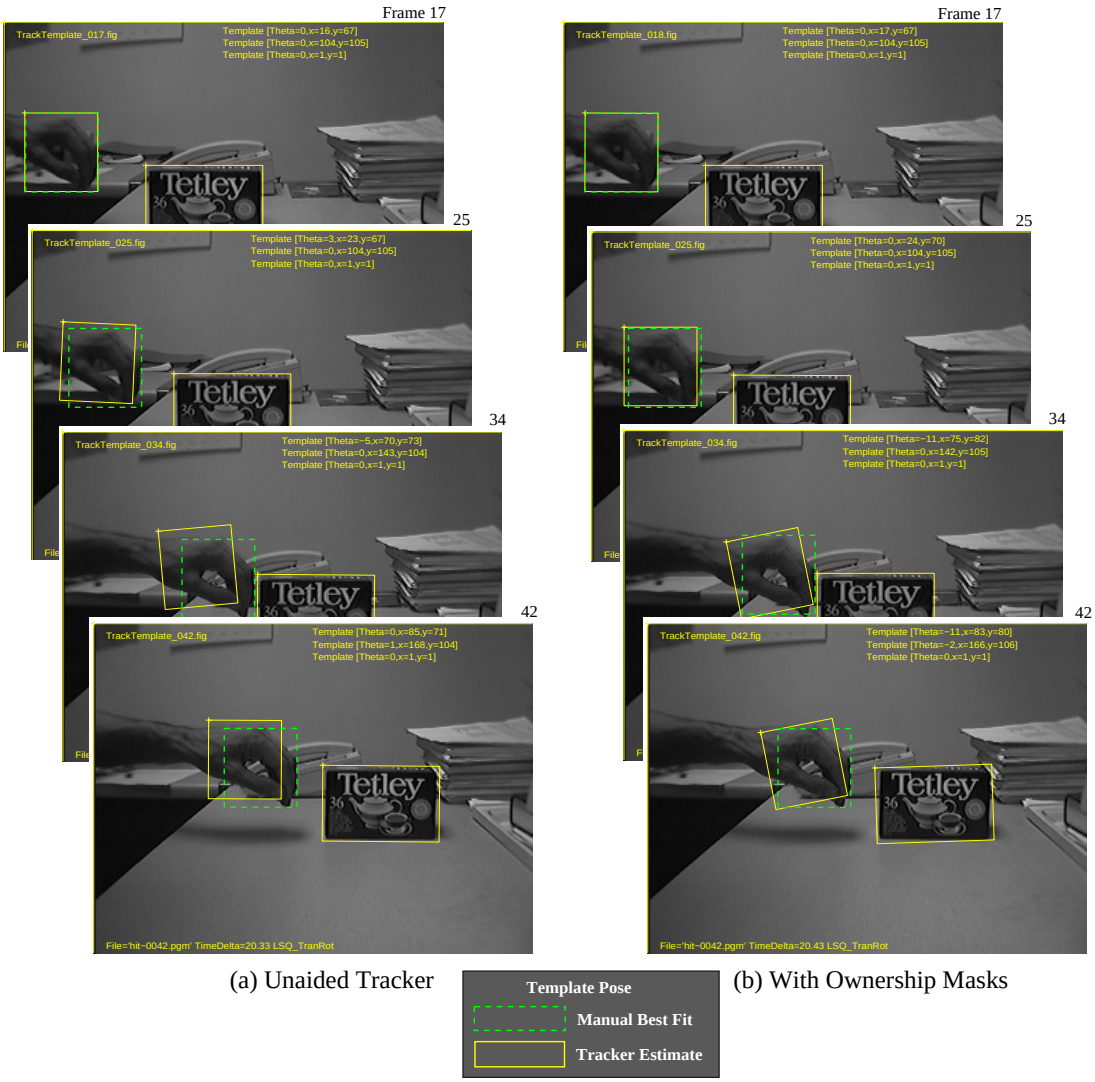
If we look at the image sequence in Figure 3.6 we see that the tracking has improved over the unaided version of the tracker. Some error still remains in the positioning of the template. Most of this error, however, comes from the time it takes to initialise the ownership mask and the reduction in the number of tracked pixels. Fewer pixels are tracked because the threshold used in creating the mask has eliminated a large fraction of the pixels from the mask and thus from the registration process.

Initially the tracker behaves as it does in the basic tracker. After the tracker has run for a while the ownership masks develop and the tracking bias settles to constant value.

The video sequence starts with the original template and a dashed rectangle manually placed after tracking[4]. As the video sequence progresses, the template wobbles slightly as the ownership mask evolves. Somewhere between frame 25 and 34 the template has become steady and thereafter it does not wobble significantly. This is due to the group of pixels in the ownership mask being reduced to a subset which all have small variance measure. In the unaided experiment the template bias gets larger and continues to be unstable.

Thus we see that eliminating these outlying pixels by the method of Ownership Masks has reduced the positioning error in the template tracker. However, this accuracy is dependent on this arbitrary threshold and may degrade over the duration of the tracking. The corresponding ownership masks for three values of threshold are shown in figures 3.7 (Threshold=0.00), 3.8 (Threshold=0.06) and 3.9 (Threshold=0.10). In Fig. 3.7 the hand becomes distinguished in mask (bottom insets) but rudimentary thresholding causes 'masking out' of important hand details allowing the background pixels to bias the tracker. In Fig. 3.8 the harsher thresholding eliminates the background pixels that caused the bias in previous experiment but also 'masks out' more of the pixels that allow accurate tracking

[4]The dashed rectangle represents the accurate position of an ideal tracker.



(a) Unaided Tracker (b) With Ownership Masks

Figure 3.6: Ownership masks appear to have improved tracking accuracy. In the first sequence (a) the template rapidly falls behind the tracked hand. This is corrected for in the second sequence (b) where the Ownership Mask keeps the template closer to the correct position. The small error is due to the delay in forming the ownership mask for the object being tracked and the small set of pixels remaining in the mask. This error accumulates in the first few frames but thereafter is fairly constant.

of the hand. Masking out the pixels in areas of high texture allows the template to rotate on the hand. In Fig. 3.9 the very heavy blur has eliminated most of the pixels on the hand. Pixels in areas of strong texture are eliminated first in this rudimentary thresholding scheme. This allows the hand template to slip completely off the hand.

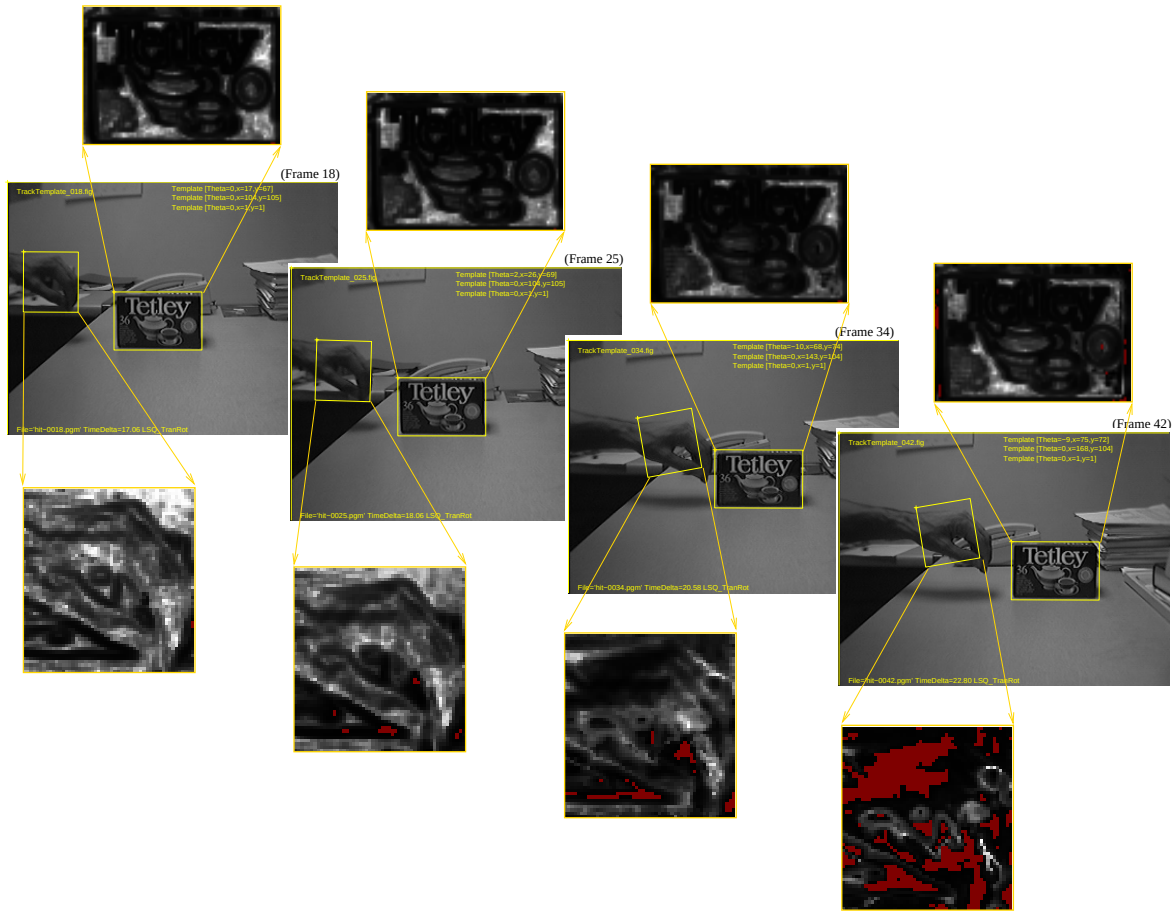


Figure 3.7: Threshold=0.00. Evolution of hand and box ownerships. The hand becomes distinguished in its mask (bottom insets) but rudimentary thresholding has not 'masked out' the background pixels allowing these pixels to bias the tracker just as in the basic tracker. Note the red pixels in the mask show pixels completely eliminated by the mask from the iterative registration process.

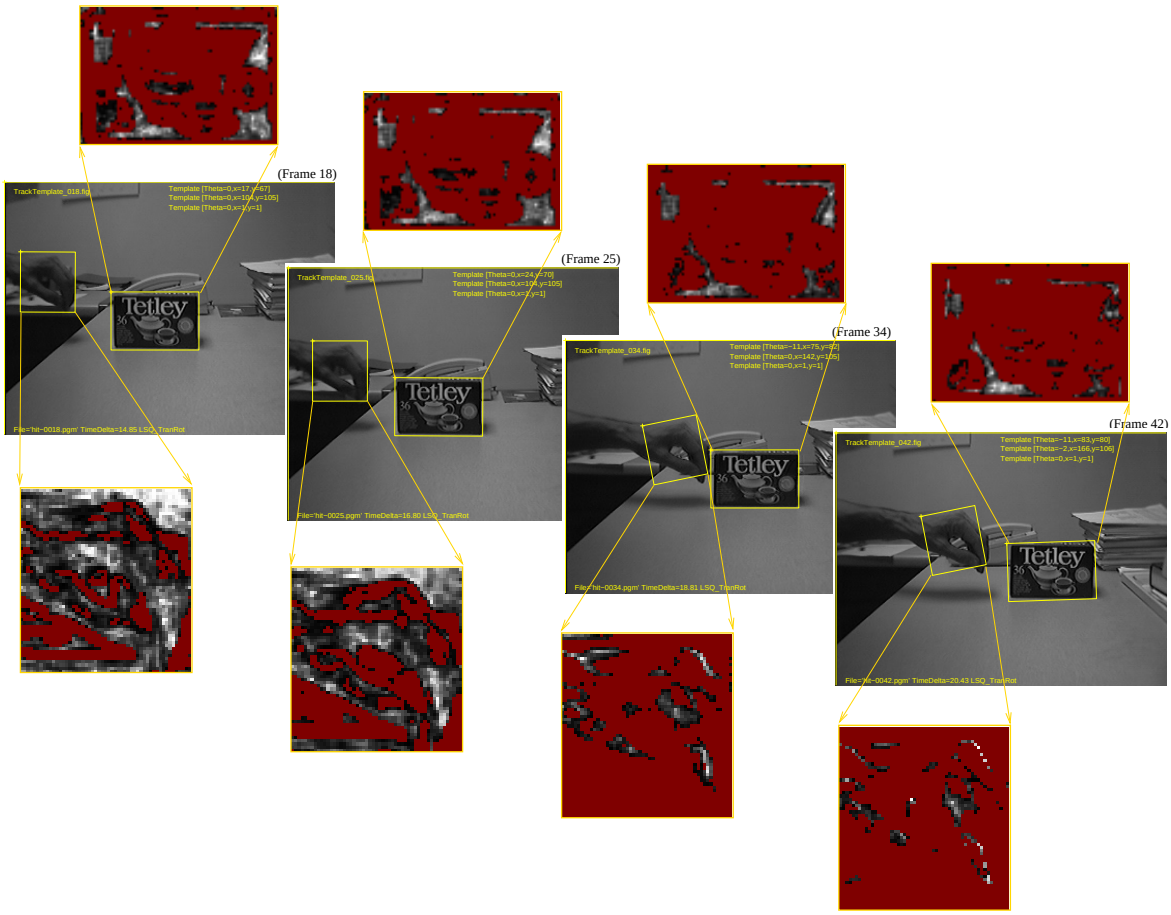


Figure 3.8: Threshold=0.06. Evolution of hand and box ownerships. Harsher thresholding eliminates the background pixels that caused the bias in previous experiment but also 'masks out' more of the pixels that allow accurate tracking of the hand. Masking out the pixels in areas of high texture allows the template to rotate on the hand. Note the red pixels in the mask show pixels completely eliminated by the mask from the iterative registration process.

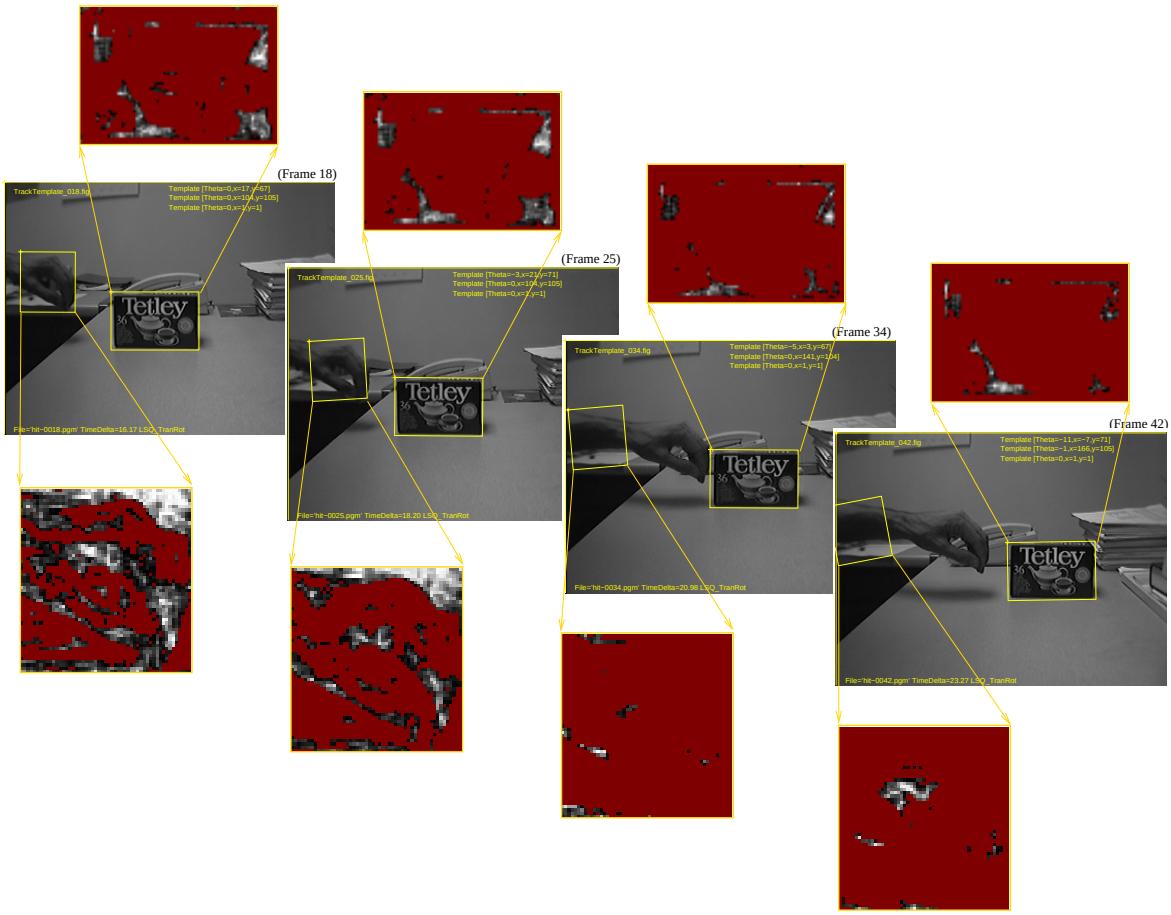


Figure 3.9: Threshold=0.10. Evolution of hand and box ownerships. Very heavy blur has eliminated most of the pixels on the hand. Pixels in areas of strong texture are eliminated first in this rudimentary thresholding scheme. This allows the hand template to slip completely off the hand. Note the red pixels in the mask show pixels completely eliminated by the mask from the iterative registration process.

Chapter 4

Cooperative Templates

In the previous chapter Ownership Masks were developed to deal with the problem of background or extraneous pixels within the boundary of the template. Ownership masks helped solve the extraneous pixel problem. However, there are problems with Ownership Masks: templates may overlap in the image. When this happens the previously defined ownership probability is no longer correct.

If more than one template covers an image pixel then that pixel may come from either template. The ownership probability will be a measure of how accurately the template's pixel intensity matches the image's pixel intensity at that point. If the two intensities match (no error) for each template covering the pixel the probability of ownership, for that pixel, will be one for each template. This is not correct.

4.1 Shared Probability

In figure 4.1 there are four templates: the background and templates one through three T_{BG}, T_1, T_2, and T_3. If we assume all templates views and the image are of the same uniform

intensity then the probabilities should be distributed to each of the covering templates as shown. If two templates cover a pixel then they each get $\frac{1}{2}$ of the probability and a similar distribution occurs for other combinations of templates.

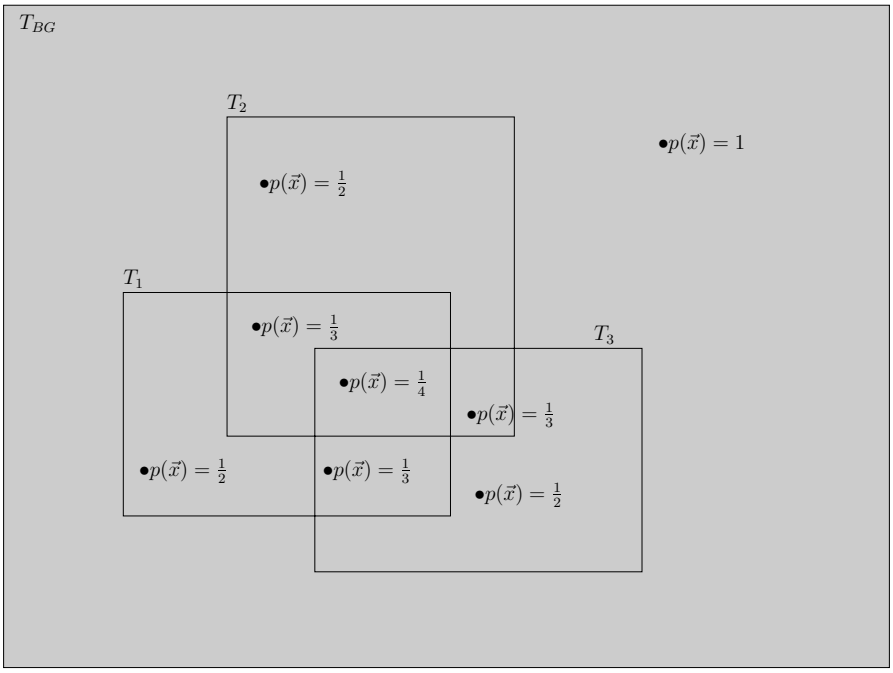


Figure 4.1: Cooperating templates are designed to share probability between templates. When more than one template covers a pixel position that pixel may match closely the corresponding pixel in more than one of the covering templates. Here the template views and image are all of the same uniform intensity so the four templates share the ownership probabilities as shown.

Cooperative templates are just multiple templates that share the probability mass for each of the pixels that they and other templates cover.

In the Coke-Put video sequence the hand is initially holding the pop can and is thus occluding the middle and bottom of the can during the first half of the video. Our technique

of using a key-frame to initialise the template views for the templates has run into a problem: the templates overlap. We could go back and manually initialise the template view from a frame where the full template was visible but that may not always be possible. Furthermore, since our goal is to have a fully automated template tracker, we would like them to auto-initialise.

In an *occlusion* event the pixel intensity error typically increases for the pixels that are being covered by the occluding object because they do not match in intensity. Typically in an *unocclusion* event the intensity error significantly decreases as an occluding object uncovers the tracked object. This is due to the image intensities being similar on the template view to those in the newly exposed area in the image. This assumes we know the tracked object's full view. When we do not have this information, as in the above example (Coke-Put), the pixel error will increase as in the occlusion event. The reason for this is that the template view has been initialised with the image of the occluding object and the true view of the object appears as an occlusion. Thus we have no way of knowing whether the increase of intensity error is from an occlusion event or an unocclusion event.

Cooperative templates help us in this situation. When we have multiple templates in an image, part of the task of tracking is being able to segment the tracked object pixels from the background and other moving, possibly occluding, objects.

With a background template and one or more moving templates we have the situation where one or more templates are always covering all the pixels in the image. These are called covering templates. What we would like to do is to segment this group of pixels between the covering templates so that each pixel is assigned to the template that is tracking the object from which the pixel came.

If two templates, A and B, cover an area of pixels the probability that a pixel in this area belongs to template A is not independent of the probability that it belongs to template

B. The ownership probabilities, based on the pixel intensity error, we developed in the Ownership Mask chapter are actually conditional probabilities dependent on the condition that we know to which template the pixel belongs.

Using Bayes' rule we can combine these conditional probabilities and the prior probabilities of the templates into a segmentation of the pixels.

4.2 Probabilistic Formulation

When tracking is in progress an image pixel can be assigned to one of N active templates $T_1, T_2, \ldots, T_N$ (which includes the background template). The conditional probability of the image pixel intensity value given that it came from a particular template T_i is $P(I(\vec{x})|T_i)$. Where this is the ownership probability $P(I(\vec{x}))$ from the previous chapter.

4.2.1 Conditional Probabilities

When one or more templates covers a pixel $\vec{x}$, we need to know the prior probability $P(T_i(\vec{x}))$ that the template is the visible template. Given that the number of covering templates was defined to be $N(\vec{x})$, $\sum_{i=1}^{N(\vec{x})} P(T_i(\vec{x})) = 1$ and the probability should be distributed uniformly between the priors, we chose the proir $P(T_i(\vec{x})) = \frac{1}{N(\vec{x})}$.

Now using Bayes' rule to produce the conditional probability that the pixel came from template T_i given its intensity is $I(\vec{x})$:

$$P(T_i|I(\vec{x})) = \frac{P(I(\vec{x})|T_i)P(T_i)}{\sum_{j=1}^{N} P(I(\vec{x})|T_j)P(T_j)} \tag{4.1}$$

In the previous chapter the ownership probability, $O(\vec{x})$, was set to the pixel intensity probability $P(I_i(\vec{x}))$. Now we can use the more accurate conditional probability $P(T_i|I(\vec{x}))$

as the ownership probability $O(\vec{x}) = P(T_i|I(\vec{x}))$.

4.2.2 New Ownership Mask

Referring back to Fig. 4.1, if the posterior probabilities are equal then the ownership mask threshold should be inversely proportional to the number of covering templates. This is the division point where the probabilities $P(I(\vec{x})|T_i)$, that the each template caused the given image pixel intensity $I(\vec{x})$, are equal. Greater probability for a templates pixel suggests that this template is the more likely owner and lesser probability suggests the reverse. This is the new ownership threshold $\lambda(\vec{x})$:

$$\lambda(\vec{x}) = \frac{1}{N(\vec{x})} \tag{4.2}$$

Where, $N(\vec{x})$ is the number of covering templates at pixel $\vec{x}$. This is the threshold used to mask the ownership probabilities when creating the ownership mask.

If the ownership for any pixel in a template exceeds its Ownership Threshold then this is evidence that the pixel should belong to that template [1]. Taking the group of pixels in a template whose ownership value exceeds the threshold therefore suggests the extent of the tracked object within the template.

If the ownership is lower than the threshold then probability has moved away from that template and will be higher on one or more of the other templates covering the pixel.

Thus the ownership, $O(\vec{x})$, minus the ownership threshold, $\lambda(\vec{x})$, is a measure of how well the template pixels match those of the corresponding template view. If we have a

[1] Remember that the threshold is now defined to be that point at which we have no information about the segmentation. Given $P(T_i) = \frac{1}{N(\vec{x})}$ and $P(I(\vec{x})|T_i) = P(I(\vec{x})|T_j) \forall i, j \in [1 \dots N]$ the conditional probabilities are $P(T_i|I(\vec{x})) = \lambda(\vec{x})$.

positive difference then we use that difference as the raw [2] ownership mask, $R(\vec{x})$ which we stretch back to the range $[0 \ldots 1]$ in the ownership mask $M(\vec{x})$:

$$R(\vec{x}) = \begin{cases} O(\vec{x}) - \lambda(\vec{x}) & \text{if } O(\vec{x}) - \lambda(\vec{x}) > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \forall \vec{x} \in T_i \tag{4.3}$$

$$M(\vec{x}) = \frac{(R(\vec{x}) - min_{x \in T}(R(\vec{x})))}{(max_{x \in T}(R(\vec{x})) - min_{x \in T}(R(\vec{x})))} \tag{4.4}$$

Consider the example of two moving templates and a background template all covering the same pixel x (see Fig 4.2) where all three template views are initialised from (a). Evidence of the ownership of a pixel $\vec{x}$ by a template T_i is an ownership value greater than $\frac{1}{3}$ (up to unity). Initially the prior probabilities are all equal to $\frac{1}{3}$ and the conditional probabilities $P(I(\vec{x})|T_i)$ are all one because the template pixel intensities were set to this image's intensities. Therefore, the posterior probabilities $P(T_i|I(\vec{x}))$ are all equal to $\frac{1}{3}$.

As the tracked objects move we can expect the conditional probabilities to change. In our example the background template, T_{BG}, has become lighter at pixel $\vec{x}$. The foreground template T_2 has turned darker at pixel $\vec{x}$. Thus their conditional probabilities $P(I(\vec{x})|T_{BG})$ and $P(I(\vec{x})|T_{T_2})$ have both decreased. However, $P(I(\vec{x})|T_{T_1})$ has stayed the same because the intensity error is still zero. Thus the conditional probability $P(T_1|I(\vec{x}))$ has gone up and $P(T_{BG}|I(\vec{x})) + P(T_{T_1}|I(\vec{x})) + P(T_{T_2}|I(\vec{x})) = 1$

Similarly if the two moving templates' conditional probabilities, $P(I(\vec{x})|T_{T_1})$ and $P(I(\vec{x})|T_{T_2})$, drop for this pixel and the background probability $P(I(\vec{x})|T_{BG})$ stays high then the posterior probability moves to the background and so does the ownership.

[2] Before scaling to $[0 \ldots 1]$.

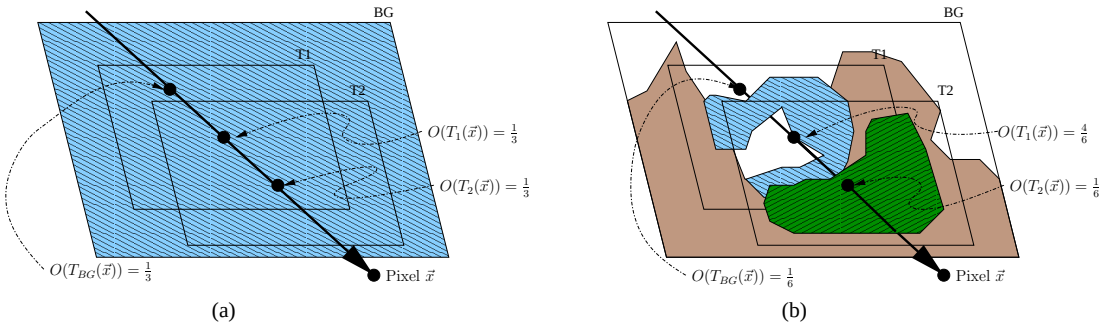


Figure 4.2: Progression of ownership probabilities. Initially (a), before motion evidence, pixel ownership is evenly distributed between templates. Later (b) as motion clues help to distinguish pixels in one template from those in the background and other templates, the ownership probabilities change to reflect this.

4.2.3 Extracting the Probabilities

For each template pixel we must accumulate the probabilities from all the other templates which cover that pixel. Since the pixel positions do not necessarily correspond from one template to another, determining which templates cover the current template's pixels can be involved. This could be done by interpolation in the image coordinate frame but that brings up the problem of determining which template pixels are covered by the other templates. Converting the current template to each of the other templates' coordinate system allows us to do simple inequality tests to determine this.

In Fig 4.3 the method used is diagrammed. Assuming we want to accumulate the probabilities for template T_1, we must first convert the pixel coordinates from the T_1 coordinate system (a) to the image coordinate system (b) with the forward transformation $m(\vec{x}; a_{T_1}^{\rightarrow})$. Then we have to apply the reverse T_2 transformation $m(\vec{x}; a_{T_2}^{\rightarrow})$ to transform those coordinates into the T_2 coordinate system (c). The T_1 pixels that cover T_2 are then

determined with simple inequalities $T_2x, min \leq x_1 \leq T_2x, max$ and $T_2y, min \leq x_2 \leq T_2y, max$. If we had chosen instead to work only in the image frame coordinates we would have had to test each templates' pixels against all the other template edges to determine which of these fall within the boundaries of the template.

The probabilities for each T_1 pixel $\vec{x}$ that are covered by T_2 can then be easily determined using bilinear interpolation. Once this is done for all covering templates, $T_2, \ldots, T_N$ we have accumulated the total probability $\sum_{i=1}^{N} P(I(\vec{x}|T_i))P(T_i)$.

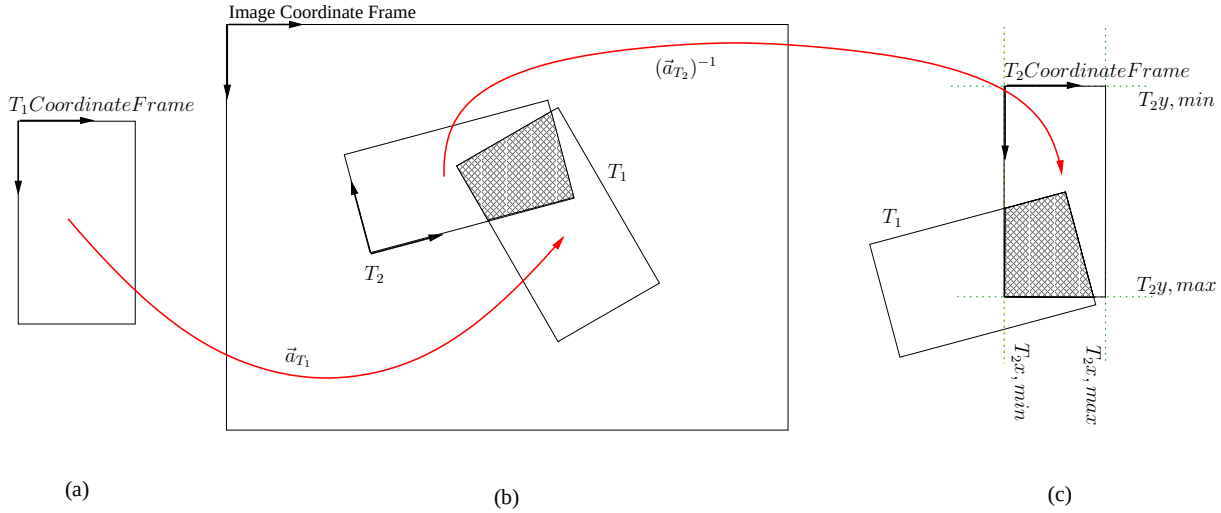


Figure 4.3: Cooperative template transformations. Original template T_1 with T_1 coordinate system (a). Template T_1 in Image coordinate system (b). Template T_1 in template T_2's coordinate system (c) where T_2's extent defines the points to be interpolated (cross-hatched).

4.3 Summary

4.3.1 Distinguishing Occlusions from Unocclusions

With cooperating templates we can now use ownership probabilities to distinguish occlusions and unocclusions. As was said before, an *occlusion* will cause a reduction in the previous version of the ownership probability $P(I(\vec{x})|T)$. However, in this version of ownership probability this is exactly what the *unocclusion* did when the template view was not known.

When it comes to distinguishing the type of occlusions and unocclusions we have a new tool that did not exist in the previous chapter. The ownership probabilities are dependent on the number of covering templates as well as the individual intensity probabilities $P(I(\vec{x})|T_i)$. When the number of templates covering a pixel decreases, the ownership probabilities on the remaining templates can go up even if the intensity probabilities stay the same. This new mechanism can work to our advantage.

In the case where we have two templates that have had their template views initialised from the same key image, the ownership probability will be shared evenly until unocclusion. If the upper template then moves off, the ownership probability on the unoccluded template will go up or down depending on how well the occluded template view matches the newly exposed image. However, since there is only one covering template now the ownership probability $O(\vec{x})$ $(= P(T_i|I(\vec{x})))$ will go to one; the template pixel must be owned by the only covering template.

When there are three covering templates we have a different situation. If all templates have been initialised with the same view we have no way of knowing which template is the owner. Therefore, they all have the same ownership probability until one of the templates moves away from the others. If the upper-most template moves away the ownership

probability will be distributed between fewer templates. Since they have equal intensity probabilities then their ownership probabilities will go up and the unocclusion has been detected in both templates.

Although we have not been able to prove mathematically that the ownership probabilities will go up consistently in the case that one of the covering templates has a known view we do have experimental evidence that this happens in at least some situations.

The evidence that cooperative templates help in occlusion discrimination comes in Fig. 4.4. Here the hand has released the pop can and has started moving up occluding the top. The hand is also unoccluding portions of the bottom of the pop can. Originally (a) this would produce a lowering of the ownership probabilities due to the mismatch between the template view, which includes the occluding hand, and the true view of the object in the image. In (b) the cooperative template method has changed the low (dark grey) probability to a much higher probability (white).

4.3.2 Improvement in Ownership Masks

As we saw in the last chapter, division of pixels into groups of inliers and outliers by using a single threshold for all template pixels was problematic. The threshold was used on the ownership probabilities which were simply the probabilities that the pixel intensities matched from one template to the image. Thus we had no notion of a model for either the background or other moving objects. In this chapter we have addressed these problems partially by adding background and other object models and by thresholding pixels differently.

The other models used now allow us to determine the probability that an image pixel came from one of the templates by comparing how well the image pixel intensity matched the intensity of the template pixel in each of the templates. Thus instead of having a high

probability of matching when the template and image pixels were close in intensity they now have high probability if that template's pixel matched the image's pixel better than all the other covering templates.

The arbitrary threshold of the last chapter has been replaced by a per-pixel threshold which is inversely proportional to the number of covering templates $N(\vec{x})$. Thus if each template's pixel intensity probabilities $P(I(\vec{x})|T_i)$ are equal then the probabilities that each of the templates is the owner of that pixel are all equal to $\frac{1}{N(\vec{x})}$.

The results of these two changes can be seen in the improvement in the tracking in Figure 4.5 and in the corresponding ownership masks in Figure 4.6. The tracking is more accurate and does not have the angular bias seen in the previous chapter. As the ownership masks show the weighting on the hand pixels is now quite high(white) and those of the background pixels are quite low(black). Thus the registration process is using the information from the tracked object's pixels and ignoring the background pixel information.

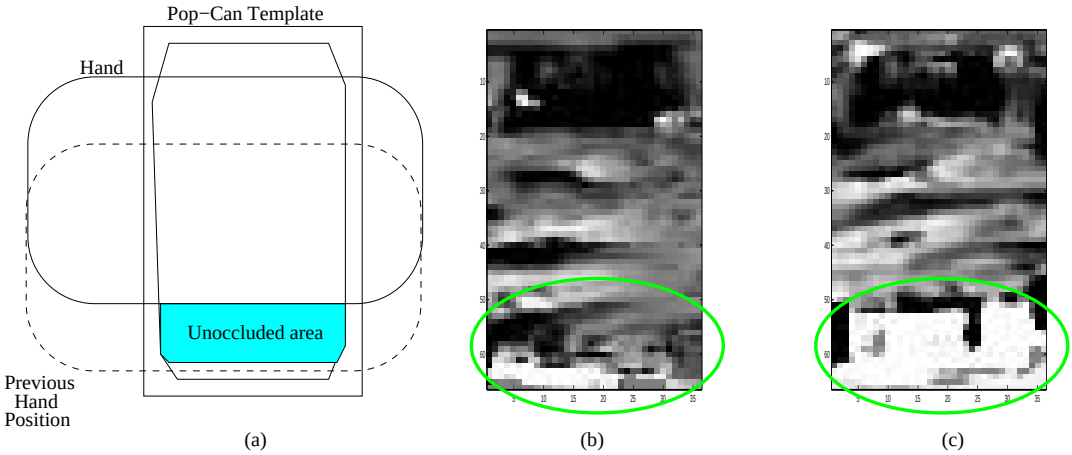
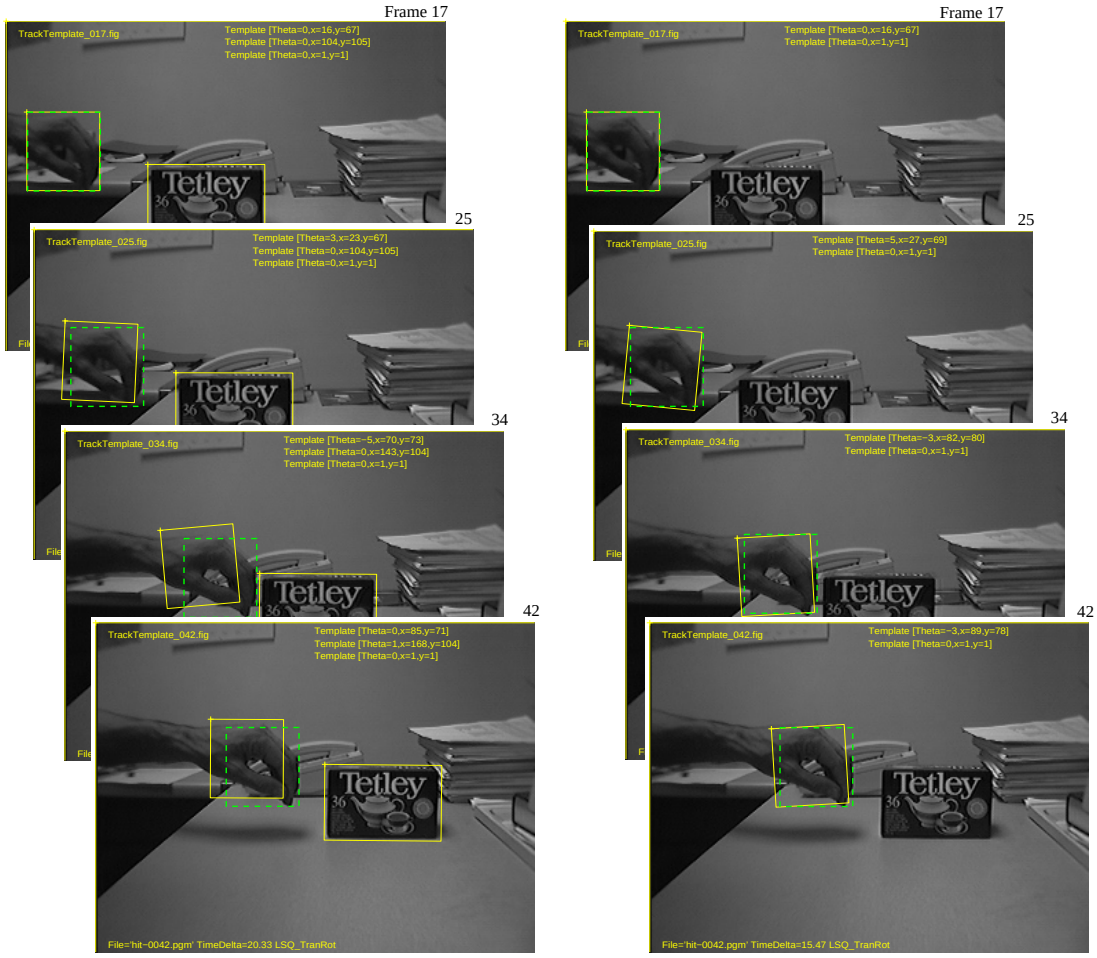


Figure 4.4: Results of Cooperative Templates change. The green ellipses highlight the difference in the ownership probabilities from the basic tracker (b) to the enhanced tracker (c) as an unocclusion event happens. Cooperative templates have changed the low probability (dark grey) to a much higher value (white). The schematic in (a) shows where the unocclusion is occuring.



(a) Unaided Tracker (b) With Ownership Masks

Figure 4.5: Ownership masks have made a significant improvement in tracking accuracy. In the first sequence (a) the template rapidly falls behind the tracked hand. This is corrected for in the second sequence (b) where the Ownership Mask keeps the template closer to the correct position. The small error is due to the delay in forming the ownership mask for the object being tracked. This error accumulates in the first few frames but thereafter is fairly constant.

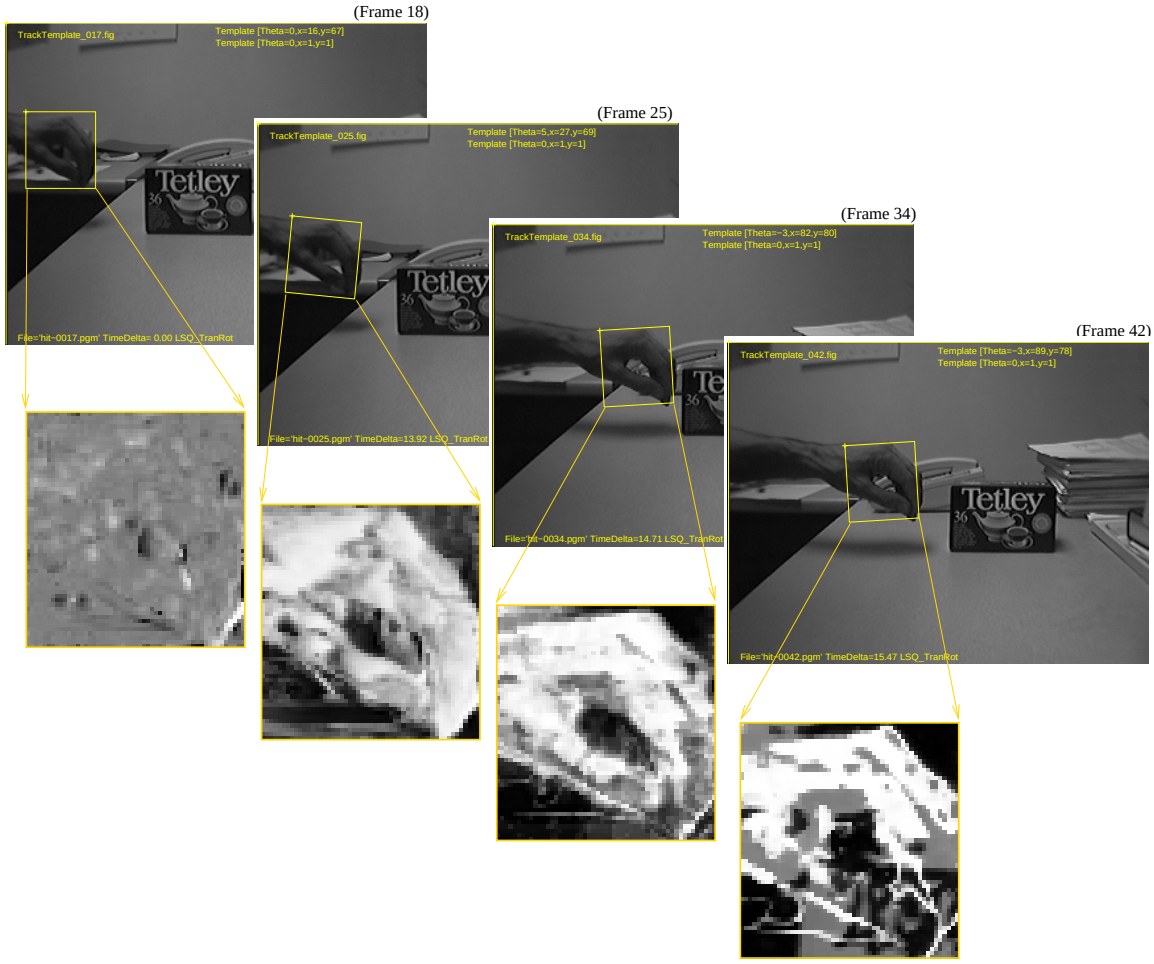


Figure 4.6: Strong spatial partitioning of pixels is evident. The areas of strong(white) and weak(black) ownership correspond roughly to the tracked-object and background pixels.

Chapter 5

Evolving Views

In chapter 3 the Ownership Masks described were shown to improve the tracking of the template tracker in the case where there was substantial background information within the region of the template. They served to remove most of the bias due to objects with other velocities within the template. However, there are other sources of error that Ownership Masks cannot correct. The error induced by the change in the template view as the object goes through occlusion and unocclusion events can be dealt with effectively with the aid of the ownership probabilities described in the Cooperative Templates chapter. These probabilities will allow us to detect occlusion events and modify our template view as it changes.

5.1 Occlusion and Unocclusion Events

There are many examples of occlusion and unocclusion events in everyday life. Occlusion events have been dealt with in the Ownership Mask chapter, however, unocclusions are still problematic. Often separate objects travel together through a scene and then go their

separate ways: a pedestrian walks out from behind a low wall that had occluded their legs, a car drives by a telephone pole obscuring successive sections of the car, or perhaps an AI researcher puts a pop can down and moves his hand out from in front.

In a video sequence the initial segment up until the unocclusion may cause the Ownership Masks to incorrectly use the occluding object's view as if it was part of the tracked object (see Figure 5.1). When the courses of the two objects diverge, the template tracker is left with the problem of determining which object to follow. This is essentially the same problem that was dealt with using Ownership Masks: multiple different velocity sections within the same template. However, there is a twist: the Ownership Masks are already defined and the template view will include the occluding object due to its initialization.

When two objects come together and travel in tandem there may be another problem for a basic template tracker. If the tracked object is occluded when they come together a simple tracker would not be able to distinguish the occlusion from the tracked object. Part of the tracked object will be occluded. The occluding pixels will cause a biasing of the pose estimation away from the occlusions. Again this can occur in many common tasks such as: walking behind a low wall, picking up and moving a large box, or moving your hand to pick up a pop can (see Figure 5.2).

5.2 Changing Template View

In template tracking, a view of the object being tracked is kept as a visual model which can be compared to a subsection of the current image in the image sequence. Often the view of the object is manually created prior to the video sequence or is predefined to be a subsection of a key frame. It is then used to determine the object's pose in the current frame. The template view is transformed into the image coordinate system to the pose

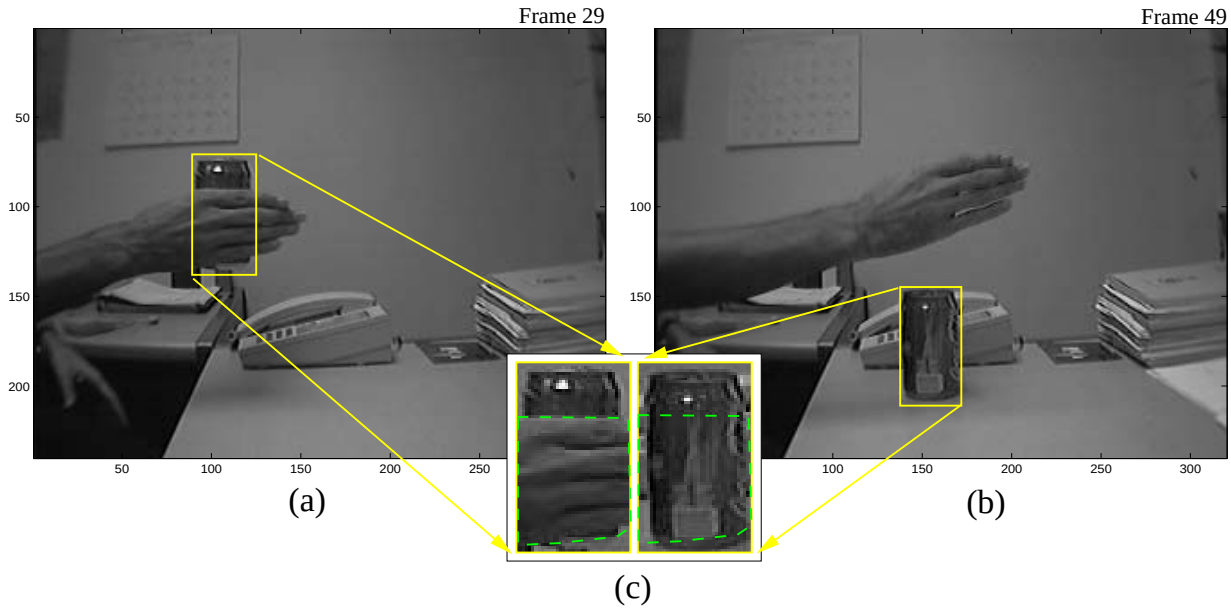


Figure 5.1: Unocclusion sequence. While the pop can (tracked object) is moved to the desktop by the hand (a) it is being occluded. After the pop can is released (b) the tracked object becomes completely visible. The tracker must recognize the unocclusion event indicated by the green dashed line in the inset (c) and must not bias the tracking.

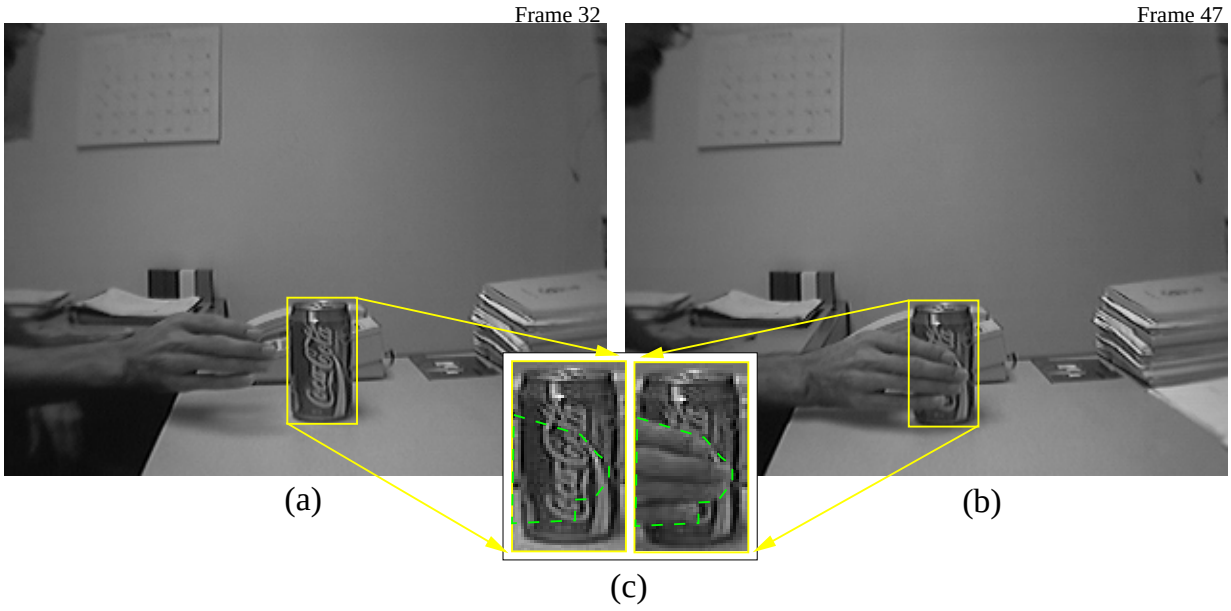


Figure 5.2: Occlusion sequence. Before the hand has reached the pop can (a) the template view is complete. When the hand grasps the can it also occludes part of the view of the can. In the inset (c) the difference in the two template views are shown (green dashed line). The tracker must not be biased by this incursion into the template view.

that best registers it with the current position of the object. An error function is used to determine how well the template is registered in the current frame. The smaller the error the better the registration. However, the actual view of the tracked object in the image frame generally does not match the template view either before an unocclusion (Fig. 5.1) or after an occlusion (Fig. 5.2) event. This presents the tracker with the problem of determining where in the template view the occlusion has occurred, how to maintain the template view and which pixels should be used to track the object.

In an unocclusion event the tracker needs to update the template view with the new view information from the exposed part of the tracked object when it becomes available.

In an occlusion event the template view should not be updated because the new information is from the occluding object and not the tracked object. However, the tracker should not use the occluded part of the template view to determine the change in pose of the tracked object because this would induce a bias similar to the bias from extraneous pixels in chapter 3.

5.2.1 Detecting an Occlusion

For the moment let us assume we have been tracking an object for some time and the ownership mask has had time to evolve. We will also assume that we have a completely visible template view. When a tracked object has been occluded the ownership mask algorithm will detect the occlusion and respond by reducing the ownership probability for the occluding pixels. This decreases the weighting on these pixels in the pose estimation and will, as a result, make the tracking more robust against occluding events. This should prevent the template from being pushed off the tracked object as would happen with a simple tracker (see Fig. 5.3).

But more information can be derived from this event. As the occlusion is happening the

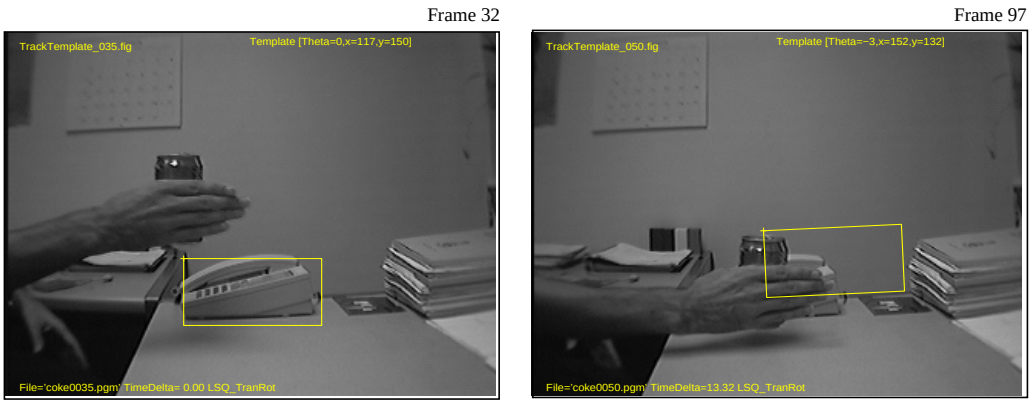


Figure 5.3: Basic template tracker may have problems with occlusions. Here the incursion of the hand causes the template on the phone to be biased away from its proper position.

ownership probability will drop as the pixels on the occluding object obscure the image pixels under the template. The falling probability will be reflected in the reduction of the weights for these pixels in the ownership mask from that of the previous frame (see Fig. 5.4). When we detect a significant reduction we can conclude that an occlusion event is in progress. Thus the change in the ownership probabilities can serve to flag an occlusion.

5.2.2 Detecting Unocclusions When View Known

When an unocclusion event is in progress, if we know the template view already, then an unocclusion results in a higher probability for the unoccluded image pixels because they match the template view (see Fig 5.5). This causes the ownership mask's weights for these pixels to increase. The template view pixels could be updated at this point but they should be the same as the image pixels anyway.

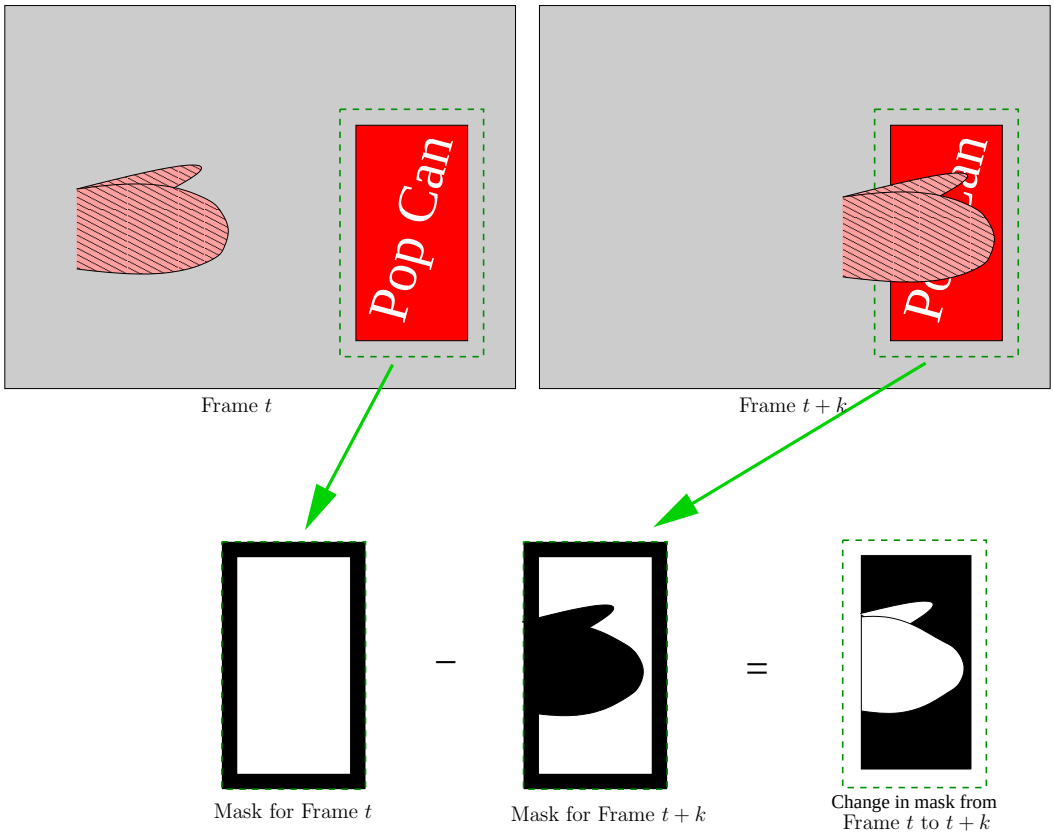


Figure 5.4: Changes in ownership mask highlights new occlusions. In the two top frames a hand passes over the tracked pop can, occluding it. In the bottom frames the mask for the pop can is shown. The white represents the on-object pixels and the black the background. When the hand occludes the pop can the ownership mask responds by eliminating those pixels from the ownership mask. The change between old and new mask indicates where an occlusion has occurred.

5.2.3 Detecting Unocclusions When View Unknown

When an unocclusion exposes a new section of a tracked object there is the problem of detecting this type of event as well as determining how to extract the newly exposed area of the tracked object to update the template view.

The ownership mask improvement allowed us to detect unocclusions when the view *was known* by using the decreasing ownership probabilities that reflected the increasing pixel intensity error. However, there was no mechanism to distinguish occlusion events from unocclusion events when the view *was not known*. Cooperative templates provide this mechanism by changing decreasing ownerships to increasing ownerships during an unocclusion event. Now both versions of unocclusion events will cause an increase in ownership probabilities. This increase in ownership can then be used to trigger an update to the template view.

We want to update the template view in these circumstances because the larger the area of the tracked object being used, the more robust it is to occlusions and errors that may cause problems in the registration. For instance, in the 'Coke-Put' video sequence the can is carried to the desk top and released. The hand which initially occluded the middle and bottom of the pop can now starts moving upwards. This causes the hand to occlude the top of the can. If we only track the top of the can (because it was all we could see initially), then as the hand moves up it completely occludes the only portion of the can that is being tracked. This brings about the complete failure of the tracking of the pop can. Because the basic template tracker searches for the local minima in the error space, it may move the template completely off the image when what is tracked is completely occluded. Obviously no matter how well designed your tracker is, the tracked object can be completely occluded at some time during a video sequence. [1]

[1] An idea to correct for this problem is described in Future Work.

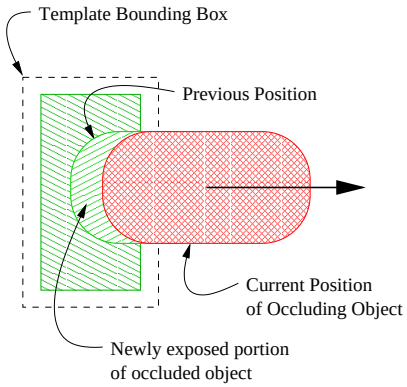


Figure 5.5: Unocclusions when the view is known or unknown. When a tracked object is being unoccluded the newly exposed area will have an ownership $\Delta\hat{O} > 0$ if the template view is already known, otherwise, $\Delta\hat{O} < 0$ in the area the occluding template just vacated.

Care needs to be taken when updating the template view of a tracked object because if we cannot distinguish tracked from non-tracked pixels, we may inadvertently update our template view with pixels from the background or the previously occluding object. This would induce serious tracking problems. When we update the template view pixels they will be included in the next round of pose determination by the tracker. If we have incorrectly included background pixels as part of our template then we will have gotten back into the situation where we are tracking the object as well as the background.

5.2.4 Partial or Full Update of View?

When we know an unocclusion event is occurring we could completely update the template view. This has the advantage of being simple. However, it has a number of disadvantages. Changing the entire template view requires us to re-initialise the Ownership Mask, which induces a time delay before it 'locks' onto the tracked object again. This can cause tracking

bias. This method also suffers because we may lose previously gathered information of the template view when the occlusion transits across the tracked object (see Fig 5.6).

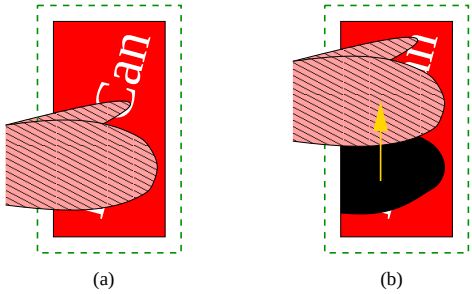


Figure 5.6: Occlusion transits make whole-template-view updates unworkable. If we were to update the entire view when we detected an unocclusion then we would be losing previously known sections of the template view. In (a) we can see the word 'Can' exposed on the tracked pop can. Later as the had moves up (b) an unocclusion has occurred on the lower portion but we cannot update the entire view because the upper portion is now obscured.

Instead what we want to do is limit the updates to the newly visible section of the tracked object (black area in Figure 5.6(b)). This can be done with the help of the cumulative ownership probability as the change in this probability from the previous frame to the current one indicate which pixels in the bounding box are part of the newly exposed portion of the tracked object and which are not.

5.2.5 Leading and Trailing Edges

When an unocclusion is occurring within a template's boundaries the changing ownership probabilities reflect this change, allowing us to detect where the newly exposed areas of the template are occurring.

The changes from one frame to the next will highlight two types of changing regions in

the image. The leading and trailing edges of a moving occluding object will show significant change. The leading edge will be occluding the template or background pixels that it passes over. The trailing edge will be exposing previously covered background or other templated objects. This leaves us with the problem of distinguishing these two types of change.

The leading edge change will be due to the image being occluded by the moving object and the change in ownership will be negative due to the discrepancy between the intensity of the image pixels and that of the Template View pixels.

The trailing edges are where the moving object exposes the image pixels of the underlying objects. As was said in the previous section, the template view of the underlying template may be complete or incomplete in this area. If it is complete the ownership mask values will increase significantly as the template view matches the unoccluded view of the object in the image. When the underlying template view is incomplete, the exposed section of the object in the image belongs to one of the templates that cover that pixel and the cooperative templates fix changes this from the previous negative change in ownership to a positive change.

5.3 Updating Views When Previously Unknown

If multiple templates mutually occlude each other, the template views of these templates will include parts of other objects. When the occluding object exposes more of the tracked object, the newly exposed area of the tracked object does not match the initial template view.

Evolving the template using the change in ownership probabilities as weights for the current image and the current template view allows the template view to evolve enough that a significant unocclusion event may not completely lose tracking as it had before.

The new template view, T_{t+1}, is a combination of the old template view T_t and the current view, C_t, of the tracked object extracted from the image. The proportion of each being related to the strength of the ownership change:

$$T_{t+1} = (1 - f(\Delta\hat{O}_t))T_t + f(\Delta\hat{O}_t)C_t \tag{5.1}$$

$$\Delta\hat{O}_t = \hat{O}_t - \hat{O}_{t-1}$$

$$f(\Delta) = \begin{cases} \Delta & \text{if } \Delta > 0 \\ 0 & \text{otherwise} \end{cases}$$

Where:

- T is the Template View,
- t is the current time or frame,
- $\hat{O}$ is the Cumulative Ownership probabilities, and
- C is the Current View derived from the image I_t.

Equation 5.1 uses the change in ownership to update the template view. A negative change in ownership would indicate that the current view is not matching the template view very well. This would occur when an object is being occluded. A positive change happens when we have an unocclusion of an object that has a complete or incomplete template view and this will trigger an update of the template view.

The results of the addition of cooperative templates and evolving template view can be seen in the Figure 5.7. Here the hand holding the pop can occludes most of the middle and bottom of the can. When the hand releases the can on the desktop it starts unoccluding the bottom and middle of the can but at the same time occludes the top. A simple

tracker would lose tracking for the pop can. The pop can template would just follow the hand Fig. 5.7(a). When we add the cooperative templates, evolving template views and ownership masks to the basic tracker the pop can template stays with the pop can and will start tracking the bottom of the pop can as it is exposed, Fig. 5.7(b) . There is still a little instability that allows the template to wobble as it gains the newly exposed view. Perhaps this is something for future work.

We see in Figure 5.8 that initially the template view of the pop can includes the occluding hand. As the sequence progresses the occlusion in the template view is replaced with the newly exposed view of the pop can. Due to the slight angular bias to the template as the hand leaves the region the view updates with a bend in the can and some spurious wedges coming in on the left.

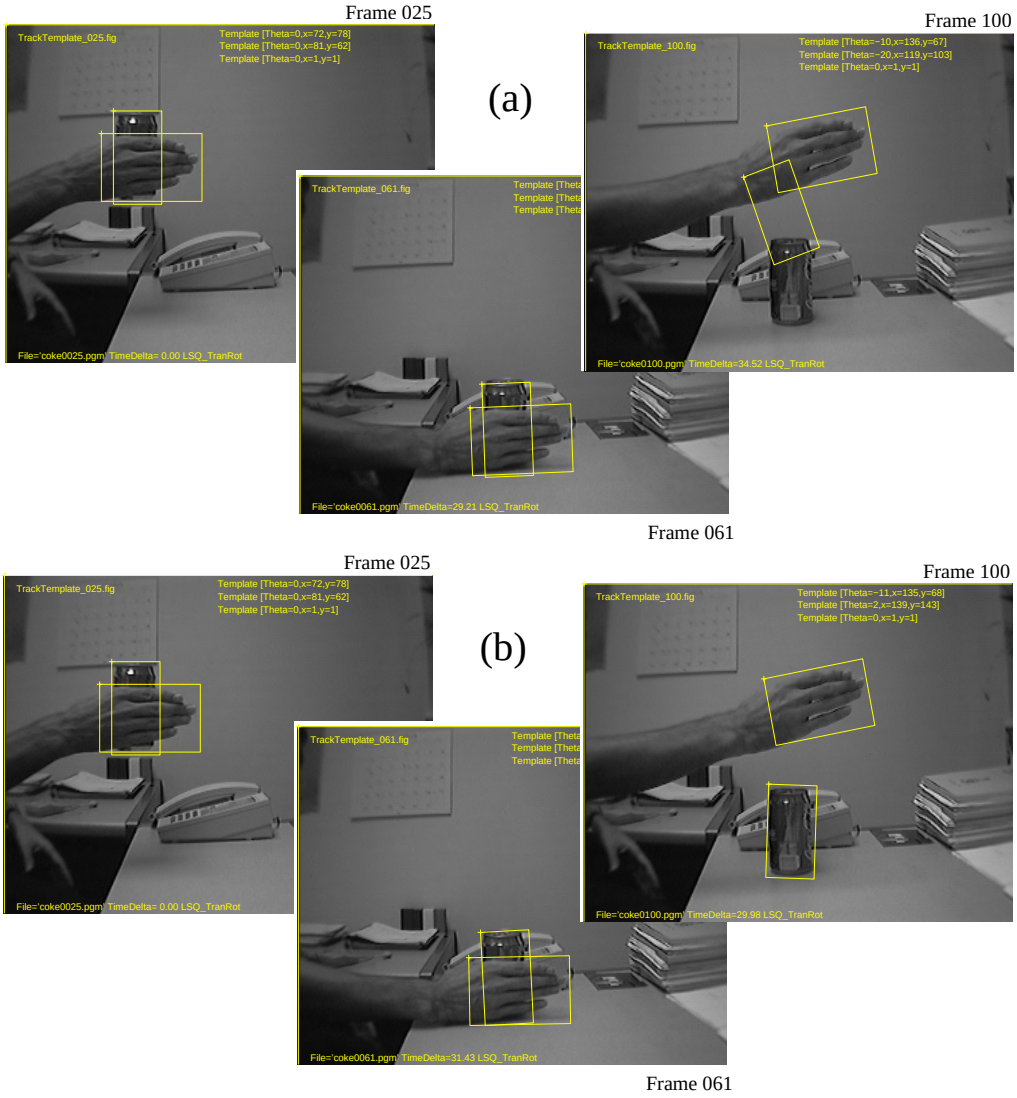


Figure 5.7: Final enhanced tracker can track past occlusion transits. Performance of basic tracker (a). After adding ownership masks, cooperative templates, and evolving views to the basic tracker, it tracks successfully past the occlusion problems.

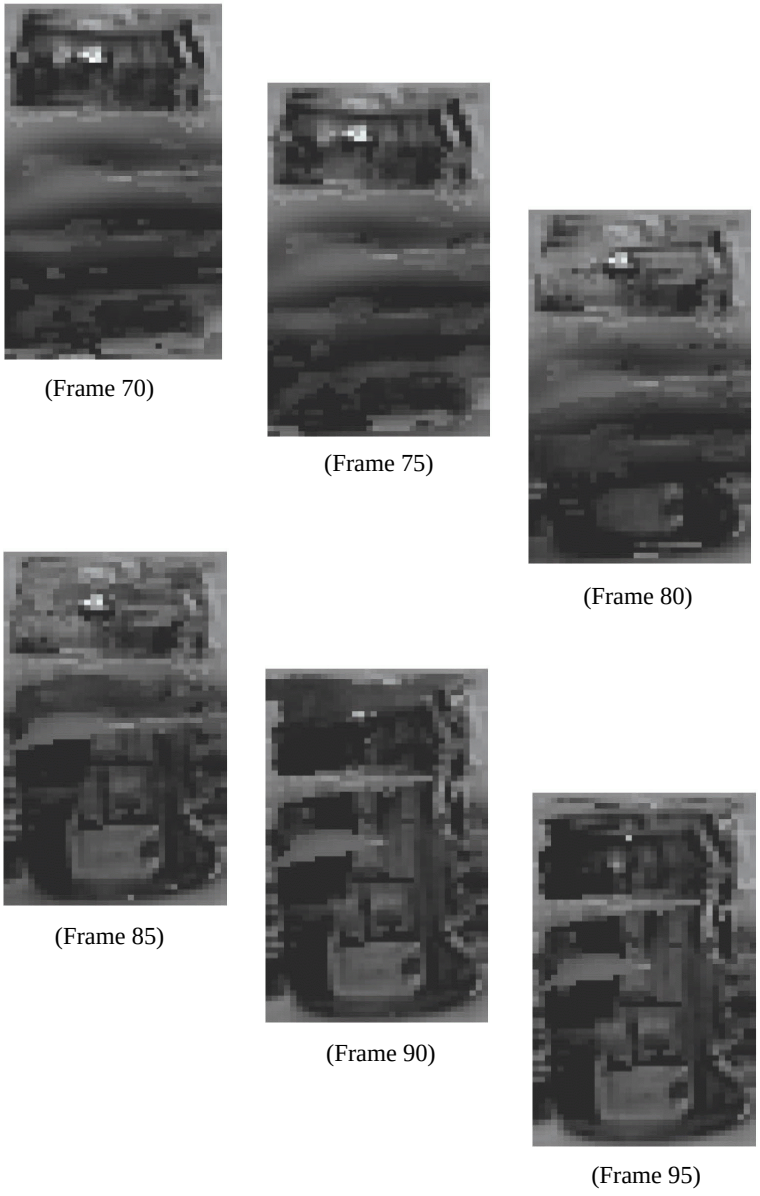


Figure 5.8: Evolving Views updates the template view in an unocclusion. As the image sequence progresses the template view for the pop can updates. The view of the occluding hand is changed to the pop can view.

Chapter 6

Conclusion

6.1 Summary of Work Done

Template tracking in computer vision has been with us for a long time. A basic template tracker works well in most circumstances as long as some conditions are met as was seen in the Introduction.

First, extraneous or non-object pixels have to be kept to a small fraction of the pixels within the template region. Otherwise they may bias the template away from the tracked object. Extraneous pixels in the background of a moving template will tend to bias the template's motion away from the direction of object movement. A static background tends to keep the template stationary as the moving pixels of the tracked object pull the pose estimation in their direction. Often the bias is fairly constant. Other times, depending upon the nature of the background textures, the template will become unstable and wobble back and forth.

Second, occlusions of all varieties must not occur. They act to bias the tracker's pose estimation away from the incursion of the occlusion. In severe cases an occlusion may

actually push a template off the tracked object, causing the tracking to fail completely. The occlusion of a hand reaching to grasp a pop can may cause the template positioned on the can to be pushed off the can away from the incoming hand.

In this thesis we started from a basic template tracker that uses a combination of iterative registration and course-to-fine techniques to register the template view within the image sequence and attacked the above problems one by one.

6.1.1 Ownership Masks

Ownership masks were used to deal with the problem of extraneous or background pixels within the template boundaries. These extraneous pixels were shown to bias the pose estimation of the tracker. Highly detailed backgrounds bias the tracker more than uniform backgrounds. The greater the percentage of extraneous pixels within the template's bounding box the greater the bias in the tracking. Elimination of these extraneous pixels using the ownership mask technique reduced this bias and steadied the tracker thus making the pose determination of the tracker more accurate and robust.

6.1.2 Cooperative Templates

The other two problems dealt with occlusions. When another object in the scene occluded part of the object being tracked, the occluding pixels would cause an area of high error. This would tend to bias the template away from the correct pose, sometimes pushing the template completely off the tracked object. In this case the ownership masks provided the solution by triggering a masking of the occluding object within the template's boundaries. Thus the masked pixels' influence on the pose estimation was reduced and they did not induce as much bias.

The other occlusion problem occurred due to the tracked objects being occluded when the templates were initialised [1]. This was not a problem as long as the objects were travelling with each other but when the occluding object moved away there was the problem of trying to match the true, previously unseen, view of the tracked object with the template view which contained portions of the occluding object. This problem was partially solved using the cooperative templates technique. Using this technique we attempt to segment the image pixels between the templates by deriving the conditional probability of the pixel belonging to a template based on the intensity of the image pixel and the template view's pixel statistics. Some success was found in applying the cooperative template technique to this problem. However, the real purpose of the cooperative templates method was to support the third enhancement: Evolving template views.

6.1.3 Evolving Template Views

When the cooperative templates method was incorporated another problem became obvious. That problem was an occlusion transit over an initially occluded object. An occlusion transit occurs when a tracked object has successive portions of itself occluded as the occluding object moves from one end of the object to the other. What is happening in these circumstances is that tracker is trying to track only a portion of the full object because it is partially occluded when tracking starts. As the occlusion transit continues the originally tracked portion of the object gets occluded and the originally occluded portion becomes visible. The fraction of the object being used to track goes to zero and the template falls

[1] This thesis is my first step in making a fully automated tracker. As such, some of the initial conditions were set with my final goal in mind. One such condition was that tracking would start without any prior knowledge about what a tracked object would look like. Therefore, one goal was to fill in incomplete template views of an object. This is a simple attempt at automated model building.

off the object and tracking is lost.

The evolving template view method was used in conjunction with ownership masks and cooperative templates to capture the newly exposed view of the object and update the tracker's view of what the tracked object looks like. This technique was also found to be useful and experiments showed that this solved the occlusion transit in the video sequence tested.

6.2 Evaluation

In the Ownership Mask chapter the masks based on the image intensity probability showed that eliminating background pixels from the template during the template registration improved the translational accuracy of template. However, this success was tempered by the heavy-handed reduction in all pixels within the template leaving the method open to angular error and potentially eventual tracking failure due to the elimination of too many object pixels (see Figures 3.7, 3.8 and 3.9).

The Cooperative Templates technique partially solved this problem by using the posterior probabilities, that the pixel belonged to the template, which were based on the above pixel intensity probabilities. Thus the ownership is now explicitly accounted for by all the covering templates and not implicitly by some undefined process as was the case previously. Now the thresholding to produce the template masks could be based on the ownership levels, where the likelihoods are equal, instead of an arbitrary threshold as it was before. As well, the Cooperative Templates enhancement allows us to extract information allowing us to differentiate between occlusions and unocclusions when the view is unknown in some circumstances (see Figures 4.4, 4.5 and 4.6). This enhancement has thus improved the tracking problem we had with the above rudimentary ownership masks, improved the

coherence of these masks, reduced the number of object pixels that are eliminated from the masks and given us a foundation for the next enhancement: Evolving Views.

The last enhancement, Evolving Views, takes the Cooperative Templates ability to differentiate between occlusion and some unocclusion events and uses that to trigger an update to the template's view in the case where it was previously unknown. This change has allowed the tracker to avoid the occlusion transit problem discussed in the introduction and has had some success in completing the missing parts of the template view (see Figures 5.7 and 5.8).

6.3 Future Work

There are a number of directions this work can be taken. This tracker was intended as the first step in producing a fully autonomous tracker; one that can initialise and maintain a template throughout the object's presence in the image sequence and beyond. As was previously mentioned, there are a number of key areas that present themselves for future projects.

6.3.1 Object Search

During the development of this tracker, the error surface of Equation 2.1 was explored. As Fig. 6.1 and the closeup in Fig. 6.2 suggest, the error surface is quite complex and local minima abound.

A video of a single plane through the 3D error space was made and this showed a bifurcation of the global minima. Likely this was caused by only exploring a subspace of the error volume but it would be interesting to explore the error space further and find the shape of the constant error surfaces. This may lead to further understanding of the global

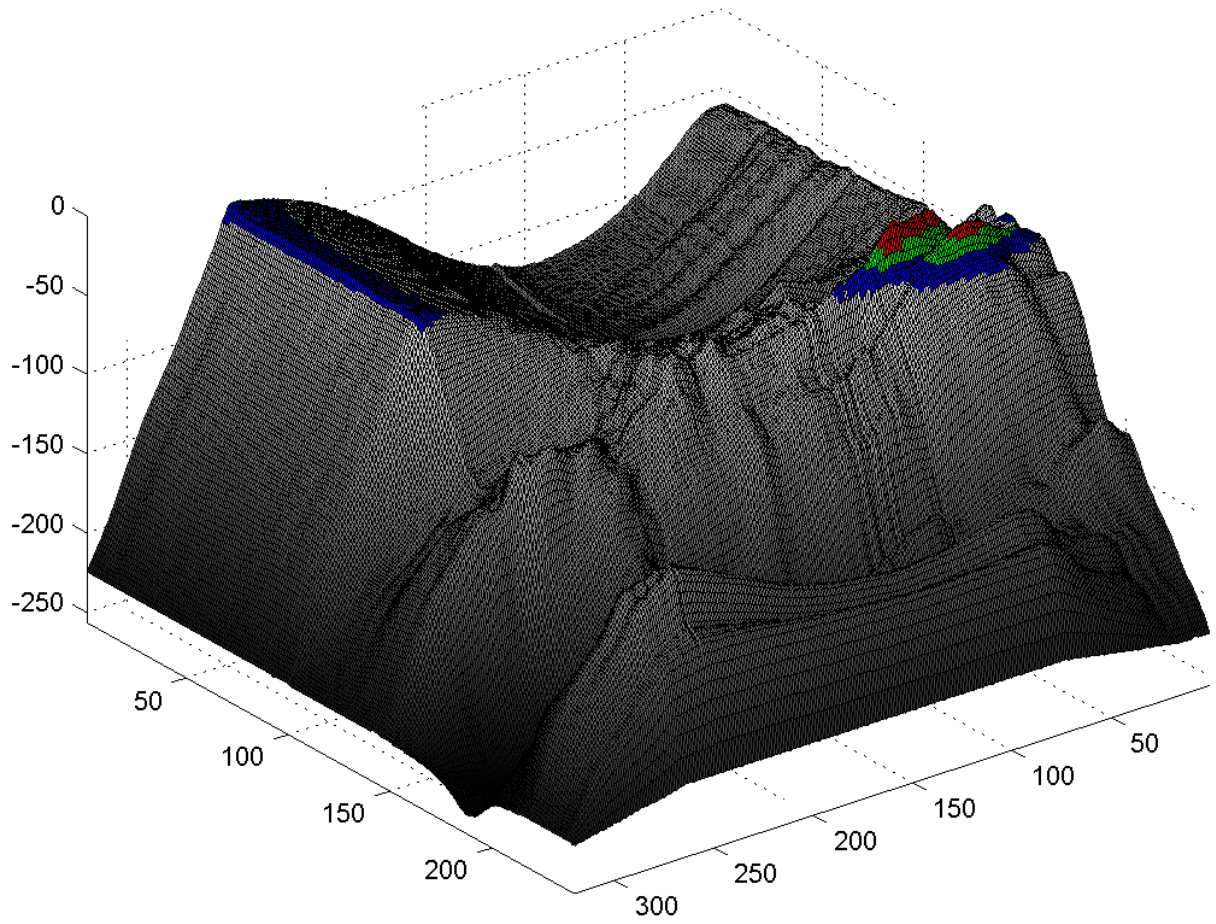


Figure 6.1: Inverted error surface in the Coke-Put sequence. The global and a local minima (the two peaks on the RHS) are highlighted in colour. This local minima is very close in location and magnitude to the global minima making it a significant problem. Note the ramp visible on the front two edges is due to the template travelling partially off the image which induces a higher error by design.

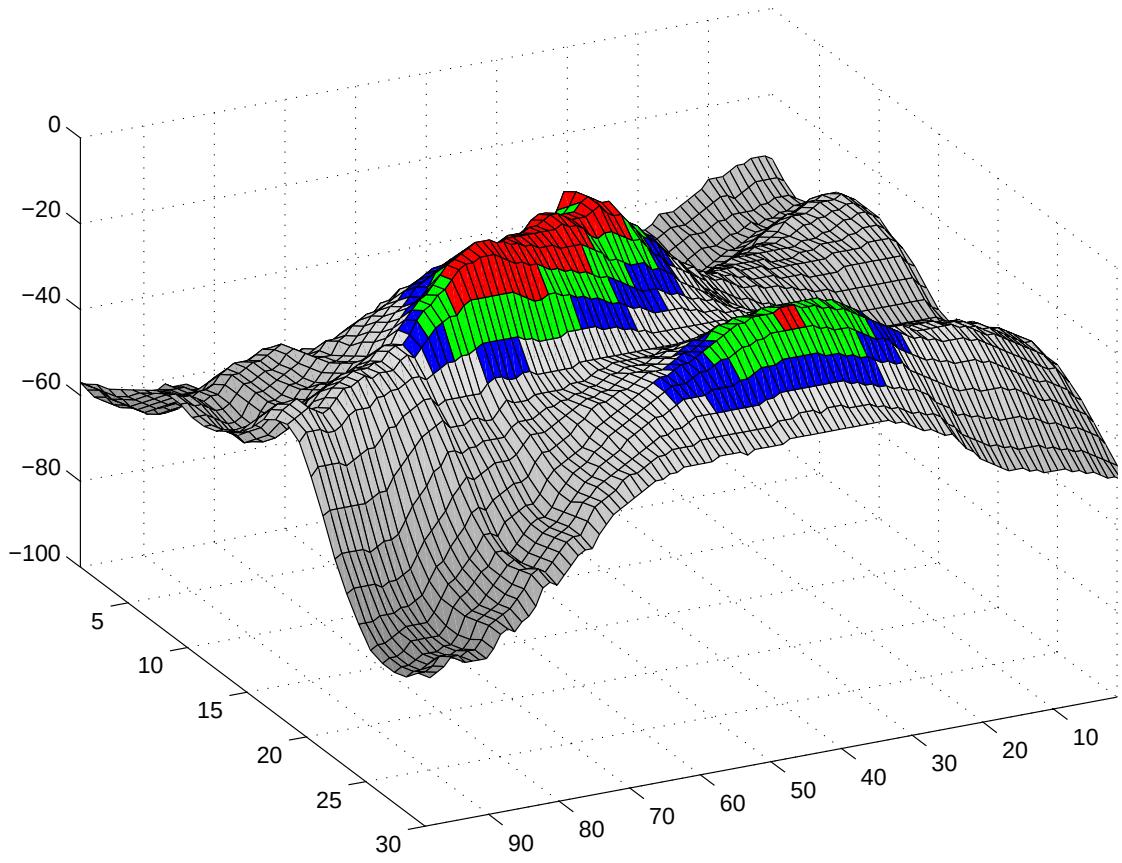


Figure 6.2: Closer view of neighbouring global and local minima in prevous figure. The difference in the heights of the minima is small, which may cause a search algorithm to get caught in the wrong minima.

minima search problem.

6.3.2 Auto-Initialization

Another challenge is to be able to auto-initialise the templates based upon movement in the image. A number of techniques to initialise present themselves. Optical flow or the much simpler frame differencing method can be used to highlight the portions of the image undergoing movement (see Figures 6.3 and 6.4). Both of these techniques would initialise template bounding boxes that are fairly loose and have a large number of extraneous pixels. Ownership masks have been partially developed to solve this problem.

Stationary Object Segmentation

Except for the background, all the templates in the video sequence could be moving at some point in time and stationary at others. When an object moves, this provides an opportunity to segment the moving object as well as any stationary objects which may occlude it. If a hand moves behind a stationary mug sitting on the desk we can invoke the rule of object permanence again and say the hand did not disappear; it was occluded by something stationary.

Malleable Templates

Problems with static-sized templates abound. Objects in real world scenes do not behave the way we would expect they would. Unocclusions may reveal protrusions that cannot be accounted for within the boundaries of a pre-initialised rectangular template. A rectangular template also may not be refined enough to track an irregular object correctly. The ownership mask method reduces the effect of extraneous pixels but requires accurate tracking while the mask matures. There are circumstances where this is not feasible. However,

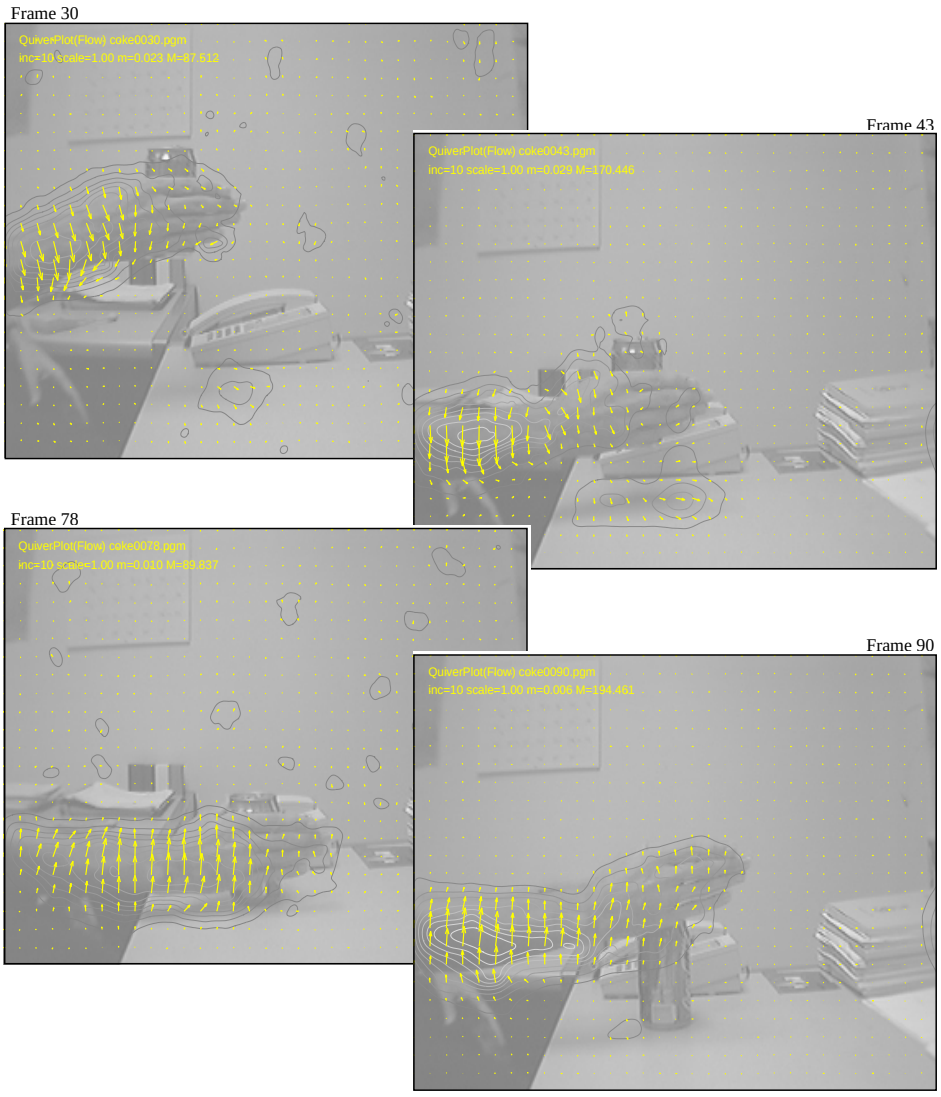


Figure 6.3: Optical Flow run on the Coke-Put sequence. As can be seen by the arrows in Matlab's quiver plot the velocity of the movement can also be extracted. The extent of the moving object is roughly delineated by the larger flow vectors although the flow on the shadow below the hand (Frame 43) may cause problems. Contours on the image are the constant velocity contours.

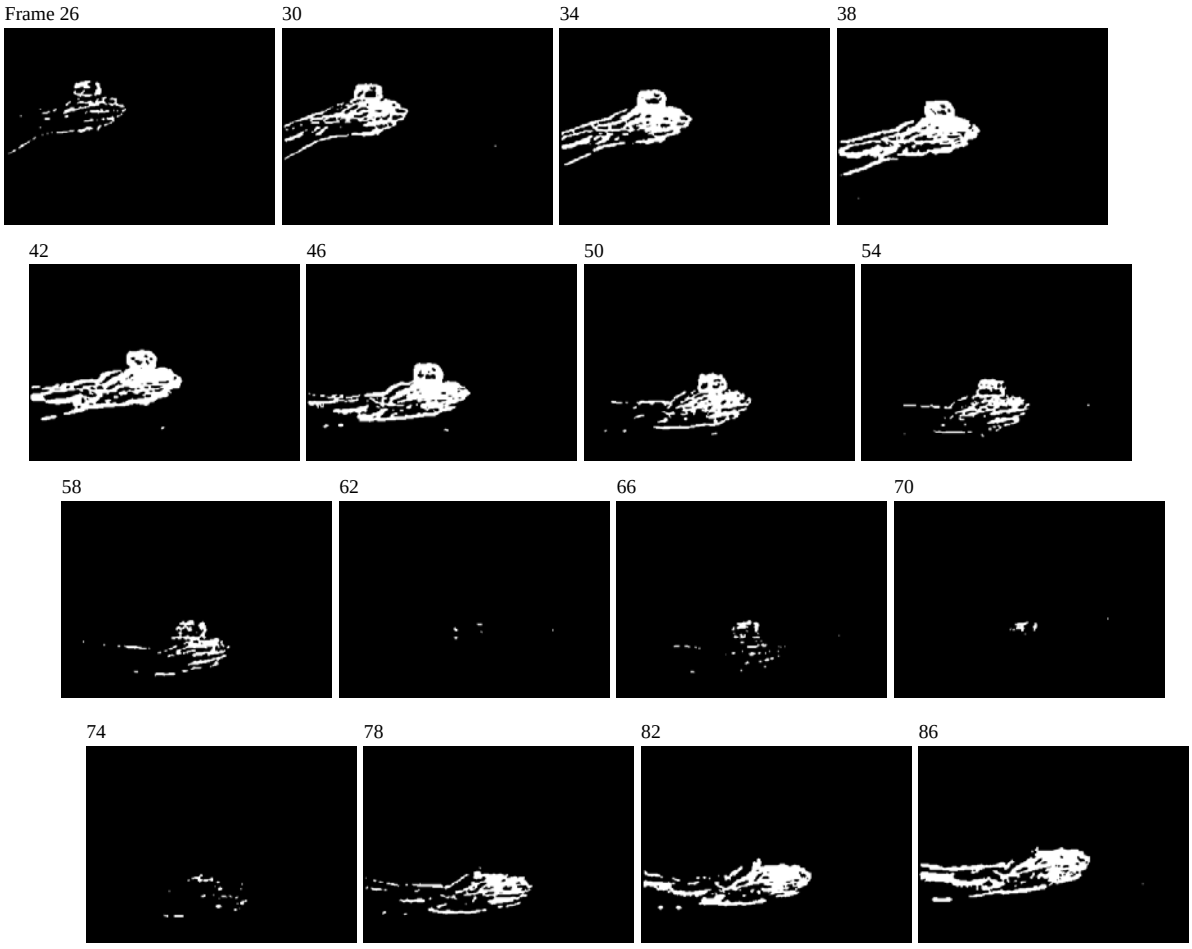


Figure 6.4: Extent of moving object may be provided by frame differencing. In this sequence frame differencing is is followed by thresholding and removing the speckling(isolated pixels).

the optical flow or frame differencing techniques may offer a solution to this problem. They can provide a raw estimate of the object's extent. We can then provide a tightly bounding template to cover the object. Malleable templates would then adjust the boundary of the object if there was evidence of the object beyond the current boundaries or if the margin of background around the object is too large.

6.3.3 Object Permanence

Another challenge is the maintenance of the templates as the objects evolve in the image. Objects coming on screen or going off screen change in size depending upon the percentage visible at the time. When they have moved off screen they may come back on screen again. It would be desirable to be able to associate a previous instance of an object with its current instance. A tracker should have a sense of object permanence.

Full Occlusions and Model Switching

All the problems with occlusions have not been solved. Full occlusions of the tracked object cause the basic tracker to completely lose tracking. This is because the tracker searches for a best match for the tracked object in the image. If the object disappears completely then it is either occluded within the scene or occluded because it moved out of the scene. A simple threshold on the error function may be useful in detecting these events. Then when the tracker detects a full occlusion it can switch models to something else: like a constant velocity model.

6.3.4 Articulated Objects

Articulated objects present a problem for a template tracker that is based on a rigid internal model (the template view). But often articulated objects are constructed from multiple rigid segments joined together. The human body, for instance, is based upon a skeletal system that is rigid locally but articulated globally. Templates on the arm and leg segments, the body and the head may be able to track their locally rigid parts. Shannon Ju, et. al. found in their Cardboard People paper [19] that this could be done with constraints on the joints. But what conditions would be necessary to discover these constraints in an auto-initialised tracker?

Appendix A

Template Tracking Mathematics

A.1 Template Tracking Algorithm

Template tracking works by registering two views of the same object. One, the template view that the algorithm keeps as a visual model of the tracked object, and the other the view of the tracked object in the current image.

The object is to find the transformation parameters $\vec{a}$ that register the template view with the view of the tracked object in the current image, I_t, of a video sequence. This registration minimises the error function $\mathcal{E}(\vec{a})$ between the template view (taken from the key image, J, in the sequence at some time prior to t) and the corresponding image, I_t, at the current time t and transformational parameters $\vec{a}$.

The error in the template T is given by:

$$\mathcal{E}(\vec{a}) = \sum_{\vec{x}\epsilon T} \rho(J(\vec{x}) - I(m(\vec{x};\vec{a}))) \tag{A.1}$$

Where T is the set of pixels in the template with $(0,0)$ in the upper left corner and

aligned with the axes. That is $T = \{0, \ldots, W-1\} \times \{0, \ldots, H-1\}$ where W is the width of the template in pixels and H is the height. The coordinates $\vec{x}$ of the pixels are in the template coordinate system and the coordinates of the pixels in the image coordinate system are $m(\vec{x}; \vec{a})$. The parameters $\vec{a}$ define the pose of the template when registered.

A.1.1 The Estimators

The function $\rho(\ldots)$ is the estimator or weighting function that weights the contribution of the error in each pixel's intensity to the total error $\mathcal{E}(\vec{a})$:

LSQ Estimator:

$$\rho(e) = e^2 \tag{A.2}$$

Robust Estimator:

$$\rho(e) = \frac{e^2}{\sigma^2 + e^2} \tag{A.3}$$

Where σ is a constant defining the spread of the function along the x-axis.

The LSQ weighting function has the problem that it weights larger errors more significantly than smaller errors. This causes large errors to have larger significance in the calculation of the total error, which may bias the tracker towards the outliers. The Robust weighting function asymptotically approaches C from below as e approaches infinity (see Fig A.1).

In this paper the ownership masks were used to exclude outliers so the Robust weighting function was avoided in favor of the simpler LSQ function.

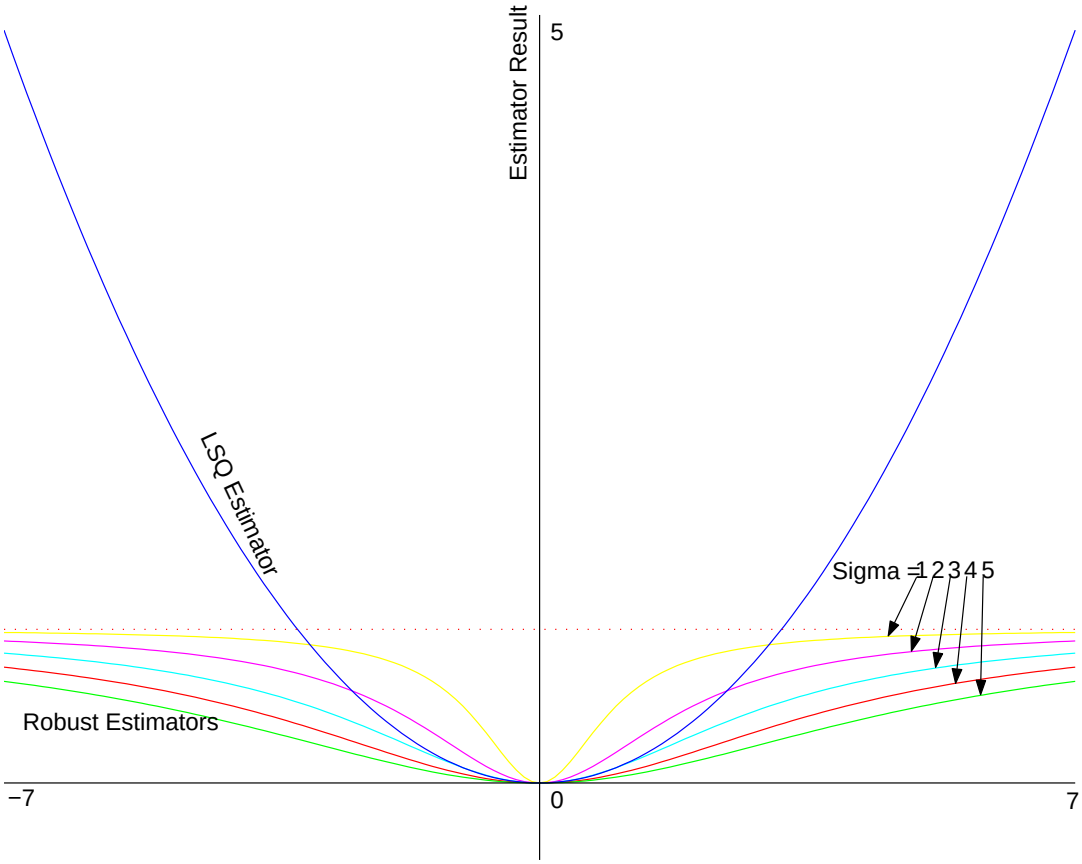


Figure A.1: The LSQ Estimator, Robust Estimator and asymptote.

The vector $\vec{a} = (a_x, a_y, a_\theta)$ is a set of parameters that represents the pose of the template. In this case it allows for translation and in-plane-rotations (Z-axis rotations) of the template.

Iterative registration was used. During each step the pose of the template $\vec{a}$ is updated by the estimated change in pose $\delta\vec{a}$.

$$\vec{a}\,' = \vec{a} + \delta\vec{a} \tag{A.4}$$

A.1.2 Linearized Intensity

To solve for the optimal $\delta\vec{a}$, we linearize the template image intensity map at each point by using a first degree Taylor approximation:

$$\begin{aligned} I(m(\vec{x};\vec{a}\,')) &= I(m(\vec{x};\vec{a}+\delta\vec{a})) && \text{(A.5)} \\ &= I(m(\vec{x};\vec{a})) + \nabla I(m(\vec{x};\vec{a})) \times \frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}}\delta\vec{a} + O(\|\delta\vec{a}\|^2) && \text{(A.6)} \end{aligned}$$

Where $\bigtriangledown I(m(\vec{x};\vec{a}))$ is the gradient of the current image at the template point $\vec{x}$ and initial [1] pose $\vec{a}$. The value $\frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}\,'}\vec{\delta a}$ is the full derivative of the transformation $m(\vec{x};\vec{a})$ by the parameters of the transformation $\vec{a}$. NOTE: In the case of a translational and rotational transformation $\vec{a}\,'\epsilon\Re^3$ and $\vec{x}$ is a $\Re^2$ vector in the image plane. Therefore, $\frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}\,'}\vec{\delta a}$ is a $\Re^2 \times \Re^3$ vector (a 2x3 matrix).

The error in the Taylor's approximation is $\Theta(\|\vec{\delta a}\|^2)$ which stays small as long as the iteration in our pose parameters is small. Therefore, a good pose prediction method is important to keep the pose increment small and thus the linearization error small.

[1] For the current image frame and blur level.

To derive $\delta\vec{a}$ we need to reformulate the error function in terms of $\delta\vec{a}$ and linearize it using the Taylor's Approximation.

$$\begin{aligned}
\mathcal{E}(\vec{a}\,') &= \sum_{\vec{x}\epsilon T} \rho(I(m(\vec{x};\vec{a}\,')) - J(\vec{x})) && \text{(A.7)}\\
&= \sum_{\vec{x}\epsilon T} \rho\{I(m(\vec{x};\vec{a})) + \bigtriangledown I(m(\vec{x};\vec{a}))\,\frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}\,'}\vec{\delta a} && \text{(A.8)}\\
&\quad +\Theta(\|\vec{\delta a}\|^2) - J(\vec{x})\}\\
&\approx \sum_{\vec{x}\epsilon T} \rho\{I(m(\vec{x};\vec{a})) + \bigtriangledown I(m(\vec{x};\vec{a}))\,\frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}\,'}\vec{\delta a}\\
&\quad -J(\vec{x})\} && \text{(A.9)}\\
&= \mathcal{E}(\vec{a}+\vec{\delta a}) && \text{(A.10)}\\
&= \mathcal{G}(\vec{\delta a}) && \text{(A.11)}
\end{aligned}$$

Where $\mathcal{G}(\vec{\delta a})$ is the total pixel error in the template dependent on $\vec{\delta a}$, the deviation from the starting point[2].

Now as $\vec{\delta a} \longrightarrow 0$, $\Theta(\|\vec{\delta a}\|^2) \longrightarrow 0$ and $\mathcal{G}(\vec{\delta a}) \longrightarrow \mathcal{E}(\vec{a}\,')$.

To solve for $\vec{\delta a}$ we have to find the minimum of $\mathcal{G}(\vec{\delta a})$. Where $\frac{d\mathcal{E}}{da} \approx \frac{d\mathcal{G}}{d(\vec{\delta a})}$. This can be done by solving $\frac{d\mathcal{G}}{d(\vec{\delta a})} = 0$. To do this we note in Equation A.9 that $J(\vec{x})$ is just the intensity of the *base* image pixel $\vec{x}$. $I(m(\vec{x};\vec{a}\,'))$ is the intensity of the pixel in the current image I. $\bigtriangledown I(m(\vec{x};\vec{a}))$ is the image intensity derivative. $\frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}\,'}$ is the derivative of the transformation and $\vec{\delta a}$ the variable. Note the parameter to the function $\rho(\ldots)$ is linear in $\vec{\delta a}$. The value of this vector that minimises $\mathcal{G}(\vec{\delta a})$ is the change in transformation that will bring the templates closer to the proper alignment.

[2]For the current image and blur level.

Note also that since $\vec{x}$ after transformation (and possibly before) will have non-integral component values, we will need to interpolate when calculating the image intensity values for $\vec{x}$ and $m(\vec{x};\vec{a}\,')$. This interpolation can be done in a variety of ways, but the simplicity and speed of bilinear interpolation (see Appendix B.3) makes it an attractive solution. Other interpolation methods that may be useful for higher order smoothness are: Bicubic Interpolation and Bicubic Spline (see [25]).

Now using Eqn. A.9 we differentiate and equate to the zero vector to find the $\vec{\delta a}$ for which the function $\mathcal{G}(\vec{\delta a})$ is a minimum.

$$\mathcal{G}(\delta\vec{a}) = \sum_{\vec{x}\in T} \rho(\{I(m(\vec{x};\vec{a})) + \nabla I(m(\vec{x};\vec{a})) \times \frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}} \delta\vec{a}\} - J(\vec{x})) \tag{A.12}$$

For the LSQ estimator $\rho(e) = e^2$ we have:

$$\begin{aligned} \frac{d\mathcal{G}(\delta\vec{a})}{d(\delta\vec{a})} &= 2\sum_{\vec{x}\in T}[I(m(\vec{x};\vec{a})) + \nabla I(m(\vec{x};\vec{a})) \times \frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}} \delta\vec{a} - J(\vec{x})] \\ &\quad \times \nabla I(m(\vec{x};\vec{a})) \times \frac{\partial m(\vec{x};\vec{a})}{\partial \vec{a}} \end{aligned} \tag{A.13}$$

We find the extrema by setting Eqn A.13 to zero and solving for $\delta\vec{a}$:

$$\vec{0} = \sum_{\vec{x}\in T} \left[I(\vec{x}\,') + \nabla I(\vec{x}\,') \times \frac{\partial \vec{x}\,'}{\partial \vec{a}} \delta\vec{a} - J(\vec{x}) \right] \times \nabla I(\vec{x}\,') \times \frac{\partial \vec{x}\,'}{\partial \vec{a}} \tag{A.14}$$

Where $\vec{x}\,' = m(\vec{x};\vec{a})$.

Converting to the matrix form of eqn A.14

$$\vec{0} = \left(\nabla I(\vec{x}\,') \times \frac{\partial \vec{x}\,'}{\partial \vec{a}} \right)^T \left[I(\vec{x}\,') + \nabla I(\vec{x}\,') \times \frac{\partial \vec{x}\,'}{\partial \vec{a}} \delta\vec{a} - J(\vec{x}) \right] \tag{A.15}$$

It is instructive to see its dimensional equation (n is the number of pixels in T):

$$[3 \times 1] = ([n \times 2][2 \times 3])^T ([n \times 1] + [n \times 2][2 \times 3][3 \times 1] + [n \times 1])$$

In this thesis we allow the template to go through two types of pose transformation: Translation and Z-Rotation (in the plane). The following section is specific to these types of transformations.

$$\vec{x}\,' = m(\vec{x}; \vec{a}) = \begin{bmatrix} m_x \\ m_y \end{bmatrix} = A\vec{x} + \vec{b} = \begin{bmatrix} cos\, a_\theta & -sin\, a_\theta \\ sin\, a_\theta & cost\, a_\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{A.16}$$

where $\vec{x}\,'$ is in the image, $\vec{x}$ is in the template, $\vec{a} = (a_x, a_y, a_\theta)^T$, and

Which gives:

$$\left(\frac{dm(\vec{x}; \vec{a})}{d\vec{a}}\right) = \begin{bmatrix} \frac{\partial m_x}{\partial a_x} & \frac{\partial m_y}{\partial a_x} \\ \frac{\partial m_x}{\partial a_y} & \frac{\partial m_y}{\partial a_y} \\ \frac{\partial m_x}{\partial a_\theta} & \frac{\partial m_y}{\partial a_\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -xsin\, a_\theta - ycos\, a_\theta & xcos\, a_\theta - ysin\, a_\theta \end{bmatrix}$$

A.1.3 Iterative Solution to Registration Problem

Rearranging equation A.15 we get:

$$\left(\nabla I(\vec{x}\,')\frac{\partial \vec{x}\,'}{\partial \vec{a}}\right)^T (I(\vec{x}\,') - J(\vec{x})) = -\left(\nabla I(\vec{x}\,')\frac{\partial \vec{x}\,'}{\partial \vec{a}}\right)^T \left(\nabla I(\vec{x}\,')\frac{\partial \vec{x}\,'}{\partial \vec{a}}\delta\vec{a}\right) \tag{A.17}$$

Identifying and substituting for the common terms:

$$\left(\nabla I(\vec{x}\,')\frac{\partial \vec{x}\,'}{\partial \vec{a}}\right)^T (I(\vec{x}\,') - J(\vec{x})) = -\left(\nabla I(\vec{x}\,')\frac{\partial \vec{x}\,'}{\partial \vec{a}}\right)^T \left(\nabla I(\vec{x}\,')\frac{\partial \vec{x}\,'}{\partial \vec{a}}\right)\delta\vec{a} \tag{A.18}$$

Where $Q = \left(\nabla I(\vec{x}\,')\frac{\partial \vec{x}\,'}{\partial \vec{a}}\right)$ and is a $n \times 3$ vector (where n is the number of pixels) and Q^TQ is a 3×3 matrix. Replacing the bracketed term by Q we get the simpler equation:

$$Q^T(I(\vec{x}\,') - J(\vec{x})) = -(Q^TQ)\delta\vec{a} \tag{A.19}$$

Solving for $\delta\vec{a}$ we get:

$$\delta\vec{a} = (Q^TQ)^{-1}Q^T(I(\vec{x}\,') - J(\vec{x})) \tag{A.20}$$

Its dimensional analysis:

$$[3 \times 1] = ([n \times 3]^T[n \times 3])^{-1}[n \times 3]^T([n \times 1] - [n \times 1])$$

Thus the solution using the pseudo (or generalised) inverse [14] $Q^\dagger = (Q^TQ)^{-1}Q$ is:

$$\delta\vec{a} = -(Q^TQ)^{-1}Q^T \times (I(\vec{x}\,') - J(\vec{x})) \tag{A.21}$$

This is the least squares solution of the over constrained system. There are other ways to solve over constrained linear systems (see [29, Appendix A.6]).

Appendix B

Mathematical Tools

B.1 Gaussian Blurring.

The smoothing, or blurring, of an image can be accomplished with a Gaussian kernel:

$$G(h,k) = Ce^{(-\frac{h^2+k^2}{2\sigma^2})}$$

This equation is separable so that:

$$G(h,k) = Ce^{(-\frac{h^2}{2\sigma^2})} \times e^{(-\frac{k^2}{2\sigma^2})} \tag{B.1}$$

Then when we convolve Equation B.1 with our image $I(h,k)$ we get:

$$
\begin{aligned}
I_G(i,j) &= I * G \\
&= \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} G(h,k) I(i-h, j-k) \\
&= C \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} e^{(-\frac{h^2}{2\sigma^2})} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} e^{(-\frac{k^2}{2\sigma^2})}
\end{aligned}
$$

Thus to convolve the full 2-dimensional Gaussian kernel with our image we can apply a 1-D Gaussian kernel first to the columns and then to the rows of the image.

To create a 1-D discrete Gaussian kernel we need to start with a 1-D continuous Gaussian kernel $N(\mu, \sigma)$. Then the mask width w must be determined. We want the width w to cover the bulk of the continuous distribution and be symmetrical about the mean. With an $N(0,1)$ distribution the CDF at 2.5σ is 0.993790. This gives a coverage of $(0.993790 - (1 - 0.993790)) = 0.98758$. Thus $w = 5\sigma$ gives a 98.76% coverage. (See [29, page 55]).

B.2 Image Derivatives.

To determine the five-point Central-difference approximations we need to start with the Taylor approximations:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + O(h^4) \quad \text{(B.2)}$$

$$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) + O(h^4) \quad \text{(B.3)}$$

$$f(x+2h) = f(x) + 2hf'(x) + 2h^2 f''(x) + \frac{8}{6}h^3 f'''(x) + O(18h^4) \quad \text{(B.4)}$$

$$f(x-2h) = f(x) - 2hf'(x) + 2h^2 f''(x) - \frac{8}{6}h^3 f'''(x) + O(18h^4) \quad \text{(B.5)}$$

Next find the difference between Equations B.2& B.3 and B.4& B.5:

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{1}{3}h^3 f'''(x) + O(h^4)$$

$$f(x+2h) - f(x-2h) = 4hf'(x) + \frac{8}{3}h^3 f'''(x) + O(h^4)$$

adding the above two equations together we get:

$$-f(x+2h) + f(x-2h) + 8f(x+h) - 8f(x-h) = 12hf'(x) + O(h^4)$$

Which, when solved for $f'(x)$, gives:

$$f'(x) = \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} + O(x^4)$$

For further information on deriving formulae for Numerical Derivatives please see Trucco&Verri [29, Appendix A.2].

B.3 Bilinear Interpolation

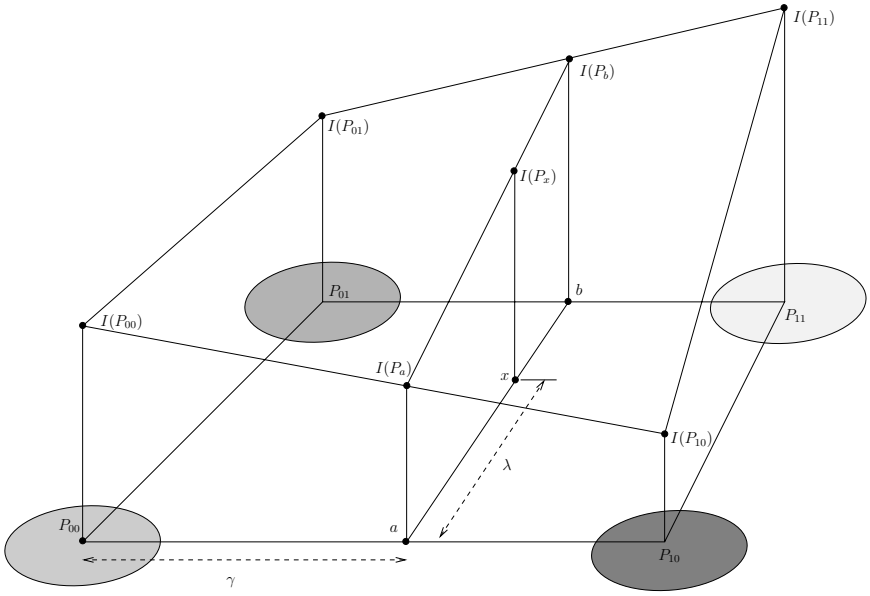


Figure B.1: Using bilinear interpolation to get non-grid pixel intensities.

Given the intensities $\{I_{00} = I(P_{00}), I_{10}, I_{01}, I_{11}\}$ of the four corners $\{P_{00}, P_{10}, P_{01}, P_{11}\}$ of the square in the regular grid that our point of interest $\vec{x}$ lies in we can approximate the intensity of the point itself: $I(\vec{x})$. First we have to find the value $I(\vec{x})$ which is dependent on $I(\vec{a})$ and $I(\vec{b})$:

$$I(\vec{x}) = \lambda I(\vec{b}) + (1 - \lambda) I(\vec{a}) \tag{B.6}$$

Where $\lambda = \|\vec{x} - \vec{a}\|$ is the fractional part of $\vec{x}$ in the first coordinate.

$I(\vec{x})$ is then the linearly interpolated value between the intensity values of the two pseudo points $\vec{a}$ and $\vec{b}$. These intensity values can be calculated from the four corner point intensity values as follows:

$$I(\vec{a}) = \gamma I(\vec{P_{10}}) + (1-\gamma) I(\vec{P_{00}}) \quad \text{(B.7)}$$

$$I(\vec{b}) = \gamma I(\vec{P_{11}}) + (1-\gamma) I(\vec{P_{01}}) \quad \text{(B.8)}$$

Where $\gamma = \|\vec{a} - P_{00}\|$ is the fractional part of $\vec{x}$ in the second coordinate.

Combining the above three equations we get:

$$\begin{aligned} I(\vec{x}) &= \lambda(\gamma I(\vec{P_{11}}) + (1-\gamma) I(\vec{P_{01}})) \\ &\quad +(1-\lambda)(\gamma I(\vec{P_{10}}) + (1-\gamma) I(\vec{P_{00}})) \\ &= \lambda\gamma I(P_{11}) + \lambda(1-\gamma) I(P_{01}) \\ &\quad +(1-\lambda)\gamma I(P_{10}) + (1-\lambda)(1-\gamma) I(P_{00}) \end{aligned} \quad \text{(B.9)}$$

This can also be used to derive the image derivatives $\nabla I(\vec{x})$ given the derivatives $\{I_{x,00} = I_x(P_{00}), I_{x,10}, I_{x,01}, I_{x,11}\}$ and $\{I_{y,00} = I_y(P_{00}), I_{y,10}, I_{y,01}, I_{y,11}\}$ at the corners.

Note that the approximation to the real surface by two triangular planes is fairly coarse and the first derivatives from neighbouring grid squares will not match.

In terms of calculation efficiency Eqn. B.9 requires 7 add's/8 mult's whereas Eqn. B.7 and B.8 requires $4+/4\times$ and substituting the answers into Eqn. B.6 brings the grand total to $6+/6\times$ (more efficient).

Matlab's interp2() function implements bilinear interpolation (among other interpolation methods).

Bibliography

[1] Martin Berger. The framework of least squares template matching. Technical Report BIWI-TR-180, Swiss Federal Institute of Technology, Communication Technology Lab, Image Science, 1998.

[2] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. *IEEE Conference on Computer Vision and Pattern Recognition*, June 1998.

[3] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *European Conf. on Computer Vision, ECCV'96*, pages 329–342. Springer-Verlag, April 1996.

[4] Michael J. Black and Allan. D. Jepson. A probabilistic framework for matching temporal trajectories: CONDENSATION-based recognition of gestures and expressions. In H. Burkhardt and B. Neumann, editors, *European Conf. on Computer Vision, ECCV-98*, volume 1406 of *LNCS-Series*, pages 909–924, Freiburg, Germany, 1998. Springer-Verlag.

[5] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992.

[6] Roberto Brunelli and Tomaso Poggio. Template matching: Matched spatial filters and beyond. Technical Report AIM-1549, MIT Artificial Intelligence Laboratory, October 1995.

[7] Robert T. Collins, Alan J. Lipton, Takeo Kanade, Hironobu Fuiyoshi, David Duggins, Yanghai Tsin, David Tolliver, Nobuyoshi Enomoto, Osamu Hasegawa, Peter Burt, and Lambert Wixson. A system for video surveillance and monitoring. Technical report, CMU, 2000.

[8] Ingemar J. Cox. and Sunita L. Hingorani. An efficient implementation and evaluation of Reid's multiple hypothesis tracking algorithm for visual tracking. In *IEEE Trans. on PAMI*, volume 18:2, pages 138–150, 1996.

[9] Ingemar J. Cox and John. J. Leonard. Modelling a dynamic environment using a bayesian multiple hypothesis approach. *Aritificial Intelligence*, 66:311–344, 1994.

[10] Ingemar J. Cox and Matt L. Miller. On finding ranked assignments with applications to multi-target tracking and motion correspondence. *IEEE Trans. AES*, 32(1):486–489, 1995. Copyright (c) 1995 by IEEE.

[11] Ingemar J. Cox, Matt L. Miller, R. Danchick, and G.E. Newnam. A comparison of two algorithms for determining ranked assignments with application to multi-target tracking and motion correspondence. *IEEE Trans. on Aerospace and Electronic Systems*, 33(1):295–301, January 1997.

[12] Frank Dellaert, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Using the CONDENSATION algorithm for robust, vision-based mobile robot localization, 1999.

[13] Ahmed M. Elgammal and Larry S. Davis. Probabilistic framework for segmenting people under occlusion. In *Eighth IEEE International Conference on Computer Vision*, pages 145–152, 2001. http://citeseer.nj.nec.com/elgammal01probabilistic.html.

[14] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins, 3rd edition, 1996.

[15] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision*, pages 343–356, 1996.

[16] Michael Isard and Andrew Blake. CONDENSATION – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[17] Michael Isard and Andrew Blake. A mixed-state condensation tracker with automatic model-switching. *Proc 6th Int. Conf. Computer Vision*, pages 107–112, 1998.

[18] Allan Jepson, David Fleet, and Thomas El-Maraghi. Robust online appearance models for visual tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, I:415–422, 2001.

[19] Shanon. X. Ju, Michael J. Black, and Yaser Yacoob. Cardboard people: A parameterized model of articulated motion. In *International Conference on Automatic Face and Gesture Recognition*, pages 38–44, Killington, Vermont, 1996.

[20] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Image Understanding Workshop*, pages 121–130, 1981. http://citeseer.nj.nec.com/lucas81iterative.html.

[21] John MacCormick and Andrew Blake. A probabilistic exclusion principle for tracking multiple objects. *Proc Int. Conf. Computer Vision*, pages 572–578, 1999.

[22] Peter Meer, Doron Mintz, and Azriel Rosenfeld. Robust regression methods for computer vision: A review. *International Journal of Computer Vision*, 6(1):59–70, 1991.

[23] Matt L. Miller, Harold S. Stone, and Ingemar J. Cox. Optimizing Murty's ranked assignment method. *IEEE Trans. on Aerospace and Electronic Systems.*, 1995.

[24] Vishvjit S. Nalwa. *A Guided Tour Of Computer Vision*. AT&T, 1993.

[25] William H. Press, Saul A Teukolsky, WIlliam T. Vettering, and Brian P. Flannery. *Numerical Recipies in C*. Cambridge University Press, 2nd edition, 1997.

[26] Donald B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions On Automatic Control*, AC-24(6):843–854, December 1979.

[27] Azriel Rosenfeld and Gorden Vanderbrug. Coarse-fine template matching. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 104–107, February 1977. SMC-7.

[28] Qi Tian and Michael Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics, and Image Processing*, 35:220–233, 1986.

[29] Emanuele Trucco and Alessandro Verri. *Introductory Techniques For 3-D Computer Vision*. Prentice Hall, 1998.

[30] Gordon Vanderbrug and Azriel Rosenfled. Two-stage template matching. *IEEE Transactions on Computers*, C-26(4):384–393, April 1977.

[31] Greg Welch and Gary Bishop. An introduction to the Kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, November 30,1999.

[32] Christopher R. Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.