A Review of Visual Tracking

Kevin Cannons

Technical Report CSE-2008-07

September 16, 2008

Department of Computer Science and Engineering

4700 Keele Street Toronto, Ontario M3J 1P3 Canada

A Review of Visual Tracking

Kevin Cannons

Department of Computer Science and Engineering
and the Centre for Vision Research
York University
Toronto, Ontario M3J 1P3
Canada

September 16, 2008

Abstract

This report contains a review of visual tracking in monocular video sequences. For the purpose of this review, the majority of the visual trackers in the literature are divided into three tracking categories: discrete feature trackers, contour trackers, and region-based trackers. This categorization was performed based on the features used and the algorithms employed by the various visual trackers. The first class of trackers represents targets as discrete features (e.g. points, sets of points, lines) and performs data association using a distance metric that accommodates the particular feature. Contour trackers provide precise outlines of the target boundaries, meaning that they must not only uncover the position of the target, but its shape as well. Contour trackers often make use of gradient edge information during the tracking process. Region trackers represent the target with area-based descriptors that define its support and attempt to locate the image region in the current frame that best matches an object template. Trackers that are not in agreement with the abovementioned categorization, including those that combine methods from the three defined classes, are also considered in this review. In addition to categorizing and describing the various visual trackers in the literature, this review also provides a commentary on the current state of the field as well as a comparative analysis of the various approaches. The paper concludes with an outline of open problems in visual tracking.

Contents

Chapter 1

Introduction

This paper contains a review of the rich set of literature that analyzes the problem of monocular visual tracking. It would not be practical to attempt to describe all published works concerning visual tracking. The field has grown so quickly over the past twenty years that compiling such a list, let alone describing the various works, would be an extremely daunting task! In this review, we have elected to take a different approach. We have identified what we feel are some of the most noteworthy, impressive, and unique papers amongst the visual tracking literature. The primary focus of this review is on the visual tracking of general targets of interest in monocular videos obtained from charge-coupled device (CCD) cameras. Although particular references will be made where appropriate, this paper does not attempt to summarize the field of tracking across multiple cameras. Additionally, we do not attempt to summarize trackers that are overly specialized to particular applications (e.g., human motion analysis). Thorough reviews of human-specific trackers can be found in (Aggarwal & Cai, 1999; Gavrila, 1999; Moselund & Granum, 2001). Finally, it should be emphasized that motion in computer vision is a large field that spans a number of subproblems. Here, the focus is on the motion-related problem of visual tracking.

In addition to identifying and describing some of the most influential works in the field of visual tracking, we have also successfully divided the vast majority of these papers into three distinct categories of trackers. For each tracking category, we typically define the general method used by most of the related systems. Furthermore, we provide detailed descriptions of the unique methods and contributions of the most critical papers. This review also provides a comparative evaluation of the various categories of visual trackers and discusses some of the significant open problems in the field.

1.1 A Brief History of Visual Tracking

The field of target tracking extends back a number of years, far beyond the widespread adoption of the personal computer. To our knowledge, some of the earliest works on target tracking appeared in the 1950s (Wax, 1955), 1960s (Kalman, 1960; Sittler, 1964), and 1970s (Singer & Stein, 1971). The earliest works were primarily concerned with the application of radio detection and ranging (RADAR) signal detection and automated tracking. In this context, the problem of tracking is reduced to establishing correlations between point-wise RADAR "blips". On the surface, this problem appears much easier than that of tracking targets of arbitrary size, shape, and appearance in pictorial video sequences. However, one should not be fooled by the appearance of simplicity of the RADAR tracking problem. When multiple targets are moving in close proximity in the presence of noise, the RADAR target tracking problem is far from trivial. In fact, many of the most popular algorithms upon which computer vision trackers are based (e.g., the Kalman filter (Kalman, 1960)), were derived for the purpose of RADAR tracking. Solutions to the RADAR tracking problem existed (Wax, 1955; Sittler, 1964) prior to the widespread adoption of the Kalman filter; however, not unlike its effect on computer vision tracking, the Kalman filter spurred numerous subsequent works in the field of RADAR tracking.

Given its RADAR tracking roots along with the limited computational resources at the time, point tracking is considered one of the earliest examples of computer vision tracking (Sethi & Jain, 1987; Salari & Sethi, 1990; Rangarajan & Shah, 1991; Veenman, Reinders & Backer, 2001; Shafique & Shah, 2003). It seems that the majority of research surrounding this category of tracker appeared in the late 1980s to early 1990s. Apparently after this time, the attention shifted to the visual tracking of higher level objects rather than just simple points. Nonetheless, more recently these point trackers have seen somewhat of a resurgence. Nowadays, higher level objects can be followed by using point-based techniques to track collections of interest points. Furthermore, other application areas have made use of point trackers, including the tracking of flocks of birds and cells under a microscope. Finally, it should be noted that statistical methods for tracking multiple targets are continually being developed and refined (Fortmann, Bar-Shalom & Scheffe, 1983) and used in RADAR and computer vision problems.

In addition to point tracking, other discrete features have been used for visual tracking, including groups of edges and lines. Many trackers that rely on these discrete features leverage additional information from other sources, such as 3D object models (Verghese, Gale & Dyer, 1990; Koller, Daniilidis & Nagel, 1993; Lowe, 1991, 1992; Gennery, 1992; Vacchetti, Lepetit & Fua, 2004). However, the use of object models is not mandatory when using discrete features (Crowley, Stelmaszyk & Discours, 1988; Deriche & Faugeras, 1991; Zhang & Faugeras, 1992; Huttenlocher, Noh & Rucklidge, 1993). This tracking paradigm also appears to have been most active in the late 1980s and early 1990s. The use of 3D models has seen somewhat of a comeback in recent years, especially for the purpose of tracking humans (e.g., (Hogg, 1983; Munkelt, Ridder, Hansel & Hafner, 1998; Delamarre & Faugeras, 1999; Moeslund & Granum, 2000)). Nonetheless, it appears that the field has shifted its primary focus toward other tracking techniques when it comes to following arbitrary scene objects.

Another very different approach to visual tracking that saw its seminal publications appear in the late 1980s is that of tracking the outer contour of a target (Kass, Witkin & Terzopoulos, 1988; Terzopoulos & Szeliski, 1992; Leymarie & Levine, 1993; Caselles, Kimmel & Sapiro, 1995; Paragios & Deriche, 2000; Bertalmio, Sapiro & Randall, 2000). In a similar manner to the aforementioned discrete methods, contour-based trackers also gained significant attention upon initial inception, but received slightly less research in the mid-1990s. However, with the introduction of level set methods (Osher & Fedkiw, 2003b,a) combined with additional research regarding the incorporation of regional information within this tracking framework, contour-based trackers have seen considerable levels of activity in recent years (e.g., (Mansouri, 2002; Yilmaz, Lin & Shah, 2004; Li, Miller, Weiss, Campbell & Kanade, 2006)).

Another approach to the visual tracking problem is to represent the target as an entire region within the image. These trackers tend not to provide as precise of an outline as seen with contour trackers. However, with an increased target support, the hypothesis is that tracking can be made more reliable. Systems of this class often represent targets as blobs (Wren, Azarbayejani, Darrell & Pentland, 1997; Intille, Davis & Bobick, 1997; Rosales & Sclaroff, 1999; Stauffer & Grimson, 2000; Yin & Collins, 2007a) or using pixel-wise templates (Lucas & Kanade, 1981; Bergen, Anandan, Hanna & Hingorani, 1992; Black & Anandan, 1996; Black & Jepson, 1998; Hager & Belhumeur, 1998; Beymer & Konolige, 1999; Cham & Rehg, 1999; Olson, 2000; Nguyen, Worring & Boomgaard, 2001; Avidan, 2001; Jepson, Fleet & El-Maraghi, 2003; Enzweiler, Wildes & Herpers, 2005; Wong & Spetsakis, 2006). Kernel histogram methods have also become extremely popular in the last eight years (Comaniciu, Ramesh & Meer, 2000, 2003; Collins, 2003; Collins & Liu, 2003; Hager, Dewan & Stewart, 2004; Birchfield & Rangarajan, 2005; Leung & Gong, 2006; Cannons & Wildes, 2007). For example, the mean shift tracker (Comaniciu, Ramesh & Meer, 2003), is arguably the most popular tracking paradigm, at present.

Upon viewing this rough timeline, we elected to group our discussion of visual tracking into three main chapters concerning: (1) discrete feature trackers; (2) contour trackers; and (3) region trackers. Similar categorization of visual tracking systems has appeared in other review papers (e.g., (Yilmaz, Javed & Shah, 2006)).

1.2 Motion-Based Computer Vision Problems

It is important to take note that within the vast field of computer vision, there are other problems than visual tracking that make use of motion information. These additional subfields include: optical flow estimation; regional motion analysis; structure from motion; and motion/change detection. Since these other motion-related problems will not be the main topic of discussion in subsequent chapters, we will make brief comments about them now.

The problem of computing optical flow can be described as that of estimating the apparent motion of a scene at an infinitesimally small space-time location. This observed motion can arise due to the motion of an object within the scene, motion of the camera, or apparent motion caused by changes in the environment (e.g., due to changing specularity position caused by the movement of the sun). Typically, the so-called brightness constancy constraint (BCC) is used when computing optical flow. This constraint implies that the intensity of a pixel will remain unchanged, even as it moves. Broadly speaking, there are three general approaches for computing optical flow (Derpanis, 2006): matching methods; differential methods; and frequency-based methods.

If we relax the constraint that only infinitesimally small regions of interest can be considered, we arrive at the next subfield: regional motion analysis. In this field, researchers once again wish to describe the motion and structure in visual space-time, but to do so over larger regions of support. Work in this field includes that of attempting to describe the motion of a region using a low-dimensional parametric model (e.g., (Lucas & Kanade, 1981; Bergen, Anandan, Hanna & Hingorani, 1992; Black & Anandan, 1996)). Another area of focus is the computation of motion layers within a scene (e.g., (Ayer & Sawhney, 1995; Wang & Adelson, 1993; Wong & Spetsakis, 2006)). One can think of computing motion layers as segmenting a video sequence into coherently moving regions. Finally, other researchers have considered the problem of assigning qualitative descriptors to local spatiotemporal regions (e.g., (Jähne, 1990; Bigün, Granlund & Wiklund, 1991; Wildes & Bergen, 2000)). These regional descriptors can encompass spatial patterns (e.g., spatial texture) as well as dynamic aspects (e.g., velocity and flicker). An excellent review of the fields of optical flow estimation and regional motion analysis is provided in (Derpanis, 2006). Furthermore, it should be noted that optical flow and regional motion descriptors are often employed within tracking systems and will be discussed in more detail in this paper, as appropriate.

Another motion-based problem within computer vision is that of obtaining 3D scene structural information from motion information. It is not surprising that structure from motion is one of oldest areas of computer vision research because initially, many researchers viewed the computer vision problem as that of reconstructing a 3D scene from 2D images. As a result, the problems of structure from "X" where X is motion, shading, and stereo are well-established within computer vision. In general, the approaches of obtaining scene structure from motion can be categorized into three disparate approaches: finite displacement; infinitesimal; and direct methods.

A final subfield of computer vision that involves motion information is that of change/motion detection (Anderson, Burt & van der Wal, 1985; Liou & Jain, 1989; Lipton, Fujiyoshi & Patil, 1998; Dailey & Li, 1999; Lee, Romano & Stein, 2000; Chang, Chia & Yang, 2005). The goal of these systems is to determine what pixels in the scene are changing due to object motion. Most of these strategies are employed when the camera is stationary and the scene objects are moving. However, techniques are available for change/motion detection when both the camera and scene objects are moving (Burt, Bergen, Hingorani, Lee & Leung, 1989). Intuitively, the idea here is to realign the images prior to detecting the motion so as to cancel the camera-based motion. Motion detection systems are often used as a precursor or directly within tracking systems. Accordingly, change/motion detection will be discussed in more detail in Chapter 2.

1.3 Outline of Paper

The remainder of this paper is organized into six main chapters. In Chapter 2, we will begin with a discussion of the general tools that are used in a number of trackers. In Chapter 3, trackers that solely make use of discrete feature information will be described. Subsequently, Chapter 4 will focus on contour trackers. Systems that make use of regional target representations will be described in Chapter 5. In an ideal world, all tracking systems would correspond to one of the previous three chapters. However, given the nature of the literature, some systems use combinations of the techniques seen in Chapters 3 - 5, and others are just outright unique. These trackers will be described in Chapter 6. In Chapter 7, a more in-depth comparative evaluation between the different approaches will be presented. The paper will be concluded in Chapter 8 with a discussion of the open problems that remain in the field of visual tracking.

Chapter 2

Tools For Tracking

2.1 Initialization and Foreground Detection

A significant consideration in the field of visual tracking is that of starting, or initializing, a tracker. Visual trackers that require less user input from a human are more practical and more likely to be deployed outside of academia. Yet, as demonstrated by the research in the field, the initialization problem is often ignored. Instead, it is commonly assumed that initialization can either be performed manually or by some other system module (Kass, Witkin & Terzopoulos, 1988; Fieguth & Terzopoulos, 1997; Jepson, Fleet & El-Maraghi, 2003; Cannons & Wildes, 2007). Of course, there are exceptions (Stauffer & Grimson, 2000; Haritaoglu, Harwood & Davis, 2000; Paragios & Deriche, 2000; Enzweiler, Wildes & Herpers, 2005). Automated systems often use some form of foreground detection mechanisms to identify image region(s) to be tracked. Hence, the matters of tracker initialization and foreground detection are intertwined. However, at this point it should be emphasized that foreground detection need not only be used for the purpose of tracker initialization. Foreground detection can be used throughout the course of tracking and in fact, many trackers perform this exact procedure (Intille, Davis & Bobick, 1997; Wren, Azarbayejani, Darrell & Pentland, 1997; Rosales & Sclaroff, 1999; Stauffer & Grimson, 2000; Haritaoglu, Harwood & Davis, 2000; Isard & MacCormick, 2001).

2.1.1 Tracker Initialization

As mentioned previously, target identification can be performed manually or automatically. Many significant tracking papers simply assume that the targets can be identified by some other mechanism (e.g., human input). This assumption is typically made so that the focus of the work can be placed solely on the task of tracking the marked target through subsequent frames. When manual initialization is used, it is often assumed that all targets are present in the first frame of the video sequence.

In other words, no new targets of interest enter the field of view during the video. Accordingly, in the first frame, a user will simply draw boxes, ellipses, curves, etc. that encompass the targets and automatic tracking will be performed in subsequent frames.

Other researchers do consider the task of initializing trackers in an automated fashion. Aside from the obvious benefit of being more automated, an additional advantage is the fact that it is easier to allow new targets to be identified and tracked if they enter the field of view after the first frame. Most automatic systems initialize a tracker using some form of foreground detection technique. Once distinct foreground regions have been identified, trackers can be initialized to follow these regions.

2.1.2 Foreground Detection

In the context of this paper, "foreground detection" refers to the process of identifying potential target pixels in any frame of a video sequence, regardless of frame number. Pixels that do not belong to the target(s) or other "interesting" (i.e. coherently moving) objects will be termed "background pixels". Most foreground detection techniques can be divided into three main categories: background modeling approaches (Wren, Azarbayejani, Darrell & Pentland, 1997; Stauffer & Grimson, 2000; Haritaoglu, Harwood & Davis, 2000; Rittscher, Kato, Joga & Blake, 2000); change/motion detection approaches (Anderson, Burt & van der Wal, 1985; Lipton, Fujiyoshi & Patil, 1998; Dailey & Li, 1999; Lee, Romano & Stein, 2000; Chang, Chia & Yang, 2005; Enzweiler, Wildes & Herpers, 2005); and target spotting approaches (Viola & Jones, 2001; Zhu, Zhao & Lu, 2007; Wu, Yu & Hua, 2005). Additionally, some foreground detection schemes operate by simply identifying pixels of a particular color that is unique to the target (e.g., skin color detection (Kjeldsen & Kender, 1996; Fleck, Forsyth & Bregler, 1996; Herpers, Derpanis, MacLean, Verghese, Jenkin, Milios, Jepson & Tsotsos, 2001)). For the purpose of this review, color detection approaches will be considered as a basic form of target spotting. The rest of this section will describe the three proposed foreground detection categories in more detail.

The most straightforward background modeling approaches assume that the background scenery remains stationary. Thus, to detect a foreground target, the current frame can essentially be subtracted from the background model. Pixels that are in agreement with the background model will yield a near-zero result while pixels that deviate substantially from zero are concluded to belong to the foreground. The dividing line between "near-zero" and "deviating substantially" can be implemented by means of a simple threshold. Performing this type of raw subtraction and thresholding will yield noisy results. Correspondingly, morphological operators (Heijmans, 1994) are used to blur foreground pixels into blobs and then group these blobs into connected components.

An example of using background modeling to identify foreground regions is illustrated in (Wren, Azarbayejani, Darrell & Pentland, 1997). The authors model each pixel as a multi-variate Gaussian in YUV color space, capturing a pixel's mean color and covariance. In (Stauffer & Grimson, 2000), a slightly more complex approach is taken where each pixel is represented as a mixture of Gaussians (MOGs). The rationale is that even though a pixel may be part of the background scenery, it may still be generated by a multi-modal distribution. For example, a pixel that lies on a flashing light that oscillates between the colors red and black could be considered as part of the background scenery. Modeling this type of background pixel with a single Gaussian would be very inaccurate. However, with a mixture of Gaussians-based approach, different colors correspond to different modes within the distribution. Thus, a more accurate parametric model of the background can be obtained. Other backgrounds that would be better modeled using MOGs approach include an oscillating fan and trees swaying in the wind. Naturally, when modeling each background pixel as a mixture of Gaussians, an additional parameter is introduced — the number of Gaussians. Another disadvantage of methods inspired by (Stauffer & Grimson, 2000) is that they are more computationally intense than unimodal methods.

Foreground detection can also be performed using change/motion detection-based approaches. With background modeling approaches, it is assumed that anything different from the background model is "interesting" and is part of the foreground. Such objects may be moving quickly, slowly, or could even be stationary for a certain amount of time (if the object is stationary for too long, the background model may adapt to encompass the object). Change/motion detection techniques employ a slightly modified assumption: Interesting targets are manifest as pixels that exhibit a change in intensity from frame-to-frame, especially due to motion. A range of techniques have been developed for identifying image regions undergoing changes due to motion, ranging from temporal differencing (Anderson, Burt & van der Wal, 1985) to measuring the motion coherence strength of a pixel (Enzweiler, Wildes & Herpers, 2005).

The identification of moving targets through temporal differencing is a straightforward process. Specifically, the current image is subtracted from either the previous (Anderson, Burt & van der Wal, 1985; Lipton, Fujiyoshi & Patil, 1998; Lee, Romano & Stein, 2000; Chang, Chia & Yang, 2005) or next frame of the video. Rectification through squaring or taking the absolute value can be performed to consider only magnitude differences. Mathematically, such a temporal differencing methodology can be written as

$$D_b\left(x,y,t\right) = \left(I\left(x,y,t\right) - I\left(x,y,t-1\right)\right)^2, \tag{2.1}$$

where I is the image sequence and D_b represents the squared temporal difference image. Thresholds are applied to the difference images to eliminate pixel changes due to camera noise, small illumination changes, etc. Thus, the end result is a binary map of the foreground pixels.

To improve upon the localization of the foreground moving pixels, three frames can be used to compute the change image (Dailey & Li, 1999). The rationale is that backward and forward differences are inherently biased to be slightly behind or ahead of the true moving target in the current frame. By pooling additional information over three frames, the detected target can be better centered. Three-frame temporal differencing can be performed by computing the forward and backward differences separately and then combining the results. Another differencing-based approach that accumulates information over several frames was presented in (Jain, 1984). In this case, the current frame of the image sequence is continually compared against a reference image (typically the first frame). Thresholded positive (indicates intensity values of the reference image that are greater than the current image) and negative (indicates intensity values of the current image that are greater than the reference image) difference images are computed and accumulated over a series of frames. Combination of the positive and negative difference images allows for the identification of moving objects and the segmentation of dynamic scenes.

The standard temporal differencing approach appears to be straightforward enough, but a limitation one might immediately notice is that it appears to be confined for use with only stationary cameras. Using the techniques proposed so far, this assessment is certainly true. However, techniques are available for the purpose of stabilizing the background when dealing with moving cameras (e.g., (Burt, Bergen, Hingorani, Lee & Leung, 1989; Bergen, Burt, Hingorani & Peleg, 1990; Black & Anandan, 1996)). In (Bergen, Burt, Hingorani & Peleg, 1990), methods are shown for estimating local optical flow when transparency is present. The problem is reduced to that of computing multiple motions at the same spatial location. The authors show that a secondary motion can be uncovered by first aligning the images via the dominant recovered motion and subsequently computing a difference image. This technique can be applied in the context of visual tracking with a moving camera. Essentially, the global camera motion can be estimated and subsequently, difference images can be created for the target by considering the aligned imagery. Black et al.'s robust multiple motion estimator could be deployed in a similar manner (Black & Anandan, 1996). In (Burt, Bergen, Hingorani, Lee & Leung, 1989), a concrete example of "subtracting off" the camera motion for the purpose of visual tracking is demonstrated.

One final aspect of temporal differencing is that it is often performed in conjunction with pyramid processing (Anderson, Burt & van der Wal, 1985). If difference images are computed at full image resolution and the video framerate is high, the change detection region will include only the outline of the moving object. If the interior of the object is desired, a Gaussian or Laplacian pyramid can be used. The general approach here is to first compute the difference image. Once computed, the rectified difference image is downsampled and upsampled via an image pyramid to fill in the interior. Image pyramid processing is a topic that will reoccur throughout this paper and will be discussed in more detail in Section 2.5.

Another approach to change detection uses motion coherence measures (Enzweiler, Wildes & Herpers, 2005). In (Enzweiler, Wildes & Herpers, 2005) so-called "coherent motion energy" is computed by filtering space-time slices of a video sequences and subsequently rectifying and normalizing the result. Leftward, rightward, upward, and downward motion create characteristic diagonal structures when observed using XT and YT slices. Orientation-selective filtering is performed on these slices in directions corresponding to these four motions. Coherent motion energies are obtained by taking the difference between the energies of the opponent directions and normalizing. This detection process is performed across multiple scales using pyramid processing to ensure that the detected target of interest has support across a range of frequencies. Although this approach to motion/change detection is more complex than straightforward temporal differencing, there are several advantages to be had. To begin, the coherent motion energy-based approach is more robust to illumination changes because of the performed normalization and the broadly-tuned nature of the filters. Furthermore, the target pixels must be moving coherently in order to be detected. Thus, in a similar manner to the MOGs background models, change detection systems using coherent motion energies should not generate false-positive pixels due to swaying trees or oscillating fans. A simpler temporal differencing algorithm would undoubtedly label such pixels as belonging to the foreground.

The final general class of foreground detection schemes are known as target spotting systems. In this case, a module is typically trained to detect a certain class of object, such as a face (Viola & Jones, 2001), a vehicle (Zhu, Zhao & Lu, 2007), a pedestrian (Wu, Yu & Hua, 2005), a hockey player (Okuma, Taleghani, Freitas, Little & Lowe, 2004), etc. Other, more simple target spotting approaches detect foreground targets based purely on pixel color information (Kjeldsen & Kender, 1996; Fleck, Forsyth & Bregler, 1996; Herpers, Derpanis, MacLean, Verghese, Jenkin, Milios, Jepson & Tsotsos, 2001). Once a specialized detector has been developed, it can be applied to frames in a video sequence. A critical limitation of this approach is computation time. If the target detector is run on every frame of the video, it must be very efficient. This need for efficiency has contributed to the popularity of Adaboost-based detectors (Okuma, Taleghani, Freitas, Little & Lowe, 2004; Viola & Jones, 2001). Other popular machine learning classifiers such as support vector machines (Burges, 1998), neural networks (Bishop, 1995), etc. could also be used for detection, as long as they can be implemented in a computationally efficient manner. One method to alleviate computational requirements is to only perform detection on a subset of the video frames (e.g., every 30^{th} frame). However, this technique could lead to delayed detection results.

There are two questions to consider at this juncture: 1) Why would foreground detection be needed during tracking for any reason other than tracker initialization? 2) Can a tracker be built that will follow objects by repetitively performing foreground detection on each and every frame? These questions are almost polar opposites, but they are both worth considering.

Foreground detection can be greatly beneficial to trackers, as it helps to reduce the search space when looking for a target. Specifically, rather than having to search the entire image for the new position of the target, only the foreground regions need to be analyzed (assuming the target is always moving). Such a reduction can greatly improve the speed and accuracy of tracking. Thus, foreground detection can clearly be useful. However, we should also point out that under certain assumptions, global image searches for targets of interest are continually evolving and are approaching the speeds required for real-time tracking applications. For example, the integral histogram method is empirically argued to provide real time exhaustive image search capabilities, although its memory requirements are large (Porikli, 2005). More recently, Sizintsev et al. compared the relative merits of various popular histogram-based search strategies and proposed a so-called "distributed histogram". Distributive histograms are shown to be faster and require less memory than integral histograms (Sizintsev, Derpanis & Hogue, 2008). The one significant limitation of these techniques is that they require the target to be represented as a histogram. Nonetheless, we are already beginning to see these exhaustive histogram-based search strategies employed in numerous visual tracking systems (e.g., (Adam, Rivlin & Shimshoni, 2006; Nejhum, Ho & Yang, 2008)).

Given that foreground detection can be useful, naturally, the next question is: why not build a system that simply tracks objects by performing foreground detection? In some limited situations, this approach may be sufficient. Specifically, if it is known that there is only one target in the video and that the background is roughly stationary, then tracking by continuous detection is possible. However, the detection mechanism must operate within the framerate of the video. In realistic scenarios, however, we do not always have the luxury of tracking an isolated, single target in a scene. Oftentimes there are multiple moving targets. In these situations, there will be ambiguity when attempting to match foreground regions from frame-to-frame if continuous detection is employed. The issue of properly linking a detected target in the previous frame with the same target in the current frame is known as "data association", a topic that will be discussed in more detail in Section 2.4. Thus, although it may be possible in some limited situations to perform visual tracking purely using foreground detection techniques, in most cases, at a bare minimum the data association problem must be resolved for successful tracking. Many trackers in the literature have been implemented that follow the foreground detection plus data association framework and they will be discussed primarily in Chapter 5.

2.2 Feature Extraction

A critical component of intelligent systems including recognition, detection, and tracking systems is that of feature extraction. For the purpose of this document, a feature is defined as a local, meaningful representation of a target or part of a target. Furthermore, we will refer to a "feature set" as the one or more features used to describe a target. A feature set should not be confused with a "target template". A template describes the appearance of a target, regardless of the features that are being used. For example, a very simple feature set would be the pixel-wise intensities of a target. Alternatively, the individual pixel values could be considered following application of spatial and/or temporal filters (e.g., bandpass filters that select certain frequency content). On the other hand, the target template specifies the feature values assumed by the target that, when combined, define the appearance of the target. More specifically, a template might specify that the top row of a four-pixel wide target is defined by the intensity values $(128, 120, 110, 100)$. In this example, the feature set being used are pixel-wise intensities, and the specific numeric values define the template.

One of the most important aspects of a feature is that it should be unique to the target and meaningful for the particular application under consideration. For example, in target tracking a feature set is desired that will uniquely identify the target of interest from other scene objects. For instance, a poor choice of feature when tracking a specific zebra that is moving amongst many other zebra would be color. Since all zebras have roughly the same white and black pattern, a tracker that uses color-based features would be challenged to follow the zebra of interest. Thus, for successful tracking, the feature set must be descriptive; however, it must also be flexible. A target may deform, rotate, change in illumination, and change in scale. Trackers are desired that can continue following a target through such changes. Indeed, feature sets that are robust or invariant to these transformations are sought after. A final critical requirement for feature sets is that they must be amenable to efficient extraction because most trackers strive to operate in real-time.

In this section, different types of features that have been used in various tracking systems will be described. The discussion will begin with discrete features such as edges and lines. Recently, "interest points" have emerged as a useful type of feature, and they will also be described. Additionally, we will discuss color-based features that have become very popular in the last fifteen years. Finally, we will elaborate on feature sets that are based on oriented energies.

2.2.1 Edges and Lines

Discrete features such as edges and lines have been used in many tracking systems (Deriche & Faugeras, 1991; Lowe, 1991, 1992; Koller, Daniilidis & Nagel, 1993; Huttenlocher, Noh & Rucklidge, 1993). In fact, not that long ago, the majority of computer vision systems employed a bottom-up approach. Specifically, processing would begin by first computing the edge map of an image. All subsequent processing was based entirely on this edge information. The rationale behind these bottom-up approaches is that edge detection is like a form of dimensionality reduction. Specifically, it was believed that an edge map contained the most important image information (e.g., object boundaries), while suppressing uninteresting regions (e.g., regions of constant intensity). This assumption may hold true in some cases, but there are severe disadvantages of this approach. By progressing using only edge features, significant, potentially useful information is discarded (e.g. color, irregular texture patterns). Furthermore, edge detection is a non-trivial exercise. Fragmented object contours and false positives in cluttered scenes are common problems. Accordingly, these types of bottom-up approaches are currently becoming less popular.

To begin, we will discuss edge based features that have been used in trackers. For this paper, an "edge" is defined as a pixel that lies on the boundary between regions that each have their own (roughly constant) intensity. Image edges can arise due to boundaries between objects, large illumination differences (e.g., shadows), surface markings, textures, as well as large changes in surface orientation. There are many edge detectors available, including the Canny (Canny, 1986), Nalwa (Nalwa & Binford, 1986), Iverson (Iverson & Zucker, 1995), and Bergholm (Bergholm, 1987) algorithms. A difficulty in the field of edge detection is that of fair experimental evaluation. Performance analysis is complicated by the fact that different edge detectors tend to perform best on certain types of imagery. Furthermore, the quality of the extracted edge set is somewhat application-dependent. For some systems, false positives will be severely detrimental, while other systems may be most adversely affected by false negatives. These difficulties complicate the comparative process. In (Heath, Sarkar, Sanocki & Bowyer, 1997), various edge detectors are evaluated by means of unbiased human observers. Among the detectors compared in this study are Canny's edge detector and various other state-of-the art methods at the time of publication (1997). The authors conclude that Canny's algorithm is among the best when image-wise parameter optimization can be performed. However, when parameters are optimized for a large array of images, the relative performance of the Canny edge detector decreases.

Perhaps because of the difficulties in clearly demonstrating superior performance by newer methods (accuracy and computation time), the most standard edge detector in computer vision today is Canny's edge detector (Canny, 1986). The Canny edge detector is a gradient-based edge detector. Accordingly, one of the main components of the algorithm is the computation of the vertical and horizontal image gradients. Using the magnitudes and directions of the image gradients, Canny's algorithm then identifies the most likely edge pixels using non-maximal suppression and thresholding stages. The interested reader is encouraged to read Canny's original paper (Canny, 1986) or a textbook presentation of Canny's work (Trucco & Verri, 1998) to learn

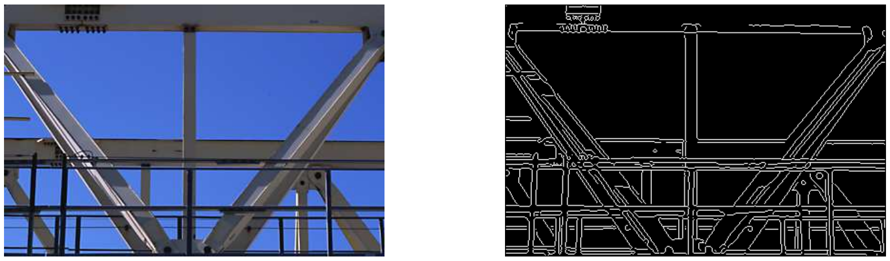


Figure 2.1: Input image (left) and binary edge map obtained using Canny edge detection (right).

more details about the algorithm. In Fig. 2.1 we provide a pictorial example of a binary edge map obtained using the Canny edge detector.

Although edges by themselves are sometimes used for tracking (Huttenlocher, Noh & Rucklidge, 1993), oftentimes edges will be further grouped into lines (e.g., (Crowley, Stelmaszyk & Discours, 1988)). One could imagine that a very simple, intuitive way to detect image lines would be to employ a filter bank approach. Under this setup, a bank of filters would be defined that correspond to lines with the orientations that are to be detected. Image convolution can be performed between the bank of oriented filters and the image. Subsequently, the response maps could be combined and thresholded. Such a "quick and dirty" approach might work in limited situations, but most line detectors in the literature provide more advanced solutions.

In (Steger, 1998), a short summary of line detectors is provided. Steger argues that most line detectors can be grouped into three classes. The first category, makes use of purely local processing of intensity values and is arguably the most closely related to the simplified approach proposed above. For example, in (Geman & Jedynak, 1996), spatially oriented filters are used and differences are obtained between points that are hypothesized to be on the line and off the line. The rationale is that two points that lie on the line should have much more similar intensities than one point that is on the line and another that is off. As this approach is prone to false positives, more advanced solutions have been proposed. The second category of line tracker assumes that lines are composed of two parallel edges (due to a thickness greater than a single pixel). Correspondingly, two parallel line detectors are used to locate the left and right sides of the line simultaneously (Koller, Gerig, Szekely & Dettwiler, 1995). The third category of line trackers view an image as a three-dimensional surface and then attempt to locate ridges or ravines within the surface (Maintz, van den Elsen & Viergever, 1996).

Even though the abovementioned line detection techniques appear to be used in the realms of remote sensing and medical image analysis, the Hough transform (Hough, 1962; Duda & Hart, 1972) seems to be the more commonly-used method for detecting lines in computer vision. Although it is not necessary, the Hough transform

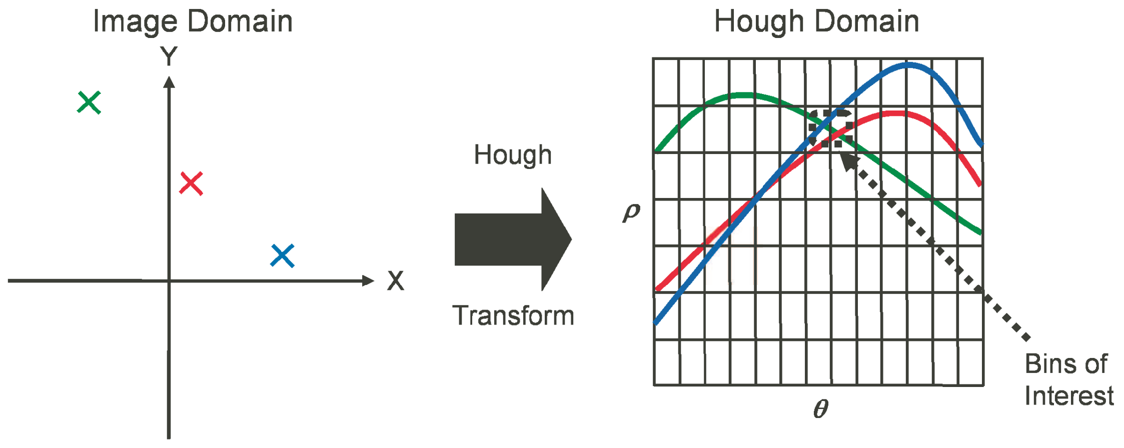


Figure 2.2: Hough transform mapping image edgels (image domain) to their corresponding sinusoidal curves (Hough domain).

typically employs a polar parameterization of lines. Specifically, a line is commonly defined in terms of its angle, α and its perpendicular distance from the origin, ρ. A "Hough domain" map is created, which is essentially a two dimensional histogram, as shown in Fig. 2.2. Using a binary edge map as input, the Hough domain map is computed as follows. Each edge pixel in the image domain is mapped to a sinusoidal curve in the Hough domain. This curve arises due to the fact that there are an infinite number of lines that could potentially pass through a particular image edge. Each bin that the edge's corresponding curve passes through has its histogram count incremented. This mapping from edge locations in the image domain to curves in the Hough domain is performed for all edges. When analyzing the complete Hough domain map, bins that have large sums have strong edge support. Thus, the matter of finding lines within the image becomes a problem of identifying the most prominent peaks in the Hough domain.

One limitation of the Hough transform is the fact that the identified lines are infinite in length. Thus, localization of the true line segment within the image is an additional problem. Furthermore, bin size selection is a challenging problem with conflicting requirements. On the other hand, the Hough transform is quite robust to noise edges. In fact, strictly speaking, the Hough transform is a robust estimator and thorough statistical analysis (Goldenshluger & Zeevi, 2004) has shown that its breakdown point is very close to 50%. Thus, despite its imperfections, the Hough transform remains one of the most widely-used straight line detection techniques today.

2.2.2 Corners and Interest Points

Another class of image features are commonly referred to as "interest points". For the purpose of this paper, we define an interest point as a point where the surrounding image region is changing in a two-dimensional manner. For example, a point at the center of a traditional "L-shaped" corner, as one would observe at the edge of a desk, satisfies this criteria. Other types of corners, such as "Y" or "T" shaped corners would also apply. However, interest points are not just limited to corner. For example, points within image regions containing distinctive texture patterns may also qualify as interest points. Generally, there are three classes of approaches for interest point detection (Schmid, Mohr & Bauckhage, 2000): contour-based; intensity-based; and parametric model-based. This paper will only focus on describing the intensity-based approaches as they appear to be the most prevalent in the tracking literature.

In general, intensity-based methods focus on the analysis of the local autocorrelation function around the point of interest. In (Moravec, 1980) a small window is placed around the point under analysis. Autocorrelation is then performed between the window centered on the point and slightly offset versions of the window. The offset windows are shifted in four discrete directions. If all autocorrelation values are greater than a threshold, the current point is classified as an interest point. The rationale is that if all auto-correlation values are large, then the image is significantly different from the analysis point as one moves in all four directions. Thus, the point under analysis is distinctive.

The autocorrelation-based approach is also used in other works such as (Shi & Tomasi, 1994) and the Harris corner detector (Harris & Stephens, 1988). In this case, however, the authors avoid the limitation of only analyzing the intensity changes in four discrete directions. Instead, a matrix is derived from the local auto-correlation function that captures the local intensity structure. This matrix can be written as

$$C(x,y) = \begin{bmatrix} \sum_W (I_x(x_i,y_i))^2 & \sum_W (I_x(x_i,y_i) I_y(x_i,y_i)) \\ \sum_W (I_x(x_i,y_i) I_y(x_i,y_i)) & \sum_W (I_y(x_i,y_i))^2 \end{bmatrix}, \qquad (2.2)$$

where W is a local window surrounding the point under analysis, and I_x, I_y are the image gradients in the x and y directions, respectively.

Once the matrix of Eq. 2.2 has been computed for a point, its eigenvalues are considered. If both eigenvalues are small, it indicates that the local region surrounding the current point is of approximately constant intensity. Situations where one eigenvalue is large, indicate that the local region is reminiscent of a valley. Specifically, moving along one axis will yield negligible changes in the intensity structure, whereas shifts along the other axis result in larger intensity changes. For the first two cases, the local window can move in one or more directions and still identify another location very similar to the original. Such locations are not ideal as interest points. The third case, where both eigenvalues are large, indicates that the autocorrelation function is highly peaked at the location under analysis. Corresponding, any shift will result in a significant change in intensity structure. Thus, it is only under this third case, that a point is labeled as an interest point. It should be noted that if using more than just a single interest point, the requirement that both eigenvalues must be large can be relaxed. For example, if one candidate point lies along a horizontal ridge and a second candidate point is on a vertical ridge, together these two points can provide similar constraints to that of a corner. A similar idea was employed in a region-based tracker where a constraint was added that ensures the smoothness of motion between neighboring edge interest points (Birchfield & Pundlik, 2008).

Another popular type of interest point used throughout the field of computer vision are scale invariant feature transform (SIFT) points presented by Lowe (Lowe, 1999). Feature point locations are taken as the minima or maxima of a difference of Gaussian function in scale space. The feature vectors are composed of location, scale, and orientation information regarding the point. Orientation information is obtained by selecting the peak orientation in a histogram of the local gradient orientations around the feature point. This canonical orientation makes the descriptors invariant to rotation. Furthermore, as demonstrated in (Lowe, 1999), SIFT features are robust to affine image transformations, illumination changes, and random pixel noise. Arguably the biggest advantage of SIFT features over other types of features (e.g., Harris corner features) is that they are scale-invariant. With the Harris corner detector, drastically different feature points can be identified when the same image is viewed at different scales. This phenomena occurs because a corner at one scale may not to be a corner at another scale. For example, at high spatial resolutions, textures on an object's surface may be visible, which may result in a number of interesting corners being detected. However, when the same object is observed with a lower spatial resolution, its surface may appear flat and textureless. Thus, at low resolutions, corners may not be detected on the same surface. Not being robust or invariant to changes in scale is a severe limitation of a feature point. SIFT features avoid this problem as they are identified through a scale-space analysis and are explicitly assigned a scale.

2.2.3 Regional Features

Features used for image processing and computer vision need not be discrete or point-based. Features that describe a larger image region have recently become very popular. Specifically, color histograms and histograms of gradients are two types of regional descriptors that are commonly found in the tracking literature.

Color features first started entering the main-stream tracking literature with the advent of the PFinder system (Wren, Azarbayejani, Darrell & Pentland, 1997). In this early work, the mean pixel color was used to describe the color of a so-called "blob region". Blob regions and trackers will be discussed in more detail in Chapter

5, but for the time being, a blob can be thought of as a connected foreground region that roughly defines a target. Subsequent works also made use of the mean color value of an image region. For example, in (Fieguth & Terzopoulos, 1997), a tracker was developed that represented a region of the target by its mean color. Comparisons with the target template were performed by taking the ratio between the candidate and template colors.

Although it is possible to represent the color of a region by a single average value, this metric becomes less meaningful when the color distribution is multimodal. For instance, if one were tracking a person's face (excluding hair) a single mean value representation should be sufficient — the predominant color within the region is the color of the skin (with some outliers for the eyes, lips, and teeth). On the other hand, if one is attempting to represent the entire human body by a single mean color, the representation begins to fail. If the individual being modeled is wearing a white shirt and blue pants, an average color representation is not very meaningful. Note that for the purpose of this review, "average color" is defined as the color obtained by computing the mean values of the respective R, G, and B channels over the target support. To address the abovementioned shortcoming, researchers have considered keeping a richer representation — the color distribution of an object. In particular, since discrete pixels are being considered, color histograms have become a popular representation (Swain & Ballard, 1991; Comaniciu, Ramesh & Meer, 2003). Color histograms are constructed by counting the number of times each discrete color occurs within an image region. Typically, the color-space will be quantized more coarsely than that used for the image itself. For example, the RGB color space (commonly used when representing images) uses 24-bits per pixel. This equates to 8-bits or 256 distinct values for each of the three RGB channels. When creating a color histogram, it is more common to use, for example, 4-bits per channel. This level of quantization would yield a histogram with $16 \times 16 \times 16 = 4096$ bins. Although it may seem like this loss of precision would be detrimental, it can actually be beneficial (Swain & Ballard, 1991). Object regions that appear to have the same color to the human eye will tend to not have constant pixel values due to illumination effects and camera noise. Under a more coarse quantization, pixels of similar color will be grouped together, even if they are not identical. A further advantage of using fewer bits is that the process is less memory intensive.

Color histograms have become extremely popular due to their simplicity, effectiveness, and efficiency. These histograms also have the desirable property that they are invariant to translation and rotation about the axis perpendicular to the image. Furthermore, color histograms change slowly when an object is rotating into the image plane, becoming occluded, or changing in scale (Swain & Ballard, 1991). Although color histograms have numerous advantages, they do have their limitations. To begin, color histograms (and histograms, in general) collapse the spatial information of the object being modeled. In other words, once the histogram has been created, there is no way knowing which color came from what part of the object. Some attempts have been made to retain limited spatial information within what is essentially a histogram framework (e.g., (Birchfield & Rangarajan, 2005; Adam, Rivlin & Shimshoni, 2006; Nejhum, Ho & Yang, 2008)).

Another limitation of color histograms is that color features are susceptible to illumination changes. There are partial solutions that have been presented that lessen these effects. The RGB color space (a color space is defined as the mathematic model that details the relationship between a color and a tuple of numbers) is the most prevalent in the literature but is also one of the least robust to changes in illumination. Other color spaces, such as normalized RGB as well as hue, saturation, and brightness (HSV) are more robust to illumination changes. Accordingly, many tracking systems that aim to follow targets in realistic surveillance settings have opted for these alternative color spaces (Hager, Dewan & Stewart, 2004). The one disadvantage of using an alternative color space is that there is a slight overhead involved to convert between RGB (color space typically used when storing images) to the alternative color space. Fortunately, these transformations are not overly expensive.

An alternative descriptor to color histograms are histograms of oriented gradients (HOGs). Although not explicitly mentioned, we have already briefly introduced HOGs when we discussed SIFT and the manner in which the orientation is selected for a feature point. HOGs are exactly as the name suggests — histograms that show the orientation distribution of the gradient vectors inside a target region. Thus, to construct an HOG, the image gradient is computed and then a count of the number of gradients that lie in each angular range is obtained. In addition to deciding on the method in which to compute the image gradient, another design decision is the coarseness of the histogram bins. Histograms of gradients have an advantage over color histograms in that they are more robust to illumination changes. However, in situations when the background is cluttered, gradients from background objects may adversely affect the target gradient model. As color and gradient histograms often excel under different experimental conditions, one can imagine, there is potential to fuse information from these two sources. The question then becomes: "how can these two sources of information be combined in a principled fashion?" This is a theme that will be explored throughout this paper.

A final commonly-employed regional feature representation is the so-called "Eigen" representation (Black & Jepson, 1998; Hager & Belhumeur, 1998). Eigen-based approaches attempt to minimize the dimensionality of a target by representing it with a spanning basis set of images. Construction of the basis set is typically performed using training imagery of the target region. The basis images are obtained using eigenvalue decomposition whereby the first basis image captures the most variance within the training data. Progressively, the basis images capture less and less of the critical information regarding the target. Thus, Eigen representations reduce the di-

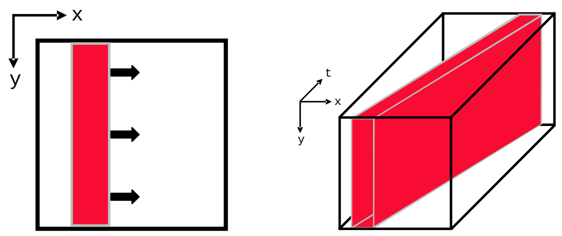


Figure 2.3: Left: A single frame of a video sequence where a vertical red bar is moving to the right. Right: The corresponding spatiotemporal volume representation.

mensionality of the target by including only a subset of basis images that account for the majority of the information content within the training data. When working with testing data, image regions can be compared against weighted combinations of the basis vectors to determine if there is a good match. There are a few problems with this regional feature representation. To begin, typically the Eigen basis images are specific to one category of target and must be computed from training imagery. An additional challenge is that variations of the true target that are not captured in the training data can not be properly modeled by the Eigen basis images. It should be noted that recent attempts have been made to resolve these problems by constructing the basis set in an online fashion (e.g., (Ross, Lim & Lin, 2008)).

2.2.4 Spatiotemporal Oriented Energies

Recall that in the previous section, we introduced the HOG representation of a target. HOGs can capture the spatial frequency properties of a target very compactly and effectively. Although HOGs can provide quite a rich description of a region's spatial structure, they are still somewhat lacking. In particular, for visual tracking, we are inherently interested in motion. Thus, it would seem logical if the feature representation of a target somehow captured the motion properties of a target as well as its spatial properties. Spatiotemporal oriented energy features (Adelson & Bergen, 1985) provide a mechanism for doing just that. Indeed, such an approach is an example of considering filtered pixel values as features, alluded to earlier.

Perhaps the most intuitive method of visualizing spatiotemporal oriented energy features is to consider the spatiotemporal volume representation of a video sequence. Consider Fig. 2.3, which shows a pictorial representation of the spatiotemporal domain for a simple video sequence where a vertical bar is moving to the right. To create this spatiotemporal volume representation, the video frames are stacked one on top of another. In the particular example shown in Figure 2.3, a diagonal structure is generated that extends from the front left to the back right of the volume. If we were to filter any individual image frame (i.e. slice) within a spatiotemporal volume at a number of orientations within the XY plane (with a orientation selective filter) we could capture the spatial properties of the image. Furthermore, if the filter's dominant orientation extended into the temporal dimension it would capture dynamic properties of the target such as velocity and flicker. This characterizing of the spatial and temporal properties of an image region is one of the fundamental ideas of oriented energies.

In addition to the filtering of the spatiotemporal domain, to compute so-called spatiotemporal oriented energies, a few addition steps are required (Adelson & Bergen, 1985). Filtering is typically performed in a quadrature fashion and the filter responses are point-wise rectified. Making use of a quadrature pair of filters is useful for eliminating unwanted phase information. The second derivative of a Gaussian, G_2 and its corresponding Hilbert transform, H_2, are commonly selected for filtering (Adelson & Bergen, 1985; Enzweiler, Wildes & Herpers, 2005; Derpanis & Gryn, 2005; Cannons & Wildes, 2007) since they are steerable, separable, and broadly tuned. Thus, oriented energies can be computed according to

$$e\left(\mathbf{x};\theta,\sigma\right)=\left[G_2\left(\theta,\sigma\right)*I\left(\mathbf{x}\right)\right]^2+\left[H_2\left(\theta,\sigma\right)*I\left(\mathbf{x}\right)\right]^2 \quad , \tag{2.3}$$

where $\mathbf{x}=(x,y,t)$ corresponds to spatiotemporal image coordinates, I is the image sequence, $*$ denotes convolution, and θ and σ are the orientation and scale at which filtering is performed, respectively. To attain a purer measure of energy that is less dependent on image contrast, Eq. 2.3 is normalized as

$$\hat{e}\left(\mathbf{x};\theta,\sigma\right)=\frac{e\left(\mathbf{x};\theta,\sigma\right)}{\sum_{\tilde{\sigma}}\sum_{\tilde{\theta}}e\left(\mathbf{x};\tilde{\theta},\tilde{\sigma}\right)+\epsilon} \quad , \tag{2.4}$$

where ϵ is a bias term to avoid instabilities when the energy content is small and the summations in the denominator cover all scale and orientation combinations.

For illustrative purposes, Fig. 2.4 displays a subset of the energies that can be computed for a single frame of a MERL traffic sequence (Brand & Kettnaker, 2000). In this particular video frame, a white car moving to the left near the center of an intersection. Notice how the energy channel tuned for leftward motion is very effective at distinguishing the car from the static background. Consideration of the channel tuned for horizontal structure shows how it captures the overall orientation structure of the white car. In contrast, while the channel tuned for vertical textures captures the outline of the crosswalks, it shows little response to the car, as it is largely devoid of vertical structure at the scales considered. Finally, note how the energies become more diffuse and capture more gross structure at the coarser scale.

Oriented energy features are quite flexible and can be utilized in a variety of ways within tracking or other sub-areas of computer vision. As described, they can provide a rich and compact feature set that captures both spatial and dynamic properties of a target and are robust to significant illumination changes.

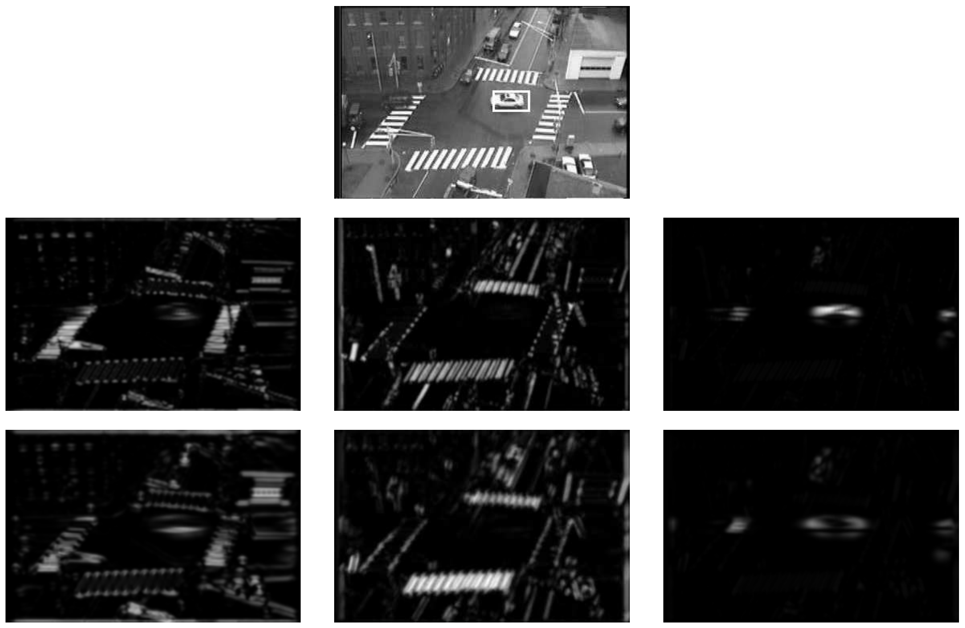


Figure 2.4: Frame from a MERL traffic video sequence with select corresponding energy channels. Finer and coarser scales are shown in rows two and three, resp. From left to right, the energy channels roughly correspond to horizontal structure, vertical structure, and leftward motion. Adapted from (Cannons & Wildes, 2007).

2.3 Prediction Versus No Prediction

One dividing line between the vast array of trackers in the literature is whether or not they incorporate a prediction mechanism. As mentioned previously, prediction is not absolutely necessary. Trackers can be constructed that perform target detection on each and every frame. Once targets have been detected, data association can be performed to link the object tracks to the currently detected targets. With these approaches, detection must be efficient if real-time performance is to be attained.

The approach of continuous detection is, however, somewhat wasteful. Significant information can be used from previous frames when tracking in the current frame. At the very least, the estimated target positions from the previous frame are known. Thus, many trackers will use this positional information as a starting point in the current frame. Assuming a high frame rate, this single piece of information can drastically reduce the search space and increase efficiency. The standard mean shift (Comaniciu, Ramesh & Meer, 2000) tracker employs this type of approach. Essentially, the previous position of the target is used as a starting point for a gradient descent-style optimization process in the current frame. Other trackers use even more than just the previous position information. Specifically, systems will often use information from previous frames (potentially over more than just a single frame) to estimate the target's velocity and predict its new position in the current frame.

There are two mechanisms that have become standard for performing prediction in the tracking community — Kalman filters and particle filters.

2.3.1 Kalman Filters

The Kalman filter's roots are in the optimal estimation theory literature where it was formally first presented by R. Kalman in 1960 (Kalman, 1960). The filter provides a means of optimally estimating the hidden state of a system by analyzing observable measurements. The fact that Kalman filters are "optimal" in some sense, makes it sound as if they are the best possible estimator available (i.e. that this problem of estimating a hidden state from a number of measurements is completely solved). What must be realized is that the Kalman filter is only optimal when a certain set of assumptions hold true. Once these assumptions are violated, the estimates provided by the Kalman filter may no longer be optimal. In practice, the Kalman filter still provides excellent performance in many tracking situations. As a result, the Kalman filter has become a favorite prediction tool among tracking researchers because of its performance in practical situations coupled with its theoretically optimal results.

Naturally, the next question one might have is: "what assumptions does the Kalman filter make?" The first assumption is with regards to the equations (or models) that capture the relationships between variables used by the filter. Specifically, these models must be linear. The second assumption is that all the uncertainty or "noise" that is included throughout the filter must be white (i.e. equal power within a fixed bandwidth), Gaussian noise. For many practical applications, these assumptions are sufficiently close to the truth that the algorithm performs well.

For this section of the paper, let $\mathbf{x}_t$ be the state of the system at time t and $\mathbf{z}_t$ be the measurements (to stay consistent with the Kalman filtering literature). Given the assumptions listed above, all distributions (or variables) are Gaussians. Hence, the distributions can all be characterized by a mean and covariance. Figure 2.5 illustrates the conceptual steps that one might to describe the Kalman filter: deterministic prediction (typically using a motion model when tracking); stochastic diffusion; and estimate refinement via a new measurement. Notice what happens to the state distribution during the filtering steps. During the deterministic prediction phase, the mean value of the state distribution changes, but the covariance remains fixed. (Note that to be mathematically correct, the covariance is actually updated simultaneously, alongside the mean of the distribution. However, for conceptual purposes, one could imagine the two steps occurring separately and sequentially. The illustrative series of steps in Fig. 2.5 connects quite nicely with particle filtering, which will be explained in Section 2.3.2). This result occurs because, during the prediction, the predictive model is assumed to be perfect. During the stochastic diffusion stage, the mean state remains fixed but the covariance is increased. Diffusion is incorporated to capture the inaccuracies of the deterministic model and other unavoidable noise that is in-

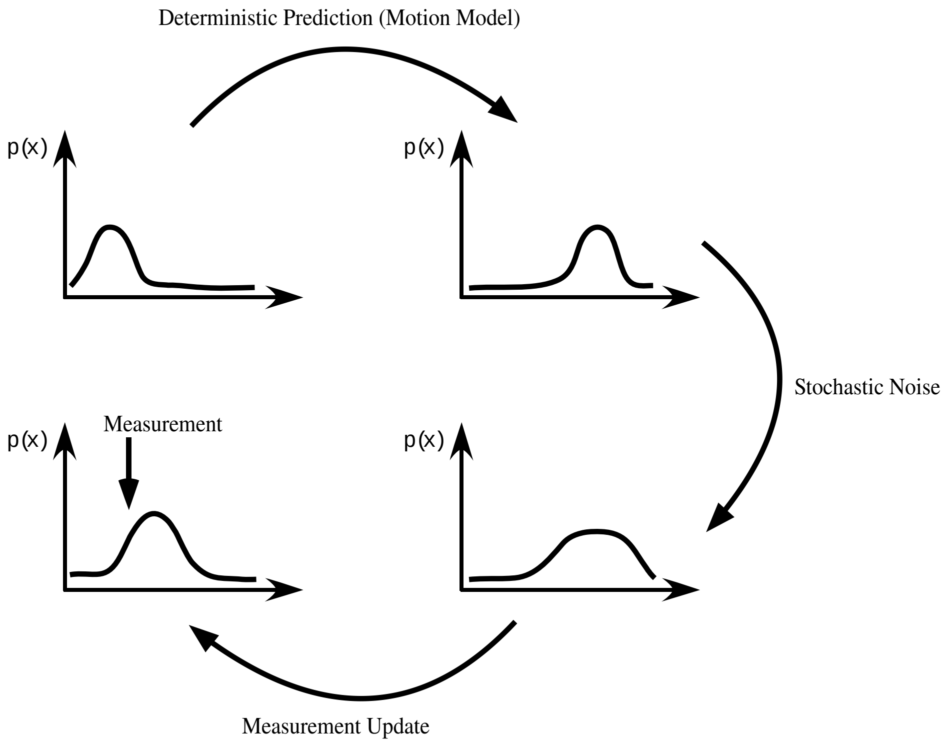


Figure 2.5: The main steps performed when Kalman filtering.

troduced during state prediction. The final step of the Kalman filter is to improve upon its predicted state estimation by using the measurements that are collected at the current time instant. Specifically, the predictive information is combined with the measurement data in an optimal manner to obtain a final estimate of the system state. This final step changes both the mean and covariance of the state. Another way to think about the process of Kalman filtering is as a prediction and correction scheme — using the deterministic model, a predicted state is computed which is subsequently corrected by the new incoming measurement. It is in this prediction-correction mindset that the Kalman filter algorithm will be described in more detail.

Mathematically, the predictive component of the Kalman filter can be described using an equation for the *a priori* state estimate and its covariance, respectively. By *a priori* state estimate, we mean the predicted state, prior to the arrival of the new measurements at time t. The state estimate can be written as follows

$$\hat{\mathbf{x}}_t^- = \mathbf{A}\hat{\mathbf{x}}_{t-1} + \mathbf{B}\hat{\mathbf{u}}_{t-1}, \tag{2.5}$$

where $\mathbf{A}$ is the system of equations (deterministic model) that relates the system state at time $t-1$ to the state at time t in the absence of control inputs, $\mathbf{u}$. The control inputs, and the system of equations, $\mathbf{B}$, that relate the control inputs to the system state are optional. Finally, the superscript "-" indicate that these are *a priori* estimates.

The system of equations, $\mathbf{A}$, need not be complex. For example, if the Kalman filter is being applied to tracking, the state of the system, $\mathbf{x}_t$ will include the position of the target and hence, $\mathbf{A}$ will include a model of the target's inter-frame motion. This motion model could be a very simple model, such as constant velocity. The main restriction is that the model must be linear (due to the Kalman filter assumptions). As an example, assume that the Kalman filter state is defined as the two-dimensional position of a target and its corresponding velocities: $\mathbf{x}_t = \{p_x, p_y, v_x, v_y\}^T$. Under a constant velocity assumption, the deterministic motion model, $\mathbf{A}$, would be:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.6}$$

The control inputs, $\mathbf{u}$, are usually omitted in visual tracking applications. However, in general the control inputs provide the Kalman predictive equation with some input from the outside world. Oftentimes this input can be from a human operator. For example, imagine that the state of a Kalman filter describes the position of a car. The deterministic update equations, $\mathbf{A}$, might represent a constant velocity or constant acceleration model while $\mathbf{u}$ might represent the human input provided via the steering wheel, break pedal, and gas pedal. The control input model, $\mathbf{B}$, uses physical rules to relate the control inputs to the system state.

In addition to the *a priori* estimation of the mean state, the covariance estimate is also required and can be computed according to

$$\hat{\mathbf{P}}_t^- = \mathbf{A}\mathbf{P}_{t-1}\mathbf{A}^T + \mathbf{Q}, \tag{2.7}$$

where $\mathbf{Q}$ is the covariance of the noise associated with this state prediction process. Note that Equations 2.5 and 2.7 define the steps performed at each frame to evaluate the Kalman filter state transition model

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_{t-1} + \mathbf{q}_{t-1}, \tag{2.8}$$

where $\mathbf{q}_{t-1}$ is the noise associated with the state transition process.

The second half of the Kalman filter is the correction stage, where the state estimate is improved by optimally combining the predicted value with the measurement information. An *a posterior* estimate of the system state can be obtained using

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- + \mathbf{K}_t \left(\mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}_t^-\right), \tag{2.9}$$

where

$$\mathbf{K}_t = \mathbf{P}_t{}^{-}\mathbf{H}^T\big(\mathbf{H}\mathbf{P}_t{}^{-}\mathbf{H}^T + \mathbf{R}\big)^{-1}, \tag{2.10}$$

and $\mathbf{R}$ is the noise associated with measurement process. Furthermore, $\mathbf{H}$ is a system of equations that maps a state vector into its equivalent measurement vector. In other words, $\mathbf{H}$ converts between state and measurements so that we can properly compare these two quantities. In the simplest case, $\mathbf{H}$ will be the identity matrix, indicating that the state and measurements are the same types of variables. However, in general, the state and measurements will not have the same form. The term $\left(\mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}_t^-\right)$ is often called the "innovation" as it is the error between the target's expected position and the position of a measurement. In a similar manner to the predictive component, the *a posterior* state covariance must be computed,

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\,\mathbf{P}_t{}^{-}. \tag{2.11}$$

Note that Equations 2.9, 2.10, and 2.11 are implicitly making use of the Kalman filter measurement model that defines the relationship between the measurements and the state. Explicitly, this measurement model can be written as

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{r}_t, \tag{2.12}$$

where $\mathbf{r}_t$ is the noise associated with the measurement process.

A more intuitive grasp of Kalman filtering can be gained by analyzing the above-mentioned equations in more detail. In Equation 2.10, $\mathbf{K}_t$ is commonly referred to as the Kalman gain. The Kalman gain determines how much the predicted state is trusted versus the measured values. The Kalman gain combines these two disparate sources of information based on their covariances. Let us consider a few asymptomatic examples. More specifically, we will assume a simple situation where H, the matrix used to relate measurements to states, is invertible. Given this assumption, first consider the situation where the measurement covariance, $\mathbf{R}$, tends to 0. In this case, $\mathbf{K}_t = \mathbf{H}^{-1}$. Hence, when we substitute the Kalman gain back into Eq. 2.9, we get $\hat{\mathbf{x}}_t = \mathbf{H}^{-1}\mathbf{z}_t$. The equations work as one would expect — when the measurements are extremely accurate, the *a posterior* state estimation relies more (or in this case, solely) on the measurements. The opposite situation occurs when the measurement covariance tends to infinity. In this case, the asymptotic Kalman gain goes to zero. Hence, Eq. 2.9 becomes $\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^-$. Again, as intuition guides us, when the measurements are inaccurate, the Kalman filter relies much more on the predicted state. Similar results are obtained when observing the asymptotic behavior of the prediction process covariance.

It should be clear that the Kalman filter can be an extremely powerful estimation tool. One advantage of this filter is that it combines multiple sources of information in a principled and optimal manner. Specifically, the Kalman filter can incorporate measurements from numerous sources to improve upon the state estimation, regardless of their accuracy or format. Another advantage of the Kalman filter is its recursiveness. The recursive nature of this filter just means that at each time step, all prior measurements and states do not have to be stored and reprocessed, like they would be with, for example, a Weiner filter. Instead, the Kalman filter implicitly incorporates all previous information and only explicitly requires information from the immediately previous time step.

Despite these advantages, the Kalman filter does have its limitations. To begin with, not all systems can be properly modeled using the Kalman filter assumptions. Specifically, some systems are not well-described by linear equations. Solutions have been proposed which help to alleviate this problem. In particular, the extended Kalman filter (EKF) allows the prediction and correction models to be non-linear (Bar-Shalom & Fortmann, 1988). In the extended Kalman filter framework, linear approximations of these non-linear equations are obtained, essentially using a Taylor series expansion. Subsequently, these linear approximations of the non-linear models can be incorporated into a slightly modified version of the standard Kalman filter framework. Another extension to the basic Kalman filter is the iterative extended Kalman filter (IEKF). In this case, Eq. 2.5 and Eq. 2.9 are essentially iteratively linearized and recomputed until convergence is reached (Bar-Shalom & Fortmann, 1988). A final popular variation is the unscented Kalman filter (UKF) (Julier, Uhlmann & Durrant-Whyte, 1995). With the UKF, rather than attempting to linearize the Kalman filter equations via a Taylor series expansion, deterministic sampling methods are used to represent the measurement and state variables.

Another limitation of the Kalman filter arises because of the Gaussian models that are employed. There are many instances where this simplified model is not appropriate. For example, if a target is being tracked throughout a cluttered environment, the measurement distribution might not be a unimodal Gaussian. Instead, it is quite feasible that the tracker might essentially have multiple hypotheses or multiple choices that are almost equally probable. A concrete example of this type of situation would be tracking a person's face when there are several other persons nearby. In this case, the target's face would be one hypothesis, but the faces of the other individuals would also be feasible alternatives for the tracker. In this scenario, the measurement distribution is far from a unimodal Gaussian! Instead, the distribution is multimodal, where there would roughly be a one-to-one correspondence between the peaks in the distribution and the human faces. Fortunately, there is another state prediction tool that overcomes the limitations of the Kalman filter — the particle filter.

2.3.2 Particle Filters

The particle filter that is known and used in computer vision has very historic roots in the fields of mathematics and physics. Specifically, the study of so-called Monte Carlo methods entered the mainstream in chemical physics in the 1950s with works such as (Metropolis, Rosenbluth, Rosenbluth & Teller, 1953; Rosenbluth & Rosenbluth, 1955). The general idea of Monte Carlo methods is to use random sampling to obtain an approximate solution when closed-form, exact solutions are infeasible. The particle filters commonly used in computer vision are sometimes referred to as sequential Monte Carlo methods, an extension to the original Monte Carlo technique. Sequential Monte Carlo methods are used for approximating a sequence of probability distributions using random sampling. A complete analysis of the Monte Carlo and sequential Monte Carlo literature is outside the scope of this review. Instead, we will now turn to a more specific discussion of the particle filtering framework in the manner in which it was introduced to the field of computer vision.

Particle filters (also commonly referred to as the CONDENSATION algorithm (Isard & Blake, 1996)) remove the major restrictions that are present when using Kalman filtering for visual tracking. Notably, with particle filters the prediction and update equations need not be linear. Furthermore, the distributions can be multimodal. As first demonstrated by Isard and Blake (Isard & Blake, 1996), the removal of these limitations makes the particle filter well-suited for tracking in cluttered environments. To make these gains, however, the particle filter foregos a closed-form solution and instead resorts to sampling methods. Previously we elected to describe the Kalman filter from more of an algorithmic perspective, with a set of recursive equations. The Kalman filter can be alternatively derived under a Bayesian probabilistic framework. To provide a more complete picture of these prediction mechanisms, we will now present the particle filter from a probabilistic point of view; however, the linkages between these two perspectives will still be made, when appropriate.

As Fig. 2.6 shows, the general steps of the particle filter are very similar to the Kalman filter. Specifically, particle filters still perform deterministic prediction, stochastic diffusion, and correction via measurement updates. The primarily and significant difference, however, is that instead of propagating the mean and variance of a Gaussian distribution, samples from distributions of unspecified shape are propagated.

When probabilistically deriving the particle filter, the final goal is to compute $p\left(\mathbf{x}_t|\mathbf{z}_t\right)$. In words, we want to find the probability distribution that describes the system state, given all the measurements that have been observed. This distribution is analogous to what we previously termed the *a posterior* estimate of the system state when describing the Kalman filter. Using Bayes' rule, we know that

$$p\left(\mathbf{x}_t|\mathbf{z}_t\right) = kp\left(\mathbf{z}_t|\mathbf{x}_t\right)p\left(\mathbf{x}_t\right), \tag{2.13}$$

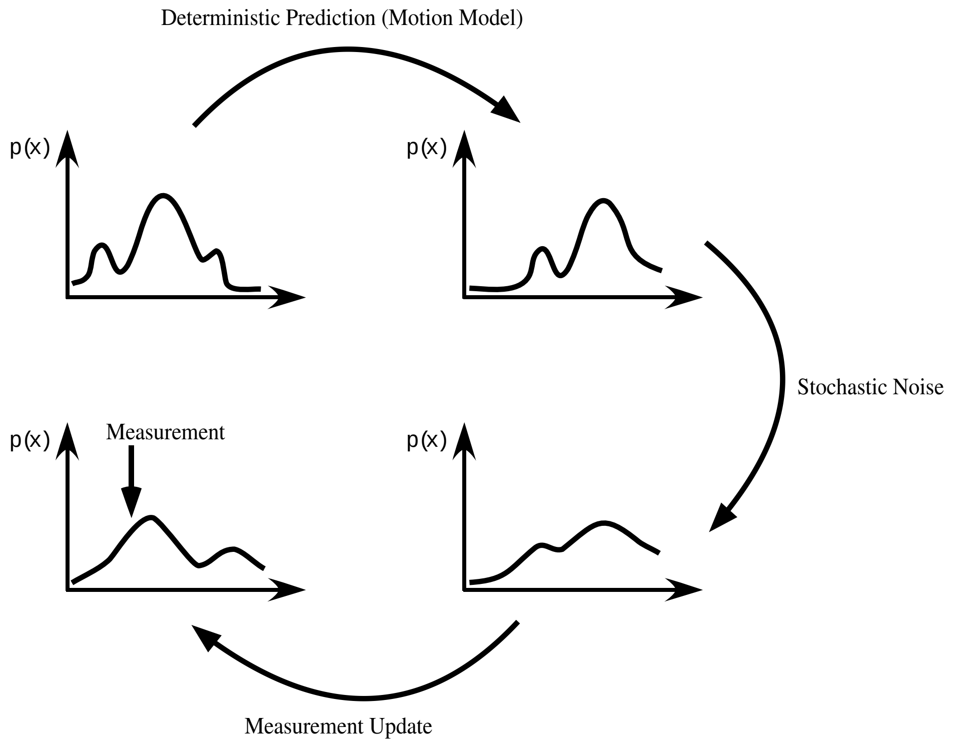


Figure 2.6: The main steps performed when particle filtering.

where k is a constant that does not depend on $\mathbf{x}_t$. Unlike Kalman filtering where all of the probability distributions are Gaussian, with particle filtering the distributions are of any arbitrary type. This added freedom leads directly to the need for a sampling-based approach. With particle filters, a factored sampling approach is used to solve Eq. 2.13. With factored sampling, a set of samples, $\left\{\mathbf{s}_t^{(1)} \ldots, \mathbf{s}_t^{(N)}\right\}$ is first drawn from the prior distribution, $p\left(\mathbf{x}_t\right)$. These samples are then each weighted according to:

$$\pi_t^n = \frac{p\left(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(n)}\right)}{\sum_{j=1}^N p\left(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(j)}\right)}, \tag{2.14}$$

where $p\left(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(n)}\right)$ is the probability of observing a measurement, given that the system is in state $\mathbf{x}_t = \mathbf{s}_t^{(n)}$. The samples, $\left\{\mathbf{s}_t^{(1)} \ldots, \mathbf{s}_t^{(N)}\right\}$, can then be randomly selected according to their probabilities, π_t^n. These final samples are selected in a manner that approximates drawing from the distribution $p\left(\mathbf{x}_t|\mathbf{z}_t\right)$. This approximation improves as the number of samples, N, increases.

When particle filtering is applied to tracking, these factored sampling steps are essentially used on a frame-by-frame basis. Specifically, when computations on the previous frame, $t-1$, are completed, an estimate of $p\left(\mathbf{x}_{t-1}|\mathbf{z}_{t-1}\right)$ is obtained. This probability is the *a posterior* state estimate at time $t-1$. Since particle filtering deals with samples, the estimate of $p\left(\mathbf{x}_{t-1}|\mathbf{z}_{t-1}\right)$ is in the form of a number of samples and their corresponding weights $\left\{\left(\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}\right), n=1,\ldots,N\right\}$. Thus when tracking with a particle filter, the first step is to sample from this approximated posterior distribution. These samples are randomly selected, with replacement, according to their weights. Once the new samples have been selected, state prediction is used to "guess" the current state of the system. As was seen with the Kalman filter, this prediction is comprised of two stages. First, the deterministic drift (in tracking, typically a motion model) is performed. In the particle filter framework, this step entails the application of the deterministic model to the state samples. Following the deterministic "drift", the second stage of prediction is performed, whereby noise is superimposed upon the modified samples. This noise accounts for inaccuracies in the predictive model. Once the predictive component is complete, the resulting sample set, in affect, is an estimate of $p\left(\mathbf{x}_t|\mathbf{z}_{t-1}\right)$. With reference to our Kalman filter description, this distribution is analogous to the *a priori* state estimate.

As noted by Fig. 2.6, the particle filter still needs to improve the predicted state estimate by incorporating measurement information from the current frame. The manner in which measurement information is used is dependent on the particular application. For instance, in (Isard & Blake, 1996), contours were used for tracking, hence $p\left(\mathbf{z}_t|\mathbf{x}_t\right)$ is defined using features along lines that are perpendicular to the contour. In contrast, particle filters have also been used in conjunction with color histograms (Nummiaro, Koller-Meier & Gool, 2002). In the latter case, the authors defined the measurement distribution using the Bhattacharrya match score between the template and candidate target locations. No matter how the observation distribution is determined, it has the affect of defining the the weights, π, for each of the current samples. Once the sample weights are defined, the particle filter is complete! According to factored sampling theory, the new set of samples along with their respective weights, provide an approximation of $p\left(\mathbf{x}_t|\mathbf{z}_t\right)$. The samples and weights will then be propagated to the next frame and the whole process begins again.

As alluded to previously, perhaps the most important advantage of the particle filter over the Kalman filter is its ability to track through clutter. However, there are additional advantages. In some ways, the particle filter can actually be argued to be simpler than the Kalman filter. The rationale is that particle filters do not require the computation of cumbersome covariance estimates or Kalman gain terms. As a consequence of using sampling methods, many researchers feared that particle filters would be computationally intense. However, it appears that the algorithm can run in near real-time (Isard & Blake, 1996) (1996-based hardware) or real-time (Nummiaro, Koller-Meier & Gool, 2002). One method to ensure that this sampling-based approach remains computationally feasible is to use a fixed number of samples, N, for the distributions.

2.4 Data Association Techniques

Target tracking within the field of computer vision is a new research area, relative to other fields of study. Specifically, in the field of military surveillance, the problem of tracking multiple targets saw its pioneering work at least as early as 1964 (Sittler, 1964; Bar-Shalom & Fortmann, 1988). Typically, in the case of target tracking in a RADAR environment, the amount of information available for processing is much more limited than that of a movie video sequence. Computer vision trackers have color, texture, and gradient information at their disposal. In RADAR-based systems, the captured measurements at time t are usually just simple, discrete points. Thus, RADAR tracking systems must rely purely on position and motion information, rather than appearance.

The RADAR target tracking problem can be summarized as follows. In the environment, there exist a number of targets to be tracked and the tracking system uses a sensor to capture information about these targets. These sensor measurements, $\mathbf{z}(t)$, are the inputs to the tracking algorithms. Ideally, there would be a one-to-one correspondence between the measurements and the targets in the environment. However, the sensors are not perfect and consequently, noise is introduced during the measurement process. The goal of the tracking system is thus, to use the noisy measurements supplied by the sensors to assign positional "tracks" to the targets of interest in the environment. More specifically, at every time step, the system must assign the measurements to the appropriate target tracks, which is commonly referred to as the data association problem. Furthermore, this assignment should be made subject to the constraint that the relationship between features and measurements is one-to-one.

Data association is a straightforward problem if there is just one target in the environment. Assuming no measurement noise and just a single target, there will be just one measurement at every time instant and hence data association is trivial. Furthermore, if there are multiple targets in the scene (creating multiple measurements), but the distance between these targets is large relative to the targets' velocities, the problem is still trivial. In particular, the targets are so far apart that there is negligible ambiguity as to which measurement belongs to which track. Realistically, the data association problem is not this simple. Invariably, targets of interest move close to one another, occlude one another, and noise can be significant. Additionally, targets will often move out of the field of view or new targets may enter the scene. Here, we will briefly describe several of the most common techniques used to address the data association problem. Specifically, we will describe: the nearest

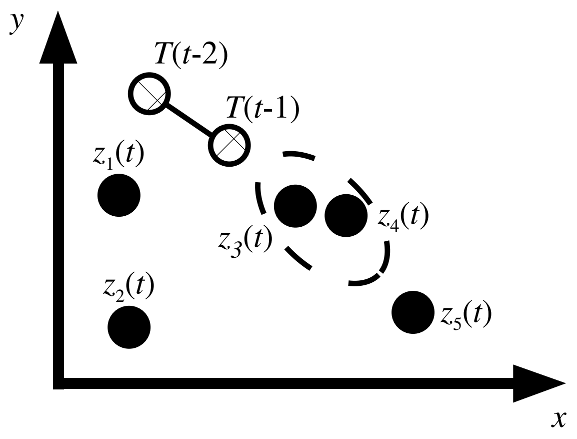


Figure 2.7: Data association gating technique using Mahalanobis distance.

neighbor algorithm (Crowley, Stelmaszyk & Discours, 1988); the track-splitting filter (TSF) (Smith & Buechler, 1975); the joint-likelihood (JL) method (Morefield, 1977); the multiple-hypothesis algorithm (MHA) (Reid, 1979); and the joint-probabilistic data-association filter (JPDAF) (Fortmann, Bar-Shalom & Scheffe, 1983).

In their standard formulations (Cox, 1993), the above algorithms make use of the Mahalanobis distance. The Mahalanobis is defined as

$$\gamma = \mathbf{v}_i^T(t)\,\mathbf{S}(t)^{-1}\,\mathbf{v}_i(t) \tag{2.15}$$

where $\mathbf{v}_i^T(t)$ is the innovation (or error) and is defined as the distance between a measurement $\mathbf{z}_i(t)$ and the predicted position of a measurement $\hat{\mathbf{z}}(t|t-1)$. Additionally, $\mathbf{S}(t)$ is the covariance of the innovation estimate. The estimated measurement position is often obtained using a Kalman filter. The Mahalanobis distance has some advantages over the Euclidean distance metric because it incorporates the uncertainties in the various dimensions of the data via the covariance matrix.

Data association algorithms will almost always employ a "gating" procedure whereby the Mahalanobis distance is computed between the predicted position of a target and all measurements. If the Mahalanobis distance for a particular measurement is greater than a threshold, the measurement is no longer considered as a potential match for the target. This region around a target's predicted position, whereby the Mahalanobis meets the threshold condition is often termed the "validation volume". When using the Mahalanobis distance, this gating procedure is theoretically well-founded as the Mahalanobis distance is analogous to the χ^2 distribution. Thus, a threshold can be selected such that, for example, all points within the validation volume have at least a 95% chance of corresponding to a feature. Figure 2.7 illustrates a tracking example when data association is required. The circles with cross hatching, $\mathbf{T}(t)$, indicate measurements that have already been assigned to the target track at previous time instants. The black circles, $\mathbf{z}(t)$, indicate measurements provided by the sensors at the current time. The dotted ellipse indicates the validation volume around the target's predicted position at time t. Notice how only two measurements, $\mathbf{z}_3$ and $\mathbf{z}_4$, lie within the validation volume. In this situation, all other measurements will be immediately discarded and the target track will be associated with either $\mathbf{z}_3$ or $\mathbf{z}_4$.

2.4.1 Nearest Neighbor Data Association

Having defined a distance metric, an obvious approach to the data association problem would be to assign the measurement to the track that is closest to the predicted position. This, suboptimal, approach is known as nearest neighbor data association (Crowley, Stelmaszyk & Discours, 1988). Clearly, there is a chance of establishing incorrect correspondences, especially when the targets are moving in close proximity. The biggest limitation of this approach, is that it makes irreversible assignment decisions immediately. Information obtained in the future can be invaluable at resolving ambiguities at the current time step. However, the nearest neighbor algorithm does not make use of this information. Nonetheless, the nearest neighbor data association technique is still popular because of its simplicity, efficiency, and its ability to provide acceptable results in certain situations.

2.4.2 Track-Splitting Filter

An alternative to the the basic nearest neighbor algorithm is the track-splitting filter (Smith & Buechler, 1975). Rather than making an immediate decision regarding ambiguous correspondences, the track-splitting filter keeps its options open until a more informed decision can be made. In particular, whenever multiple measurements are observed within a predicted feature's validation volume, all measurements are matched to the track. This algorithm is well represented as a tree structure where a path in the tree from root to leaf represents a sequence of measurements that are assumed to come from the same environmental feature. Figure 2.8 shows the track-splitting tree that corresponds to the data association scenario displayed in Fig. 2.7. In this example, for the previous two time steps, only one measurement was observed in each of the respective validation volumes, thus no branching occurs. However, at the current time step, two measurements were observed within the feature's validation volume. Accordingly, two child nodes (corresponding to the two measurements) are spawned.

Using the track-splitting filter, the two branches will continue to grow until a latter point, when a more informed decision can be made. Naturally, the size of the tree can explode if no path pruning is employed. It can be shown (Cox, 1993) that when the Mahalanobis distance is used, the modified log-likelihood of a "potential target track" (i.e. a path from root to leaf node) can be computed in a recursive

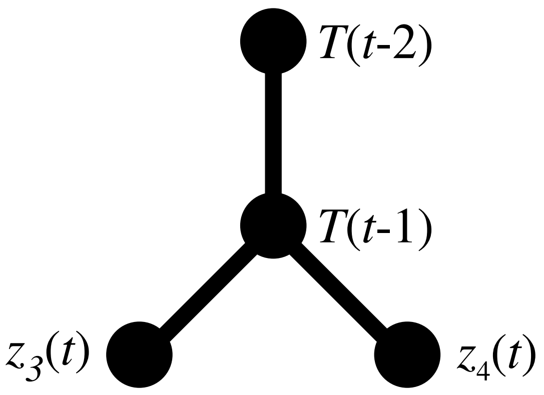


Figure 2.8: Example of a track-splitting tree.

manner. Specifically, this modified negative log-likelihood is approximated by the sum of the Mahalanobis distances between each measurement along the track and the corresponding predicted target position. This log-likelihood can be used to prune paths that have a likelihood that falls below a threshold level. Typically, this pruning strategy will be employed in a sliding window manner so that the old measurements do not dominate the likelihood computation. In addition to deleting potential target tracks, merging can be performed if multiple potential tracks are almost identical. Another pruning strategy is that of maintaining only “M” tracks that have the largest log-likelihoods after deleting and merging has been completed.

2.4.3 Joint-Likelihood Data Association

A significant limitation of the track-splitter filter is that it does not enforce the constraint that features and measurements must have a one-to-one relationship. In other words, for a set of tracks to be physically plausible, they should be disjoint. The joint-likelihood algorithm (Morefield, 1977) is an alternative that adheres to these constraints. The joint-likelihood algorithm operates by first identifying a set of possible tracks using, for example, the track-splitter filter or other comparable techniques. At this stage, it is assumed that these initial tracks are not disjoint. The joint-likelihood algorithm then enforces the one-to-one constraint by combinatorially evaluating each set of feasible, disjoint tracks. For each disjoint set of tracks (also known as a partition), a likelihood can be computed and the partition with the maximum likelihood provides the optimal solution. A significant disadvantage of this approach is that it is known to be NP-complete. Specifically, the problem is reducible to the set-packing problem. Accordingly, computer packages exist that can provide approximate solutions in a more efficient manner. These non-optimal algorithms often use techniques such as greedy algorithms or simulated annealing. Another disadvantage is that the algorithm is typically implemented in a batch manner and thus, cannot provide real-time feedback.

2.4.4 Multiple-Hypothesis Algorithm

As one might imagine, an algorithm's ability to handle situations where the number of targets changes is a significant advantage. One data association algorithm that can do just that is the multiple-hypothesis algorithm (Reid, 1979). The multiple-hypothesis algorithm can again be visualized as a tree structure. In this tree structure, each layer represents a specific time instant. For example, if data association is being performed only at time $t = 1, 2, 3$ then the tree structure will have three levels. However, unlike the track-splitting filter where each tree node corresponds to a measurement, each node within the tree corresponds to a set of disjoint object tracks. Each set of disjoint tracks is a single viable hypothesis regarding the assignment of measurements up to and including the current time instant. With these basics in mind, the main steps of the multiple hypothesis algorithm are as follows. The leaf nodes at the previous time instant become the parent nodes at the current time. Each new parent produces a set of children that represent all combinations between the parent's tracks and the new measurements. Thus, at each time instant, the multiple hypothesis algorithm exhaustively considers all possible associations between tracks and measurements. A probability is computed for each child, indicating the plausibility of the the child's track set.

As one might imagine, the multiple hypothesis algorithm has exponential complexity. As was seen with previous algorithms, pruning strategies must be employed to limit the computation time. One method to limit memory and computation time is to manage separate trees for spatially disparate targets. Of course, if the targets subsequently move closer to one another, merging of trees may be required. In terms of pruning a specific tree, the typical method used in the multiple hypothesis framework is the "N-scan back algorithm". Put simply, at time $t + N$, the tree node at time t must make a final decision and keep only a single child branch. All other child branches of the node at time t will be discarded. A pruning decision is made by summing probabilities of the leaf nodes associated with each child branch, in turn. The branch containing the leaf nodes with the maximum sum is retained, as it indicates the most probable child branch. Figure 2.9 shows an example of the N-scan back pruning method when $N = 2$.

The major limitation of the algorithms mentioned previously (excluding simple nearest neighbor) is that they are all of exponential complexity. To make the algorithms tractable, heuristic pruning strategies are employed. Noticing this limitation, researchers designed the joint probabilistic data association filter which provides a sub-optimal solution, but is much more memory and computationally efficient than the alternatives. Accordingly, the JPDAF is more amenable for real-time applications.

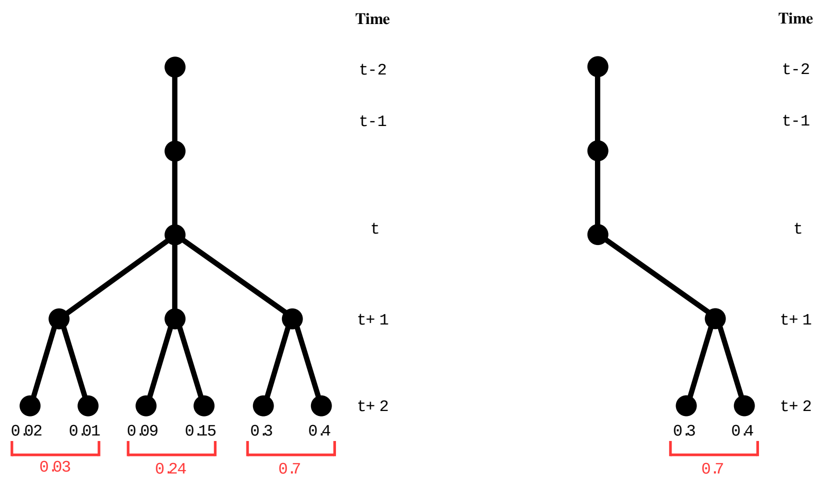


Figure 2.9: Example of the N-scan back algorithm. The left hand side shows a tree prior to pruning at time t. The right hand side shows the tree after pruning has been performed.

2.4.5 Joint Probabilistic Data Association Filter

The general idea behind the JPDAF is as follows (Fortmann, Bar-Shalom & Scheffe, 1983). Assuming that the number of targets is known and constant, the JPDAF essentially deploys a modified Kalman filter to track each target. The significant difference between situations where standard Kalman filters and JPDAFs are used, is the number of targets being tracked. Specifically, a Kalman filter is deployed to track a single target whereas with a JPDAF, multiple targets can be tracked. When using a Kalman filter, we have the luxury of knowing that all measurements arise from the target of interest. However, when multiple targets are present, these associations are not always obvious (i.e. the data association problem). Hence, the JPDAF requires a method of determining which measurements should be used to update each Kalman filter.

Instead of attempting to explicitly perform this correspondence, the JPDAF uses the novel idea of associating all measurements with every target to compute a combined innovation term. Mathematically, this combined innovation can be computed using

$$v^l = \sum_{i=1}^{Z} \beta_i^l v_{i,l}, \tag{2.16}$$

where Z is the number of measurements at time t, $v_{i,l}$ is the error when associating the predicted position for track l with the measurement $\mathbf{z}_i$, and β_i^l is the posterior probability that measurement $\mathbf{z}_i$ originated from target track l. Recall that in the Kalman filter framework, there is an innovation term that defines the difference between the expected position of the target and the position of a measurement (i.e. $\left(\mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}_t^-\right)$). When computing a target track with a JPDAF, the combined innovation, v^l, is used in place of the standard innovation. The biggest limitation of the JPDAF is that the number of targets being tracked must be known and remain constant. Nonetheless, the fact that the JPDAF is substantially less computationally and memory intensive as compared to many of its competing data association counterparts, it can be useful in certain applications.

It should be noted that recently, probabilistic data association techniques that are more specific to computer vision applications are beginning to emerge. In the RADAR tracking paradigm, one to one correspondences between measurements and targets are typically desired. However, in the case of computer vision-based visual tracking, this constraint may not be appropriate. For example, in (Khan, Balch & Dellaert, 2006) a probabilistic data association algorithm is proposed that eliminates these assumptions that a target can only generate one measurement and vice versa.

To summarize this section, Table 2.1 lists the properties of each of the data association techniques discussed. Unquestionably, there are various modifications that can be made to the abovementioned algorithms. For example, a nearest neighbor data association technique could conceivably be created that enforces one-to-one matches between measurements and objects. However, in Table 2.1, we list the properties of the most basic versions of the algorithms. Although the JPDAF appears to have similar characteristics to the nearest neighbor algorithm, in practice, it generally performs much better as it does not make absolute assignment decisions. If the number of targets is unknown or can change, the MHA is arguably the best approach, although pruning strategies will be required to make the algorithm computationally feasible.

Table 2.1: Summary of Data Association Techniques

Algorithm	Uses Future Information	Complexity	Track Termination and Initialization	Enforces One-to-One Matches
NN	No	Not Exponential	No	No
TSF	Yes	Exponential	No	No
JL	Yes	Exponential	No	Yes
MHA	Yes	Exponential	Yes	Yes
JPDAF	No	Not Exponential	No	No

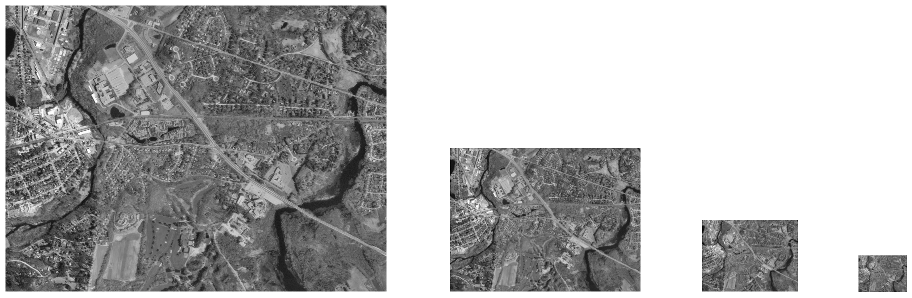


Figure 2.10: Example of a Gaussian pyramid. From left to right, levels zero to three are shown.

2.5 Coarse-to-Fine Processing

Coarse-to-fine processing is a topic that will occur throughout this review as it is employed in a number of different trackers. More generally, coarse-to-fine processing is a tool that is commonly used in several aspects of computer vision including stereo vision and motion estimation. The essential idea behind coarse-to-fine processing is to construct a pyramid representation of an image. This pyramid contains modified versions of the original image, each of which highlight a different frequency band and have a different pixel density. Two of the most widely used coarse-to-fine pyramids are Gaussian pyramids (Burt, 1981) and Laplacian pyramids (Burt & Adelson, 1983). A Gaussian pyramid is constructed by repetitively downsampling and low-pass filtering the original image. Thus, the bottom of the image pyramid is the original image containing all frequency content (low to high). As one traverses up the pyramid, the image dimensions are downsampled by a factor of two per level, and the higher frequency content is increasingly attenuated. A Laplacian pyramid is constructed in a similar manner; however, rather than using low-pass filtering, a band-pass filter is employed instead. Figure 2.10 shows a sample Gaussian pyramid.

Within the context of tracking, coarse-to-fine processing is typically used to obtain initial target motion estimates (at the coarsest scale). These initial estimates can then be refined to obtain a more precise estimate of the target motion, using the additional frequency bands that are available in the lower pyramid levels (e.g., (Anandan, 1989; Bergen, Anandan, Hanna & Hingorani, 1992; Burt, Yen & Xy, 1983)). Accordingly, trackers will often locate the location of the target in the highest level of the pyramid. Subsequently, this estimate is propagated to the next highest level (taking care to consider the differing pixel densities), where the target's position can then be refined. This process is repeated, typically until level zero or level one of the pyramid (level zero is often not used due to high frequency noise).

There are several reasons why a tracker designer may wish to make use of coarse-to-fine processing. To begin, coarse-to-fine processing can be used for computational efficiency purposes. Perhaps, even more importantly, coarse-to-fine processing allows for larger target motions to be estimated. On the other hand, errors made during the processing of the coarser scales cannot be recovered from when considering the lower levels of the pyramid. Despite this drawback, coarse-to-fine processing can provide significant advantages when deployed appropriately.

2.6 Template Updating Procedures

When tracking a target throughout a video, it is expected that the target's appearance will change, especially if the sequence is long in duration. Appearance changes can arise from changes in viewpoint, illumination changes, non-rigid target deformations, etc. Many tracking algorithms maintain a "template", representing the target that is being searched for in each image frame in terms of its aggregate features. If the target experiences large appearance changes (with whatever feature set is being used), techniques will be required to adapt the template in order to obtain reliable tracking.

Some trackers merely assume that the video sequences are sufficiently short or that the target's appearance is close enough to constant that no template updates are required. Other trackers have attempted to update the template every frame by using the optimal target representation found in the current frame. However, this second approach has the problem of being distracted by background clutter. In order to update a template, registration must be performed to bring the target, as seen in the current frame, into alignment with the template. In (Irani, Rousso & Peleg, 1994) a method is proposed for computing the motion parameters of a target, even in the presence of other motion. The approach first assumes a simple motion model (e.g., translational) and computes an approximation of the target motion under this simple model. Subsequently, the initial motion estimates provided by the low-level model are used to initialize parameter estimation for more complex motion models (e.g., affine, projective). With an estimate of the inter-frame motion, the target region in the current frame can be brought back into alignment with the template to perform updating procedures. In (Irani, Rousso & Peleg, 1994), updates are performed such that the new template is a weighted combination of the current image and a modified version of the current image that has been brought into alignment with the template. On the other hand, in (Matthews, Ishikawa & Baker, 2004), the new template is selected as a weighted combination of the original template (from the first frame) and the aligned target region in the current image.

Another approach to template updates that is that of updating the template based on a confidence match score (Comaniciu, Ramesh & Meer, 2000; Enzweiler, Wildes & Herpers, 2005). Specifically, if the target located in the current frame is well-matched with the template, only minor updates are performed on the template. On the other hand, if the optimal target location identified in the current frame is not well-matched to the template, more aggressive template updates are performed. In some situations, this heuristic approach appears to be acceptable. For example, if the target's appearance is suddenly altered via a change in illumination, the template will be quickly updated to capture that change. However, if the tracker begins to slip off the true target (resulting in low confidence matches), the template will be updated quickly, allowing for non-target regions to quickly enter into the template representation.

Perhaps the most principled approaches of dealing with target appearance changes was presented in (Jepson, Fleet & El-Maraghi, 2003). In this work, both slowly varying and quickly changing aspects of the target's appearance are modeled. The tracker is able to dynamically rely on the various aspects of the representation that are reliable within the current image frame. Results in this work show impressive adeptness at handling target appearance changes.

Template updating is an unsolved, but critical, problem that will be mentioned throughout this review. Although this issue can affect many categories of trackers, it will become most apparent when discussing region-based trackers in Chapter 5.

2.7 Active Camera Control

Active vision is another aspect of tracking, whereby the system has the ability to move the camera to a desired position and set an appropriate level of zoom. Typically, this specialized type of camera is called a pan-tilt-zoom (PTZ) camera. There are numerous reasons why a moving camera may prove to be beneficial. To begin, a target typically only remains within an immovable camera's field of view for a very short time. Additionally, when a target quickly passes through a single, stationary camera's field of view, motion blur often arises. The immovable camera is focused on both the target and the background and thus, as the target moves, blurring artifacts occur that degrade the image of the target. PTZ cameras can eliminate both of these disadvantages. In particular, a PTZ camera can be adjusted such that the target remains at the center of the field of view, causing less motion blur on the target (but more prominent artifacts on the background) and keeping it in view for longer (Coombs & Brown, 1993). To extend the amount of time that the target's image is captured by the camera even further, a PTZ camera can be mounted on a moving robot. In this case, the target can be kept in the center of the field of view indefinitely as the robot pursues the target. Furthermore, if the camera has zooming capabilities, the zoom level can be modified to obtain a certain number of pixels on the target.

Clearly, PTZ cameras can be useful in target tracking. This section will detail some of the works that develop the fundamentals of using PTZ cameras in conjunction with computer vision tracking systems. In later chapters, specific references will also be made to additional trackers that make use of active vision components (Marchand,

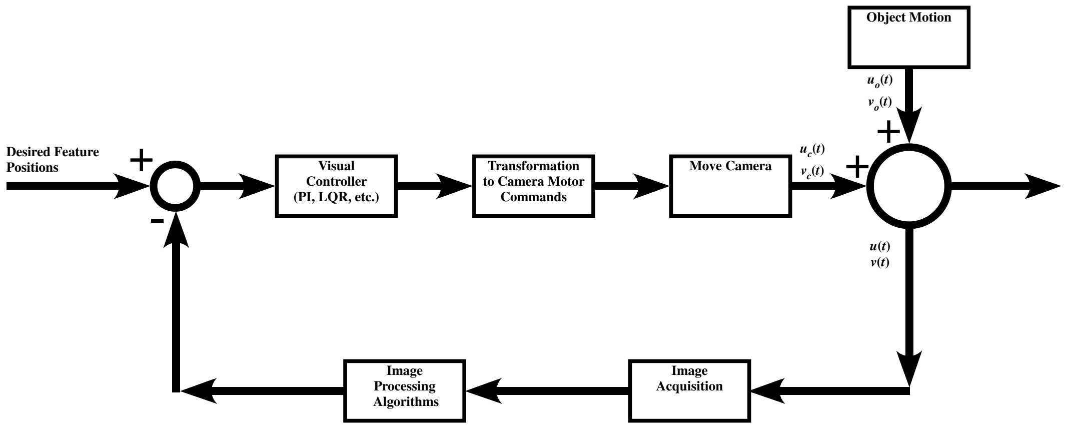

Object Motion
$u_o(t)$
$v_o(t)$
Desired Feature Positions
+
-
Visual Controller (PI, LQR, etc.)
Transformation to Camera Motor Commands
Move Camera
$u_c(t)$
$v_c(t)$
+
+
$u(t)$
$v(t)$
Image Processing Algorithms
Image Acquisition

Figure 2.11: Block diagram of the major tasks performed by most active vision tracking systems.

Bouthemy & Chaumette, 2001; Wixson, Eledath, Hansen, Mandelbaum & Mishra, 1998; Uhlin, Nordlund, Maki & Eklundh, 1995; Drummond & Cipolla, 2002).

Figure 2.11 shows a block diagram of a general control system that could be used for active tracking. The input to the control system consists of the desired feature positions. In active tracking, these features will be related to the position of the target within the field of view. In most situations it is desired to have the target remain at the center of the field of view. Thus, the input features will either be the actual center of the visual field, or some other variable that can be related to this form of positional information. To simplify matters, here it will be assumed that the input features are the center of the field of view in a particular reference frame.

The first task the control system performs is to compute an error signal. This error is obtained by subtracting the desired position of the target (i.e. center of the field of view) and the actual position of the target. In the control theory literature, the *controlled* variable is the quantity that is being measured and is typically the output of the system. In active tracking, the controlled variable is the true position of the target in the field of view. The job of the control system is to minimize the error between the desired position and the true position, ideally in a manner that provides smooth tracking of the target. To accomplish this task, standard visual controllers are used. Common controllers that can be employed include proportional-integral (PI) controllers and linear-quadratic regulators (LQR). Not unlike a Kalman filter, these controllers analyze the error as well as the system state to modify the *manipulated* variables to reduce the error. The manipulated variables are the system parameters to which the control system has access. When changed, manipulated variables have an effect on the controlled variables. In the case of active vision trackers, the manipulated variables are the PTZ camera controls.

The controller operates by taking the error signals as input and subsequently providing as output a set of signals for controlling the motors attached to the camera(s). This step is optional, but usually the output of the visual controller is not in the correct format for directly controlling the camera angles, making a transformation necessary. This transformation is quite system-specific as it depends on the camera system setup, the type of camera(s) deployed, etc. Once the signals have been modified into the correct format for controlling the camera motors, the PTZ camera is actually adjusted.

As Figure 2.11 illustrates, active vision tracking control systems are *closed-loop*. In other words, there is a feedback loop from the output of the system that eventually leads back to the input of the system. The feedback is necessary to compare the current system output with the desired input. An example of a closed-loop system that people use every day is a thermostat in one's car or home. The computer vision aspects of active vision systems primarily arises in the feedback loop, as Figure 2.11 indicates. The first step in the feedback loop is to acquire an image from the camera. The current frame will appear to be moving with respect to the previous image. This overall motion is the superposition of both the object motion and the camera motion. Image processing techniques are employed to extract information regarding the position of the target within the field of view. The techniques and features utilized are completely up to the designer; however, motion is a common feature found in the literature. The primary and significant limitation imposed upon the image processing algorithms is that they must be efficient — the control system needs to move to keep tracking the target, thus large delays due to overly complex image processing algorithms are not ideal. In fact, these delays are one of the largest problems that must be overcome by active vision trackers. Once the target's position has been extracted, it can be compared to the desired target position and the process repeats once again.

As mentioned, the task of controlling the camera motors is quite dependent on the researcher's camera setup and type. Figure 2.12 shows an overhead view of an example binocular camera rig. In general, most binocular camera systems have a setup that is quite similar to that of this figure. With this binocular setup, there are four angles of interest: the tilt angle, ϕ; the pan angle χ; and two vergence angles, θ_r, θ_l, for the right and left cameras, respectively. The tilt angle rotates the rig between the floor and ceiling, in a similar manner to a person nodding his or her head. The pan angle rotates the rig in a similar manner to a person rotating his or her head from shoulder to shoulder. Finally, the vergence angles, rotate the cameras' optical axes in a similar manner to a person moving his or her eyes to focus on a particular object in his or her field of view. By comparing the difference between the desired to current target position, the control system must determine how to change these angles to improve upon this error. The motor signals required to move the camera are typically directly related to the tilt, pan, and vergence angles.

When developing an active vision system, the two main components that require

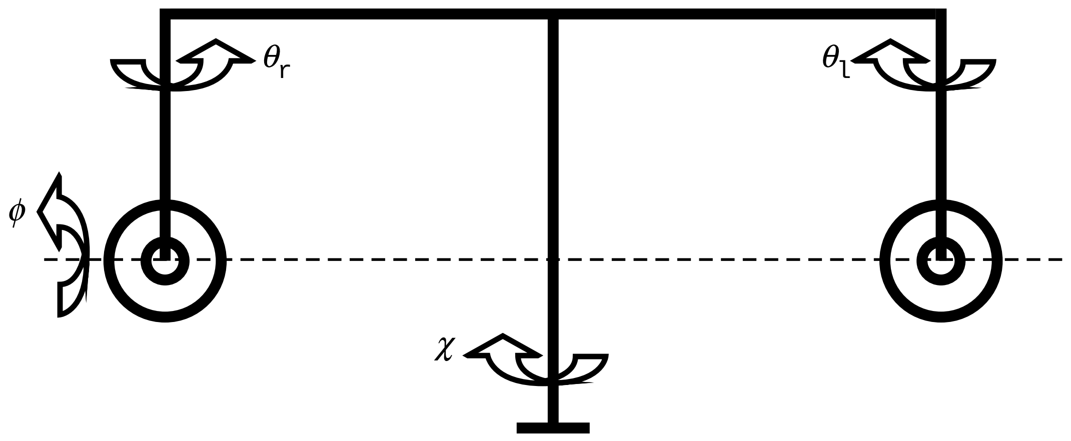


Figure 2.12: Example of the camera angles associated with a binocular rig.

the most design are the image processing components that obtain the current position of the target, and the visual controller. As mentioned previously, one of the most common methods in the literature for obtaining the target position is through the use of motion information. For example, in (Daniilidis, Krauss, Hansen & Sommer, 1998) the observed normal flow within the image, u_n, is computed. The difference between the normal flow caused by the camera motion, u_{c_n} and u_n provides an approximation for the normal flow of the object. The authors use the normal flow in conjunction with morphological processing to locate the pixels that belong to the target. Once the connected pixels representing the target have been isolated, the centroid of the pixels is computed, and is used as feedback in the control system.

In addition to their image processing algorithms, the second major component of the system presented in (Daniilidis, Krauss, Hansen & Sommer, 1998) is the controller. The task of the controller is to minimize the difference between the camera position, **c**, and the target position **o** after they have been projected onto a reference coordinate frame. The estimation and control framework performed in (Daniilidis, Krauss, Hansen & Sommer, 1998) proceeds as follows. To begin, the state of the system is described according to the vector

$$\mathbf{s} = \left(\mathbf{c}^T, \mathbf{o}^T, \mathbf{v}^T, \mathbf{a}^T\right)^T. \tag{2.17}$$

In addition to the camera and target centroid positions, the state also encompasses the instantaneous target velocity, **v**, and acceleration, **a**.

Although a multitude of controllers are available, in (Daniilidis, Krauss, Hansen & Sommer, 1998), a linear quadratic regulator is used. The LQR controller is of moderate complexity and can be used when the system is described by a set of linear differential equations. The system in (Daniilidis, Krauss, Hansen & Sommer, 1998) is cast into the LQR framework as follows. The authors employ a constant acceleration motion model to update the state

$$\mathbf{s}\left(t+1\right)=\Phi\mathbf{s}\left(t\right)+\Gamma^{T}\Delta\mathbf{u}\left(t\right), \tag{2.18}$$

where

$$\Phi=\begin{bmatrix} I_2 & 0_2 & 0_2 & 0_2 \\ 0_2 & I_2 & \Delta t I_2 & \frac{\Delta t^2}{2} I_2 \\ 0_2 & 0_2 & I_2 & \Delta t I_2 \\ 0_2 & 0_2 & 0_2 & I_2 \end{bmatrix}, \tag{2.19}$$

$\Delta\mathbf{u}\left(t\right)$ is the incremental correction of the camera position, I_2 is a 2×2 identity matrix, and 0_2 is a 2×2 zero matrix. Notice that the first term in Eq. 2.18 applies a constant acceleration motion model to the target. Thus, the target's centroid position and velocity are modified accordingly. The second term in Eq. 2.18 involves the camera motion. Hence, $\Gamma=\left[1,1,0,0,0,0,0,0\right]^T$, is used to isolate only the camera positional information in the state vector.

The last remaining aspect of the LQR controller used in (Daniilidis, Krauss, Hansen & Sommer, 1998) is that of computing $\Delta\mathbf{u}\left(t\right)$. According to the LQR controller theory, the error between the camera position and target position can be minimized if $\Delta\mathbf{u}\left(t\right)=-\mathbf{K}^T\hat{\mathbf{s}}\left(t\right)$, where $\mathbf{K}=\left[1,1,-1,-1,-\Delta t,-\Delta t,-\frac{\Delta t^2}{2},-\frac{\Delta t^2}{2}\right]^T$. Essentially, this solution indicates that the camera position should be moved to the predicted position of the target, according to the constant acceleration motion model. With that, the system is essentially complete. The camera is moved to the predicted object position using the controller and the new position of the target centroid is found using the image processing algorithms. These major two processes iterate until tracking is completed.

The block diagram of Fig. 2.11 outlines the general series of steps that almost any active vision tracker follows. Furthermore, the description of the tracker presented in (Daniilidis, Krauss, Hansen & Sommer, 1998) provides a specific example of a system that has been instantiated in the framework of Fig. 2.11. The primary and significant differences between the research in (Daniilidis, Krauss, Hansen & Sommer, 1998) and most other active vision trackers include the camera setup, the controller used, and the image processing algorithms employed. A sampling of additional noteworthy works in the field will now be briefly described.

In (Coombs & Brown, 1993), a binocular camera setup is used during active visual tracking. The use of binocular cameras offers some advantages in addition to posing some additional difficulties. An advantage of utilizing multiple cameras is that depth information regarding the scene can be obtained, which can be useful when localizing the target. On the other hand, the inclusion of another camera adds an additional degree of freedom (meaning that an additional camera angle must be controlled). In particular, in (Coombs & Brown, 1993) the problem of binocular vergence is considered, which is the point at which the two camera's optical axes intersect in the tilt plane.

Instead of using motion information to localize the target within the image, the authors identify the target's position using stereo vision and vergence theory. Specifically, the pixels immediately surrounding the vergence point have very small disparities between the left and right images. If the target is assumed to be located near the center of the field of view, it corresponds to the blob of pixels that have approximately zero disparity. The system employs three proportional-integral-derivative (PID) controllers that operate on the vergence, pan, and tilt angles directly. It should be noted that PID controllers are common in the control system literature. As the name suggests, they consist of three components. The proportional component makes adjustments based on the current error. The integral component suggests modifications to the manipulated variable based on the sum of recent errors. Finally, the derivative component makes corrections based on the current rate of change of the error. The weighted sum of these components is used to update the manipulated variable.

Another active vision system compares the relative performance of several common controllers (Papanikolopoulos, Khosla & Kanade, 1993). The image processing component of their system uses sum-of-squared difference optical flow. Errors derived from the optical flow measurements are provided as input into one of three classes of controllers that are compared: a PI controller (similar to a PID controller, only the derivative component is nulled out); a pole assignment controller; and a linear-quadratic Gaussian controller (very similar to the controller proposed in (Daniilidis, Krauss, Hansen & Sommer, 1998)). A discussion of pole assignment techniques is beyond the scope of this paper; however, the essential idea behind this class of controllers is a follows. If the system is completely state controllable (Ogata, 2002) then the "poles" of the control system (i.e. the desired state) can be set at any arbitrary location and a feedback gain matrix will exist that will allow the control system to converge to said pole.

In (Papanikolopoulos, Khosla & Kanade, 1993), the authors conclude that the type of controller to be used depends on the accuracy of the measurements. The simple PI controller is quite sensitive to noisy measurements. Thus, when measurements are noisy, more complex control systems that compensate for stochastic disturbances (e.g., pole assignment, LQG controller) are required. Another observation in (Papanikolopoulos, Khosla & Kanade, 1993) is that, rather than attempting to create more complex computer vision algorithms to reduce the nosiness of measurements, it is typically more beneficial to use a more complex controller. Gains made in measurement accuracy are typically offset by the error associated with the increased delay between object motion and the creation of an error signal.

In addition to the works cited above, at this point we would like to mention several additional contributions to the field of active visual tracking. For example, in

(Ye, Tsotsos, Harley & Bennet, 2000), 3D space is decomposed such that a minimum number of camera settings (pan, tilt, and zoom) are selected that cover the entire scene at appropriate resolutions. Tracking primarily consists of identifying the appropriate camera setting and computing the difference between the current image and a stored background model of the corresponding scene position. In (Lin & Tsai, 2000), pan and tilt motors are used to control a laser pointer as it tracks a path in a monocular, stationary image sequence. The scene consists of a plane with a specified path (a line drawn on the plane) that the laser pointer is meant to follow. The distance between the laser's resulting dot on the plane and the current point along the path are computed and is provided as input to a control system that moves the laser pointer. In (Murray, Bradshaw, McLauchlan, Reid & Sharkey, 1995), a system is developed that uses "saccades" to detect new targets of interest as they enter the field of view that are subsequently tracked using smooth camera movements. Note that a saccade is loosely defined in this review as a faster and larger camera motion that can be employed to quickly catch up to a target or to move to a new target of interest. Finally, in (Yang, Pollefeys, Welch, Frahm & Ilie, 2007) an array of four cameras and three virtual cameras is used to compute the camera motion in challenging scenes that have view-dependent appearances (e.g., scenes with reflective surfaces, specularities, etc.). The system estimates the camera's motion by computing a local linear approximation of a 6D manifold in n dimensional space (n is the number of pixels in an image). Once the local shape of the manifold has been estimated, it can be used to determine the camera motion that transpired between the acquisition of two images.

2.8 Recapitulation

In summary, this chapter has presented a number of tools that are commonly employed in one or more categories of visual trackers. We began with a discussion of initialization and detection. Subsequently, the features that are most commonly used in visual tracking were detailed. Additionally, the topics of predictive filtering and data association were discussed. The chapter concluded with the topics of coarse-to-fine processing, template updates, and finally, active vision systems. In the next chapters we will analyze the various categories of trackers in more detail.

Chapter 3

Trackers Using Discrete Features

The first class of trackers to be discussed in this review are systems that use only discrete features. By discrete features, we are referring to image structures such as discrete points, collections of edges, and lines. We will begin with a description of trackers that operate using just point-wise positional information. Subsequently, trackers that use groups of edges, including those that incorporate 3D parametric models, will be explained.

3.1 Deterministic Point Tracking

We have already discussed a certain category of point trackers in Chapter 2. Specifically, the statistical data association techniques (e.g., track-splitting filter, multiple-hypothesis algorithm, JPDAF) are all examples of statistical methods for tracking points. The rationale for including that class of trackers in Chapter 2 instead of creating a "Statistical Point Tracking" subsection here is that statistical techniques are commonly used in other classes of trackers (such as region-based blob trackers). In contrast, deterministic point trackers are rarely employed as tools in other systems. At the risk of confusing the reader, we will now return to the commonly used representation of image coordinates: $\mathbf{x} = (x, y)$. Thus, unlike the previous sections of data association and predictive filtering, $\mathbf{x}$ no longer indicates the state of a predictive filter.

Deterministic point trackers have the challenging task of following targets that all appear the same. The only distinguishing characteristic of these points is their position. Unlike their statistical counterparts that make use of principled techniques for establishing the most likely tracks, deterministic point trackers utilize more heuristic constraints. These constraints typically stipulate restrictions on the displacement or velocity of a target between consecutive frames.

The idea of performing data association amongst a set of points in a video sequence arises in a number of application domains including RADAR, cell motion analysis, bird tracking, and traffic analysis. Even in fields such as psychology, the representation of complex objects by mere points and subsequently tracking these points is sometimes considered, as evidenced by Johansson's moving light displays (Johansson, 1975). The first question that should be answered is: "where do these points come from?" In some application domains, such as RADAR or cell tracking, each target may literally represent a single image point at a given time instant. In other situations, such as bird or vehicle tracking, a point representation of a target may be obtained by performing background subtraction and then representing foreground objects with their centroids. Yet another method to obtain point-wise representations of targets is to use an interest point detector (as discussed in Chapter 2). In this last case, an overall target is likely represented as a set of points that are moving together with a rigid or non-rigid relationship.

Given the many application domains, it is no surprise that a number of papers have been published with the goal of tracking a set of points using deterministic rules (Sethi & Jain, 1987; Salari & Sethi, 1990; Rangarajan & Shah, 1991; Veenman, Reinders & Backer, 2001; Shafique & Shah, 2003). Deterministic point trackers generally have two main components that require design: the cost function and the optimization procedure. In (Veenman, Reinders & Backer, 2001), an elegant framework is proposed that encompasses the variations in the major previous works in the field. In this review, we will tend to follow the notation proposed in (Veenman, Reinders & Backer, 2001).

One of the first significant point trackers was presented in (Sethi & Jain, 1987). Prior to this work, deterministic point trackers typically performed inter-frame matching using only two frames. Thus, the strategies primarily evolved around nearest neighbor-inspired algorithms (i.e. position-based). In their work, Sethi and Jain attempted to make use of additional frames when performing correspondence. Collecting information over several frames provides at least two benefits. First, points in future frames often help resolve ambiguities at the current frame. Additionally, constrains on velocity information can be imposed rather than just position information. Of course, the downfall of such an approach is that if one is not cautious, the system will have exponential complexity.

In (Sethi & Jain, 1987), the defined distance function is based on the idea that the velocity of a point (both magnitude and direction), should change slowly. To describe this qualitative idea, the following mathematical cost function was used

$$\begin{aligned} c_{ij}^{k} = \quad & w_1 \quad \left[1 - \frac{\left(\mathbf{x}_i^t - \mathbf{x}_{\alpha_i^t}^{t-1}\right) \cdot \left(\mathbf{x}_j^{t+1} - \mathbf{x}_i^t\right)}{\left\|\mathbf{x}_i^t - \mathbf{x}_{\alpha_i^t}^{t-1}\right\| \left\|\mathbf{x}_j^{t+1} - \mathbf{x}_i^t\right\|} \right] + \\ & w_2 \quad \left[1 - 2\frac{\sqrt{\left\|\mathbf{x}_i^t - \mathbf{x}_{\alpha_i^t}^{t-1}\right\| \left\|\mathbf{x}_j^{t+1} - \mathbf{x}_i^t\right\|}}{\left\|\mathbf{x}_i^t - \mathbf{x}_{\alpha_i^t}^{t-1}\right\| + \left\|\mathbf{x}_j^{t+1} - \mathbf{x}_i^t\right\|} \right], \end{aligned} \tag{3.1}$$

where superscripts t denote the frame number and subscripts i indicate the target/measurement label. The function α_i^t provides the index of the point from the previous frame, $t-1$, to which $\mathbf{x}_i^t$ corresponds. The first term of Eq. 3.1 promotes directional coherence amongst the tracked points whereas the second term considers speed coherence. Thus, the cost criteria considers both the velocity direction and magnitude of points over a three frame time period.

Equation 3.1 provides a means of evaluating the cost of associating point i in frame t with point j in frame $t+1$. As the goal is to determine the tracks for a number of moving points, an overall cost for the set of correspondences between two frames is also required. Sethi and Jain take an obvious approach of computing the the average cost of all point associations at a particular time, which can be written as

$$C^t\left(\mathbf{A}^t, \mathbf{D}^t\right) = \left[\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{m_{t+1}} a_{ij}^t \left(c_{ij}^t\right)^z\right]^{\frac{1}{z}}, \tag{3.2}$$

where z is a parameter that can be adjusted to penalize large deviations as appropriate and M is the number of points in the scene. Additionally, $\mathbf{A}^t$ is a binary matrix that contains all point correspondences at time t and $\mathbf{D}^t$ is a cost matrix that is comprised of the elements c_{ij}^t. A single element of matrix $\mathbf{A}^t$ (i.e. a_{ij}^t) has a value of one if point i and j are deemed to correspond at frame t.

The second key component of most deterministic point tracking systems is a method of minimizing the cost function. To that end, an iterative algorithm, known as the "greedy exchange algorithm", was proposed in (Sethi & Jain, 1987). This algorithm made use of the following steps:

1. Create initial correspondences between points in frames t and $t+1$ by simply selecting the nearest neighbor. These correspondences are made across the entire video sequence.

2. Loop over each frame $t = 2 \ldots n-1$

3. For each point, i, consider changing its correspondence, c_{ij}^t, to all points other than j in frame $t+1$. Compute the cost reduction, if any, associated with making this swap.

4. Select the $i-j$ pair that resulted in the biggest cost reduction.

5. In frame $t+1$ swap i's old and new corresponding points.

6. If a swap was made, return to Step 2, otherwise terminate and return the final correspondences.

An interesting observation is that once the initial correspondences have been made using the nearest neighbor algorithm, the relationships between points in frames 1 and 2 are never altered. Thus, if correspondence errors were made at the first time instant, these errors would never be corrected. In an attempt to remedy this problem, Sethi and Jane proposed a modified greedy exchange algorithm (Sethi & Jain, 1987). This algorithm performs the forward optimization method (identical to the above algorithm) and then repeats the process going in the reverse direction (i.e. from the last frame to the first frame). Iterations are performed until no changes are made in either direction.

The ideas of Sethi and Jain provided a reasonable basis for future deterministic point trackers, but even their modified optimization algorithm has a number of limitations. Indeed, the forward-backward iterative process is not guaranteed to converge, especially if the points are dense. Additionally, the nearest neighbor initialization criterion is different from the cost criterion used during regular iterations. Veenman et al. argue that this discrepancy still causes unrepairable initial tracks to arise, even when using the modified greedy exchange algorithm (Veenman, Reinders & Backer, 2001). Furthermore, the iterative nature of this optimization algorithm makes it quite computationally expensive. Finally, this initial algorithm contained no means of dealing with track initialization or termination (i.e. targets entering or leaving the scene).

In subsequent years, the work in (Sethi & Jain, 1987) was extended by Salari and Sethi. The primary goal of this extension was to allow for point occlusions, point detection errors, and points entering or leaving the field of view (Salari & Sethi, 1990). The general idea of this work is to include "phantom points" whenever a point in the current frame can not locate a match in the next frame. Establishing a correspondence with a phantom point is not a desirable outcome and accordingly, the defined cost of doing so is high. As their experiments illustrated, the inclusion of phantom points allowed the tracker to handle occlusion events and detection errors in very simple sequences.

Following the contributions by Sethi et al., another deterministic point tracker was proposed by Rangarajan and Shah (Rangarajan & Shah, 1991). The goal of this work was to overcome the most significant issues that remained in (Salari & Sethi, 1990). In particular, the system by Rangarajan and Shah avoids initialization via error-prone nearest neighbor techniques. Furthermore, unlike Sethi's algorithms, the optimization procedure in (Rangarajan & Shah, 1991) is non-iterative and operates in polynomial time.

In describing the system in (Rangarajan & Shah, 1991), let us begin with a discussion of the cost function that is employed. Unlike the systems in (Sethi & Jain, 1987; Salari & Sethi, 1990) which assume velocity changes smoothly, Rangarajan and Shah's system assumes that a point's motion between consecutive frames is small and that the speed will remain constant. Mathematically, the cost function used is:

$$c_{ij}^{t} = \frac{\left\| \left(\mathbf{x}_i^t - \mathbf{x}_{\alpha_i^t}^{t-1} \right) - \left(\mathbf{x}_j^{t+1} - \mathbf{x}_i^t \right) \right\|}{\sum_{p=1}^{M} \sum_{q=1}^{m_{t+1}} \left\| \left(\mathbf{x}_p^t - \mathbf{x}_{\alpha_p^t}^{t-1} \right) - \left(\mathbf{x}_q^{t+1} - \mathbf{x}_p^t \right) \right\|} + \left[\frac{\left\| \mathbf{x}_j^{t+1} - \mathbf{x}_i^t \right\|}{\sum_{p=1}^{M} \sum_{q=1}^{m_{t+1}} \left\| \mathbf{x}_q^{t+1} - \mathbf{x}_p^t \right\|} \right], \tag{3.3}$$

where all variables have the same definition as in Equation 3.1. The first term in Eq. 3.3 encourages constant velocity while the second term promotes small motions.

As in most other works, in addition to defining a cost function for a single correspondence, the cost associated with all point correspondences at a single frame must also be computed. To that end, a slightly modified version of Eq. 3.2 is presented

$$C^k \left(\mathbf{A}^k, \mathbf{D}^k \right) = \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{m_{k+1}} \left(a_{ij}^t c_{ij}^t - w_1 R_a(i) - w_2 R_c(j) \right), \tag{3.4}$$

where w_1, w_2 are weights. The $R_a(i)$ function computes the average cost of all alternative correspondences (i.e. if i was not matched to j). Furthermore, $R_c(j)$, provides the average cost associated with competitors to i (i.e. if j was not matched to i). The rationale behind this overall cost is that the system should not always select a set of correspondences for a frame that lead to the lowest average cost. Instead, a more global perspective should be taken. The idea is that all point correspondences for a frame should all have roughly similar costs, c_{ij}^t. A set of correspondences where all c_{ij}^t are roughly equal is much preferable to a solution where several costs are very low but one or two are quite high. In fact, Eq. 3.4 indicates that the authors are willing to accept solutions with more consistent individual costs, even at the expensive of a slightly higher overall frame cost. This idea is somewhat analogous to minimizing the maximum error.

The optimization procedure presented in (Rangarajan & Shah, 1991) is performed in priority order. Correspondences of high priority are those that yield large

$R_a(i)+R_c(j)$ values. If incorrect correspondences for these points are made, then they will be related to points that provide a poor match. Thus, the algorithm identifies track point i and measurement j that are of highest priority and makes the appropriate association. Track i and measurement j are subsequently removed from the available pool and the process is repeated. Note that this optimization procedure is performed for just two frames (although three frames are considered when computing the velocity-based cost measure) and does not involve expensive iterative procedures like in (Sethi & Jain, 1987; Salari & Sethi, 1990). It should also be mentioned that to obtain correspondences in the first two frames, optical flow is used.

The work of (Rangarajan & Shah, 1991) definitely provides some advantages over prior works, especially in terms of computation time. However, it still has its drawbacks. For one, the inclusion of a completely different mechanism, optical flow, to perform initialization is somewhat undesirable as the framework is not uniform. Furthermore, the algorithm assumes a fixed number of points in the scene and does not allow for track termination or creation. The occlusion handling scheme is also quite crude in that when a track is not matched with a measurement, linear extrapolation is simply used. During short occlusions or missed detections such a scheme is adequate, but for long occlusions, it provides inaccurate estimates. In their work, Veenman et al. attempted to resolve some of these shortcomings.

As was mentioned previously, in addition to presenting a new point tracking system, the work of Veenman et al. (Veenman, Reinders & Backer, 2001) also proposed a common framework for most of the prior deterministic point trackers. Furthermore, this work provided a rigorous analysis between a number of point trackers and their own proposed tracker. The authors also compared their algorithm against the statistical-based multiple-hypothesis algorithm. The novel algorithm presented in (Veenman, Reinders & Backer, 2001) actually borrows many ideas from previous works, as will now be discussed.

The proposed system in (Veenman, Reinders & Backer, 2001) is very adaptable in that it can be used in conjunction with a variety of cost functions, including Eq. 3.2 and Eq. 3.4. The main contribution of their system is that optimization is performed using a standard technique known as the Hungarian algorithm [Kuhn55]. The Hungarian algorithm minimizes the following expression:

$$C = \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}w_{ij}, \tag{3.5}$$

subject to the constraints that $\sum_{i=1}^{m} a_{ij} = 1$ where $1 \leq j \leq m$, and $\sum_{j=1}^{m} a_{ij} = 1$ where $1 \leq i \leq m$. Furthermore, all variables a_{ij} are binary. The first constraint insures that a particular measurement j, at time $t+1$, has only one match at time t. The second criterion ensures that a particular target, i, is only matched with a single measurement in frame $t+1$. Thus, the Hungarian algorithm can identify the minimum cost correspondence set, where the weights are defined as the overall correspondence cost between two frames (e.g., either Eq. 3.2 or Eq. 3.4).

The Hungarian algorithm provides an elegant, standard optimization technique, but it assumes that the number of tracks and thus, measurements within the current frame remains constant. In point-based tracking situations, this assumption is frequently violated due to false/missing measurements, occlusions, track starts, and track terminations. Veenman et al. elect to forgo presenting a solution to the issue of scene points entering or exiting the field of view. The rationale behind this decision is that the challenges of dealing with occlusion and that of track initialization/deletion have conflicting requirements, making this problem even more difficult. Accordingly, they leave this task as future work. Nonetheless, to utilize the Hungarian algorithm, a mechanism must be employed to ensure that the number of tracks/measurements remains constant even in the face of incorrect measurements and occlusions. In response to this requirement, the authors introduce so-called "false tracks" when measurements are observed that do not correspond well to any track. Furthermore, "slave measurements" are introduced to handle the situation where tracks in frame t can not identify effective measurements in frame $t + 1$. These slave measurements are somewhat similar to the phantom points proposed in (Salari & Sethi, 1990). Having supplemented the true tracks and measurements with the false tracks and slave measurements, all matrices in Eq. 3.5 are square, enabling the Hungarian algorithm to operate correctly. This proposed algorithm is termed the "greedy optimal assignment" (GOA) tracker by the authors.

To achieve self-initialization during the first two frames, the authors present a modified version of the GOA tracker that performs batch-wise bidirectional optimization in a similar spirit to (Sethi & Jain, 1987; Salari & Sethi, 1990). The Hungarian algorithm is still used for determining correspondences in the first two frames; however, a modified cost function based on nearest neighbor distances is used. The authors argue that unlike the algorithm proposed in (Sethi & Jain, 1987; Salari & Sethi, 1990), one forward-backward iteration of the modified GOA tracker results in convergence.

Throughout their impressive suite of tests on real (seeds rotating in a dish) and synthetic data, the GOA tracker performs very well, almost universally outperforming the trackers presented in (Sethi & Jain, 1987; Salari & Sethi, 1990; Rangarajan & Shah, 1991) and a multiple-hypothesis tracker. The algorithms were tested using varying point densities as well as on their ability to cope with occlusions, and their parameters' sensitivity. Furthermore, tests were performed to measure the initialization performance of the respective systems for the first two frames. Interestingly, the authors elected to not compare against (Rangarajan & Shah, 1991)'s optical flow-based initialization method, arguing that it uses additional information during this stage.

Overall, the authors conclude that their GOA tracker is superior to that of Rangarajan and Shah's both in terms of qualitative accuracy and computational performance. The algorithms presented in (Sethi & Jain, 1987; Salari & Sethi, 1990) are also argued to be inferior with regards to accuracy and speed. Furthermore, these earlier algorithms do not support occlusion handling. One area, however, where the algorithms of (Sethi & Jain, 1987; Salari & Sethi, 1990) are superior to the GOA tracker is that they provide mechanisms for tracking variable number of points.

The work in (Veenman, Reinders & Backer, 2001) was undoubtedly very good, in regards to its unifying framework, extensive test set, and novel GOA tracker. Nonetheless, the GOA certainly is still not perfect. As mentioned, this tracker can only follow a fixed number of points. Additionally, the Hungarian algorithm-based optimization can only be performed using a pair of frames. Oftentimes, five or ten frames into the future, measurements will be observed that reveal that the correspondences decided upon for the current pair of frames are inaccurate. The GOA tracker provides no means of incorporating this global, longer-term information. The point tracker presented in (Shafique & Shah, 2003) does leverage information over additional frames.

In (Shafique & Shah, 2003), the optimization is performed using a graph theory approach. As Fig. 3.1 shows, the measurements in each frame can be thought of as a set of vertices (measurements originating from a particular frame are aligned in a column). At the conclusion of the previous frame, a set of edges will exist that connect measurements in each of the frames to subsequent measurements observed in other frames. Each edge has a weight that indicates the "gain" incurred by associating the two points. This gain function promotes directional and speed consistency in a similar manner to Eq. 3.1. However, in this case the function is formulated as a gain so that maximization is performed rather than minimization. Also note that an edge need not extend from a measurement at time t to a measurement at time $t+1$. Instead, measurements at time t may be connected to a vertex at frame $t+n$, or no other vertex at all. Freedom of this nature is attained because optimization is being performed across a wider range of frames.

When a new frame is observed, the next iteration of optimizations is performed. This process begins by taking the set of edges decided upon at the conclusion of the last frame and supplementing them with additional edges. This supplemental graph is known as an "extensions digraph", an example of which is shown in Fig. 3.1B. Of course, the most obvious edges extend from vertices at frame t to vertices at frame $t+1$. However, supplemental vertices are also added that extend from frame $t-n$, where n is an integer that varies from one to some maximum value. These edges that are added from the older frames (frames $t-n$) are included to consider the possibility that incorrect correspondences were made due to the limited information previously available. Once this "extended" graph has been constructed, the set of edges that maximize the gain must be selected. Fortunately, using graph theory it is known that for a directed, acyclic graph, the maximum path cover can

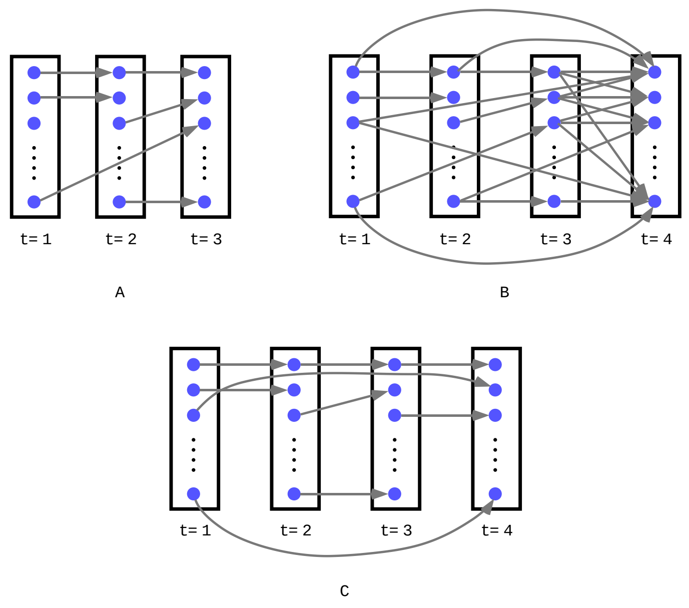


Figure 3.1: Graph representation used in (Shafique & Shah, 2003). Each circle represents a measurement and connected measurements indicate a target track. A single frame is represented by a rectangle. A) The hypothesized tracks for frame $t = 3$ B) Extension digraph at $t = 4$ C) Maximum path cover of the extension digraph at $t = 4$.

be computed in polynomial time (Shafique & Shah, 2003). Due to its construction, the extended graph in this work is directed and acyclic. Corresponding, this theorem can be employed to determine the set of edges that maximize the gain. An example maximum path cover of the extension digraph in Fig. 3.1B is shown in Fig. 3.1C.

The final issue to be discussed with respect to Shafique et al.'s tracker is the initialization procedure for the first two frames. Once again, a forward-backward procedure is used. However, unlike in (Veenman, Reinders & Backer, 2001) where the forward-backward procedure is conducted over the entire sequence, in (Shafique & Shah, 2003) only the first k frames are utilized. Although this will introduce a constant delay in processing, this modification makes the system in (Shafique & Shah, 2003) more amenable to online processing than the GOA tracker. Due to its initialization strategy, the GOA tracker is essentially forced to operate in a batch manner.

In terms of performance, using a sliding window of size five (i.e. the maximum value for the variable, n, seen above), the authors show that their tracker equals or betters the performance of the GOA tracker when the number of points tracked remains constant and the number of noise points is minimal. However, when the level of noise is increased, the proposed tracker is relatively unaffected whereas the GOA's performance suffers. Finally, the proposed tracker performs almost as well when points are allowed to enter and leave the scene — a situation which the GOA tracker was not designed to accommodate.

Table 3.1 summarizes the deterministic point trackers presented in this section. In the table, S&J, S&S, R&S, GOA, and KS&MS refer to the systems in (Sethi & Jain, 1987), (Salari & Sethi, 1990), (Rangarajan & Shah, 1991), (Veenman, Reinders & Backer, 2001), and (Shafique & Shah, 2003), respectively. It should be noted that this summary table, and other similar tables presented in this review, can be somewhat deceptive. In Table 3.1, we indicate whether or not the proposed system attempted to solve, or presented a partial solution to a particular subproblem. In other words, an entry of "Yes" does not indicate that the system completely solved the sub-problem. For example, the system in (Salari & Sethi, 1990) did not completely solve the problem of automatic initialization. The system proposed a solution for automatic initialization, but subsequent works (Veenman, Reinders & Backer, 2001) have shown that the automatic initialization scheme in (Salari & Sethi, 1990) is actually quite flawed.

Table 3.1: Summary of Deterministic Point Trackers

Algorithm	Track Termination and Initialization (i.e. variable number of targets)	Automatic Initialization	Track Continuation (occlusion or measurement errors)
S&J	No	Yes	No
S&S	Yes	Yes	Yes
R&S	No	Yes	Yes
GOA	No	Yes	Yes
KS&MS	Yes	Yes	Yes

3.2 Tracking Groups of Edges

In addition to representing a target as a single point descriptor, edge pixels can also be grouped into higher-level structures. This grouping can be done explicitly (e.g., grouping edge pixels into a line) or implicitly (e.g., comparing a set of edge pixels against the predicted position of a line). As just alluded to, the most common grouping of edge pixels is into lines. Intuitively, focusing on lines appears to be a logical approach as many man-made objects in our world are based on a Cartesian grid system and are composed of numerous straight lines. However, other trackers have explored the grouping of edge pixels into arbitrary shapes. The common theme throughout the section is that all trackers perform edge detection and then group these edges in some fashion to follow the object of interest. For the purposes of this review, we subdivide this category into two classes. The difference between these two classes is whether or not the tracker makes use of a three dimensional object model.

3.2.1 Model-Free Approaches

This section will discuss the set of trackers that use a set of edges to follow a particular target of interest while not making use of a 3D object model. The majority of the trackers in this section use edge pixels to track lines within the image sequence (Crowley, Stelmaszyk & Discours, 1988; Deriche & Faugeras, 1991; Zhang & Faugeras, 1992). However, one final work that is discussed shows that arbitrary shapes can be tracked using a collection of edge pixels without utilizing a 3D model (Huttenlocher, Noh & Rucklidge, 1993).

The model-free line trackers discussed in this section share a very similar approach. In general, they determine a particular line representation that will be used in their system. With the representation in place, tracking is typically performed using a Kalman filter or a bank of Kalman filters (one for each line parameter). Each system must design a search strategy for identifying the image line that most closely corresponds to the predicted line positions. Finally, the parameters of the predicted line are optimally combined with the parameters of the corresponding image line. This, fairly standard, Kalman filter-based tracking procedure is repeated for all frames of the video sequence.

An early work by Crowley et al. (Crowley, Stelmaszyk & Discours, 1988) follows the above outline quite closely. As Figure 3.2 shows, the particular line representation in this work consists of four parameters:

- c, the perpendicular distance from the origin to the line
- d, the distance from the line's midpoint to the perpendicular intercept from the origin
- θ, the orientation of the line with respect to the x-axis
- h, the half-length of the line

The authors motivate this particular representation as it is known that the end points and length of a detected image line are often unreliable. These measurements are unreliable because oftentimes continuous lines in one frame may become fragmented in subsequent frames due to noise or occlusion. It should be noted, however, that in situations where fragmentations occur, the orientation and perpendicular distance of the resulting line segments will remain unchanged from that of the original line. Thus, θ and c are much more robust and consistent parameters. Another motivating factor behind this representation is that even though c and d are dependent upon

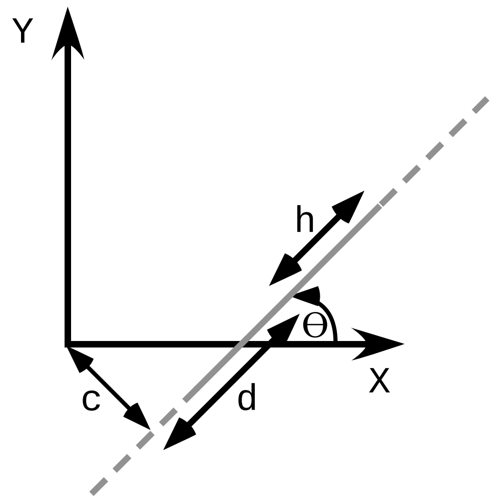


Figure 3.2: Line parameterization used in (Crowley, Stelmaszyk & Discours, 1988).

θ, the authors argue that for most practical situations they can be assumed to be independent. As a result of this assumption, the authors can deploy a separate Kalman filter for each line parameter.

Hence, the tracker in (Crowley, Stelmaszyk & Discours, 1988) makes use of four Kalman filters. In the Kalman filter framework, the state of each line parameter is represented by the parameter itself and the rate of change of the parameter. Specifically, for the Kalman filter controlling perpendicular distance, the state is represented as $\mathbf{x}_t = \{c_t, c_t^{'}\}$, where $c_t^{'}$ is the temporal derivative. Additionally, the filter must update the variance and covariance of the line parameter and its derivative.

When predicting the line parameters in a subsequent frame, the authors assume a very simple motion model with no acceleration. The state estimate equation, Eq. 2.5, for the trackers in this work can be written as

$$\hat{\mathbf{x}}_t^{-} = \mathbf{A}\hat{\mathbf{x}}_{t-1}, \tag{3.6}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \tag{3.7}$$

and Δt is the time interval between frames, $\mathbf{x}_t$ once again refers to the state at time t, and the superscript "$-$" indicates that this is the predicted state prior to updating via measurements. As there will undoubtedly be (at a minimum) small accelerations, the parameter's variance includes a term that models the acceleration as noise.

In this work, a fairly heuristic approach is taken for matching predicted lines to image lines. For each predicted line, every image line is considered in turn. Three comparisons are made between the predicted and measured lines. First, the orientations and perpendicular distances of the two lines are compared to ensure that they are within three standard deviations of one another. These two initial checks are based on the most reliable line parameters and help to eliminate ineligible measured lines quickly. For lines that pass these first two tests, a third test is performed to check how much of the lines "overlap" with one another. If all three tests are cleared, the Mahalanobis distance is computed between the predicted and image line. The image line that yields the smallest distance (and has a distance less than a maximum threshold) is matched to the predicted line.

The final stage of the Kalman filter tracking framework in (Crowley, Stelmaszyk & Discours, 1988) involves combining the predicted position and the measurement. As the authors optimized their system for speed (on ancient, by today's standards, hardware) they made use of a simplified version of the Kalman gain

$$\mathbf{K} = \left[\begin{array}{cc} \frac{\sigma_s^2}{\sigma_s^2+\sigma_o^2} & \frac{\sigma_{ss'}^2}{\sigma_s^2+\sigma_o^2} \end{array} \right]^T, \tag{3.8}$$

where σ_s^2 is the variance of state variable, σ_o^2 is the variance of the measurement, and $\sigma_{ss'}^2$ is the covariance between the variable and its derivative.

To handle short occlusions, the authors assign, to each tracked line, a discrete "health". The health of the tracked line can range from 0 to 5. When a line is first initialized in the video sequence, it has a health of 3. Every time a measured edge is found that corresponds to the predicted position of a line, the health of that line is incremented (to a maximum value of 5). In contrast, when no correspondence is made, the health is decremented. As soon as the health drops below zero, the tracked line is removed from the list of active lines. This occlusion handling technique is quite ad hoc (no rationale is provided for deciding that the maximum number of occlusion frames should be 5), but at least a simple mechanism is included for handling occlusion.

For its time, this paper was undoubtedly influential as it helped pave the way for subsequent line trackers. However, its performance and design are very basic, largely because of the hardware being used at the time. The matching method, occlusion handling, and Kalman update procedures were clearly designed to keep computations to a minimum at the expensive of keeping the tracker simple.

A subsequent work by Deriche and Faugeras (Deriche & Faugeras, 1991) extends the general approach presented in (Crowley, Stelmaszyk & Discours, 1988). In this work, the authors consider various parameterizations of lines, including one that is almost identical to that used in (Crowley, Stelmaszyk & Discours, 1988). Representations such as the one seen in (Crowley, Stelmaszyk & Discours, 1988) have the problem that the uncertainty of line segments varies, depending on their position in the image. Specifically, two line segments that have the same orientation and length can have very different uncertainties for c and d, depending on their position (Deriche

& Faugeras, 1991). As a result of this discovery, the line representation used in this work consists of $[x_m, y_m, \theta, l]$ where (x_m, y_m) is the mid-point of the line, θ is the line's orientation, and l is the length of the line. This new representation also completely decorrelates the four line parameters.

Tracking via Kalman filters proceeds in a very similar fashion to that described for (Crowley, Stelmaszyk & Discours, 1988). In the Deriche and Faugeras's work, however, the state vector is modified to include the acceleration of the parameter (in addition to the parameter itself and its rate of change). The state prediction equation assumes constant acceleration. When matching predicted lines to lines within the image, the authors directly compute the Mahalanobis distance rather than first employing ad hoc rules. The Mahalanobis distances between the predicted and measured lines are computed for each of the four line parameters. This process is performed for each of the measured lines in the image and the line that yields the lowest sum is selected as the match. More sophisticated results are presented in this work than what was seen in (Crowley, Stelmaszyk & Discours, 1988). In (Crowley, Stelmaszyk & Discours, 1988) tracking of high contrast lines in a very constrained, laboratory environment were shown. In this subsequent work, more realistic results are provided where tracking is performed on the outlines of walls and doors in a corridor.

The two works described previously were in fact, quite similar to one another. Undoubtedly, Deriche and Faugeras made a number of improvements to the initial line tracking algorithm, but these improvements were mostly incremental and were improvements to sacrifices made by Crowley et al. to ensure that their tracker could run on the hardware at the time. Another evolution in the line tracking paradigm was seen in (Zhang & Faugeras, 1992). Much of the line tracking procedures observed in the prior two works once again remain unchanged in this work by Zhang and Faugeras. However, the most novel aspect of this system is that rather than tracking the lines in the image plane, stereo vision is used to follow lines in three dimensional space.

As the tracker in (Zhang & Faugeras, 1992) follows points in 3D space, a new line representation is required. Here, the authors parameterize a line using five values. Two of the parameters are used for describing a line's orientation (using an Euler angle representation) and the final three parameters are for the line's mid-point in three dimensional space. Special attention is paid when representing the mid-point of a line. In particular, in a noise-free environment, the mid-point could be computed according to $M = \frac{(M_1+M_2)}{2}$, where M_1 and M_2 are the endpoints. However, as was described previously, a line's mid-point and end points are not as reliable as its orientation. Accordingly, an additional noise term is included which randomly displaces the midpoint along the direction of the line. This noise models the additional uncertainty associated with the mid point.

The 3D line tracker in (Zhang & Faugeras, 1992) makes use of an extended Kalman filter. The EKF state is defined as follows, $\mathbf{x}_t = \{\omega_i, v_i, a_i\}^T$, which correspond to the angular velocity, translational velocity, and translational acceleration at time

t, respectively. A constant angular velocity and constant translational acceleration motion model are assumed during predictive state estimation. A measurement is composed of two lines and can be written as follows: $\mathbf{z}_t = \{\psi_1^t, m_1^t, \psi_2^t, m_2^t\}$. The first line in the measurement corresponds to the predicted position of a line obtained during estimation, while the second corresponds to the observed image line. Note that a system of equations is required to determine the predicted line parameters (ψ_1^t and m_1^t) based on the predicted state estimate because the state and measurement of the filter are of different units. The equations that relate state to measurement are nonlinear, necessitating the use of the EKF.

Given that a measurement consists of the predicted and measured lines, the question remains: "How is line matching performed?" As has been seen with other line trackers, once again the Mahalanobis is employed. Rather than matching a predicted line with every line obtained from the images, a bucketing strategy is employed to only consider lines that are in the vicinity of the predicted location. Bucketing is performed purely for computational considerations.

Another novel aspect of this work is the strategy used once the Mahalanobis distances have been computed. The line trackers discussed previously only matched a single image line with a given predicted line. This approach is certainly logical. However, in situations when there are numerous lines in close proximity, the data association between lines may not be quite so clear-cut. As we have discussed with respect to point trackers, oftentimes it is beneficial to maintain multiple hypotheses and make a final decision once more information becomes available. The line tracker in (Zhang & Faugeras, 1992) follows this logic. If multiple measured lines yield a Mahalanobis distances that is lower than a threshold, the two lines with the lowest scores are retained. All lines that are tracked have a likelihood associated with them, representing the confidence that the line is correct. Once this likelihood drops sufficiently low, an incorrect hypothesis will be eliminated. Furthermore, this likelihood allows the system to deal with occlusions. The likelihood is decreased if no image match is found for a tracked line in a given frame. If the measurements are not obtained for just a few frames the line can be tracked once it reappears, thanks to the predictive component of the system. However, if the line is occluded for a great number of frames, the likelihood will eventually drop too low and the line will be eliminated.

The results shown in this work are quite impressive — illustrating that 3D line tracking can be achieved when either the camera or scene objects are moving. In some of the sequences shown, the number of lines being tracked is large and the scene is quite cluttered. This research makes several advances over the work of (Deriche & Faugeras, 1991), the most significant of which are: the extension of tracking into the 3D; the propagation of multiple hypothesis; a more principled approach to handling occlusion; and a more thorough test set. The system also incorporates a completely novel mechanism that groups lines with similar motion characteristics into higher-level objects. Specifically, lines are grouped into high-level objects based on their kinematic parameters. If the kinematic parameters of two lines are within a certain Mahalanobis distance of one another, they are assumed to belong to the same object. An overall estimate of the high-level object's kinematic parameters is maintained and refined as lines are added. Of course, the system in (Zhang & Faugeras, 1992) is not perfect, with one of the most significant downsides being that it is incorporating the use of stereo cameras which are not as prevalent and are more costly than a monocular camera.

All of the 3D model free trackers up until this point have focused on grouping edges into lines and subsequently tracking those lines. However, there is nothing to say that edge pixels cannot be used in other manners to create a model free tracker. One example of such as system was presented in (Huttenlocher, Noh & Rucklidge, 1993). The main idea is to disregard object motion (any arbitrary motion is allowable) and perform matching solely based on the object shape. The shape of the object is represented as the edge pixels that define the target's outline and internal structures. Details of the system in (Huttenlocher, Noh & Rucklidge, 1993) will be provided next.

Assuming that a template binary edge map is available for the target, the goal is to find the set of image edges whose shape most closely resembles that of the template. The authors have the novel idea of matching object shape (represented as a set of edges) using the Hausdorff metric. The Hausdorff metric is defined as

$$H\left(P,Q\right) = \max\left(h\left(P,Q\right), h\left(Q,P\right)\right), \tag{3.9}$$

where

$$h\left(P,Q\right) = \max_{\mathbf{p}\in P}\min_{\mathbf{q}\in Q}\left\|\mathbf{p}-\mathbf{q}\right\|. \tag{3.10}$$

Additionally, P and Q are the point sets to be compared. Equation 3.10 considers each point $\mathbf{p}$ and computes the distance to the closest point in the other point set Q. The point $\mathbf{p}$ that yields the largest distance has its distance returned by the function. Note also that Eq. 3.10 is not symmetric. Thus, Eq. 3.9 computes the overall Hausdorff metric by allowing each point set to take on both roles as matcher and matched.

A shortcoming of the Hausdorff metric in its current form is its sensitivity to outlier points (due to the max operation in Eq. 3.10). Specifically, this function can be adversely affected by a single noise point in one data set that has no good correspondence in the other set. To circumvent this challenge, in (Huttenlocher, Noh & Rucklidge, 1993), Eq. 3.10 is modified as follows

$$h_K\left(P,Q\right) = K^{th}_{\mathbf{p}\in P}\min_{\mathbf{q}\in Q}\left\|\mathbf{p}-\mathbf{q}\right\|. \tag{3.11}$$

Rather than selecting the point, $\mathbf{p}$, that has the largest distance, this new measure selects the point that yields the K^{th} smallest distance. If K is selected to be half the total number of points, then this new measure provides the median distance, a more robust statistic than the max value. This new matching measure also has a number of qualitative properties that intuitively make it a reasonable choice for tracking. In particular, $h_K(P, Q) = d$ indicates that there are at least K points in P that match arbitrary points in Q with distances less than d. Furthermore, this new distance measure also allows a portion of the points in P to have very poor matches to points in Q. For tracking, this could be very useful in occlusion situations.

With this background in place, the goals now are to identify the image region that will minimize the Hausdorff-based cost and to update the target model throughout the sequence. The authors utilize the asymmetry of the Hausdorff metric to their advantage for these two purposes, as will be described next.

To identify the target's position the authors introduce an arbitrary group of transformations, G, that are applied to the template point set. This group of transformations could simply be all translational transformations or alternatively, it could include affine, projective, or other transformations. The distance metric is then computed a number of times (once for each allowable transformations) and the transformation that minimizes the cost corresponds to the target position. Making use of the asymmetry of the Hausdorff metric, the authors define the following distance function when locating the target's position in the current frame

$$d = \min_{g \in G} h_K\left(g\left(M_t\right), I_{t+1}\right), \tag{3.12}$$

where M_t is the set of point describing the target template and I_{t+1} is the point set from the current frame. Note that this function indicates that there are at least K points from the transformed template, $g(M_t)$, that are of a distance less than or equal to d to arbitrary points in the current image I_{t+1}. In this particular work, the authors simply elected to use the set of translations for G. Thus, after computing Eq. 3.12, their system has identified the location within the current image that best matches the template model.

The second aspect of the proposed tracking system in (Huttenlocher, Noh & Rucklidge, 1993) is the method of updating the template. Again, the authors take the novel approach of using the asymmetry of the Hausdorff metric to achieve this goal. In this case, the goal of the system is to identify the set of edge points in the image, I_{t+1} that are in agreement with the transformed template model, $g^*(M_t)$ identified using Eq. 3.12. This set of edge points identified in I_{t+1} becomes the new target model. These ideas can be expressed more formally as

$$M_{t+1} = \{q \in I_{t+1} \min_{p \in M_t} \|g^*(p) - q\| \leq \delta\}, \tag{3.13}$$

where δ is the matching distance threshold.

These equations for the location and model updates define the fundamental ideas of the tracker in (Huttenlocher, Noh & Rucklidge, 1993). The authors do mention several extensions to the standard algorithm to improve its performance. For example, to eliminate the effects of edges caused by background clutter, a temporal differencing approach is included to remove stationary edges. Simple mechanisms can also be included to relocate a target once it has been lost (by searching for old template models) and to distinguish between targets in the scene that have a similar shape (using simple motion models).

At its time of publication, this work provided very respectable tracking results and an extremely novel solution. In fact, to this day, this is a very novel approach that has received limited attention or continued study after this initial work. There are some shortcomings of this tracker, of course. Manual initialization is necessary and the search method for identifying the target's position has not been optimized for speed. There is always a trade-off between increasing processing speed through prediction and performing an exhaustive search for the best fit in the image. The system proposed in (Huttenlocher, Noh & Rucklidge, 1993) uses the latter approach because the rest of their algorithm is quite efficient. However, if a larger set of transformations were used (e.g., affine), computation time may become a problem. Modestly reducing the search area might be a beneficial approach. Additionally, although a simple mechanism was proposed for dealing with multiple targets with similar shapes, the approach would undoubtedly fail if a large number of similar targets were moving within a small area.

We should also quickly mention another very novel work that considered the tracking of a collection of interest points. Specifically, in (Torresani, Yang, Alexander & Bregler, 2001) the authors show that a collection of feature points can be tracked using so-called "rank constraints". These constraints are placed on a matrix that describes the optical flow of a set of 2D image interest points. A basis set of point tracks can be constructed using a subset of the most reliable feature points. Using the rank constraints in conjunction with the basis set, the optical flow can be computed for the entire set of feature points (even those that could not be reliably tracked in isolation). Thus, the end result is a system that can track "corner" interest points (discussed in Chapter 2) as well as the less reliable "ridge-like" interest points (also discussed in Chapter 2).

To summarize, this section has concentrated on systems that track targets using edges but no three dimensional object model. Most of these systems elected to merely track lines (presumably an overall object would consist of multiple lines) by first grouping the edges into lines and then making use of a predictive filter framework. The systems in (Zhang & Faugeras, 1992) and (Huttenlocher, Noh & Rucklidge, 1993) could actually track a complete target and not just a set of individual lines. In (Zhang & Faugeras, 1992) overall target tracking was attained by grouping lines with similar motion properties. On the other hand, in (Huttenlocher, Noh & Rucklidge, 1993), a completely different methodology was employed where the target's shape was represented by a set of edges and tracked by performing frame-to-frame matching.

In the next section, our discussion will continue with more systems that use edge and line features. However, these upcoming trackers make use of 3D object models during tracking.

3.2.2 3D Model-Based Approaches

In some situations, tracker designers are fortunate enough to have 3D models of the target. These 3D models can be obtained through CAD, range-finders, or using manual methods. In this section, we will discuss feature-based 3D model trackers that match 2D features to 3D models and point out how these trackers are related to the problem of detection. Typically, these tracking and detection systems use feature sets that include edge or line-based features. We will conclude this section with a brief discussion of a more recent approach, where 3D models are matched to 3D data. Another extensive review paper on 3D model-based approaches can be found in (Fua & Lepetit, 2005).

Feature-Based Approaches

A popular approach to tracking in the early 1990s was to use discrete features, such as groups of edges, curves, or lines, and optimally align the projection of a 3D target model with these image domain features. Conceptually, the core problem is that of obtaining the transformation between the object coordinate frame and the image coordinate frame. To fully define this transformation, six parameters must be estimated — three rotational parameters and three translational parameters. Most of the works in this field follow three, generic steps: (1) image feature extraction; (2) establish correspondence between model features and image features; (3) pose estimation using the established correspondences in Step 2. Throughout this section we will explain these three steps in more detail by making reference to specific works in the literature. We will begin with a clear and illustrative paper on vehicle tracking presented by Koller et al. (Koller, Daniilidis & Nagel, 1993).

In (Koller, Daniilidis & Nagel, 1993), the authors' goal was to track vehicles in realistic scenes. Many of the techniques used with the model-free line trackers will once again be used with the 3D model-based trackers (and, in fact, throughout the rest of this paper). Specifically, a matching strategy is still required for comparing image data with other data. As alluded to already, with 3D model trackers, this "other data" stems from the projection of a 3D model. In addition to the three main stages listed above, many 3D model-based trackers also employ predictive filters (e.g., Kalman filters) to restrict the search range. In fact, the vehicle tracker in (Koller, Daniilidis & Nagel, 1993) does just that.

The first building block in any 3D model-based tracker is the 3D model itself. The object model, of course, defines and restricts the type of object that will be followed throughout a video. In (Koller, Daniilidis & Nagel, 1993), the authors use a model of a car that has 12 lengths associated with it. By varying the length of these parameters, vehicles of different sizes and shapes can be tracked. Unfortunately, once the lengths are manually set during initialization, these size parameters cannot be changed. Hence, during tracking, the vehicle model is rigid and the goal of tracking is merely to determine the position, p_o, and orientation, ϕ_o of the 3D model.

As mentioned previously, one of the major building blocks of 3D model-based trackers is an algorithm for establishing correspondences between the measured image features (in the case of (Koller, Daniilidis & Nagel, 1993), image edges) and the projected lines of the model. In (Koller, Daniilidis & Nagel, 1993), the matching strategy is almost identical to that used in (Deriche & Faugeras, 1991). Specifically, edge pixels in the image are first grouped to form lines. In a similar manner to (Deriche & Faugeras, 1991), lines are parameterized using the midpoint, orientation, and length. Finally, the Mahalanobis distance is used when comparing a particular projected model line with an image line. Correspondence is established between the pair of lines that provide the lowest distance score.

When a particular pose of the 3D model is assumed, the matching strategy between image edges and projected model lines is almost identical to the approach seen in (Deriche & Faugeras, 1991). However, this data association stage is only a part of the overall matching algorithm of Koller et al.'s tracker. In (Deriche & Faugeras, 1991), the predicted position of a line (obtained through Kalman filtering) was being compared against the image data. In this case, we are comparing the model projection against the image data. Here we are completely free to adjust the parameters of the 3D model to see if it can be brought into closer alignment with the image data. This task of optimizing the model parameters while holding correspondences constant is the third stage mentioned above, which we term pose estimation.

In the work of Koller et al., the goal of pose estimation is to minimize the overall Mahalanobis distance between corresponding lines. The optimization algorithm used in (Koller, Daniilidis & Nagel, 1993) is similar to those techniques originally proposed by Lowe (Lowe, 1991, 1992). Specifically, let $\mathbf{p}^i$ be the set of model parameters during iteration i. The ideal goal of this stage is to find the parameter adjustments, $\mathbf{g}$, that reduce an alignment residual error, $\mathbf{e}$ to zero. Assuming local linearity, the change in the overall error caused by modifying a particular parameter, g_i, can be computed. Specifically, the error can be quantified by multiplying the change in the parameter value by the partial derivative of the error with respect to that parameter. To capture how all parameter adjustments affect the error in one compact equation we can use

$$\mathbf{J}\mathbf{g} = \mathbf{e}, \tag{3.14}$$

where $\mathbf{J}$ is the Jacobian matrix defined as

$$J_{ij} = \frac{\partial e_i}{\partial g_j}. \tag{3.15}$$

In an ideal world, parameter adjustments could be made that would make the residual error vanish. This result is not feasible in practice and instead, an over-constrained system of equations is constructed where the goal is simply to minimize the residual. Mathematically, this minimization is expressed as:

$$\min \|\mathbf{Jg} - \mathbf{e}\|^2 . \tag{3.16}$$

The parameter adjustments, $\mathbf{g}$, that minimize this squared error can be found using standard linear least squares techniques.

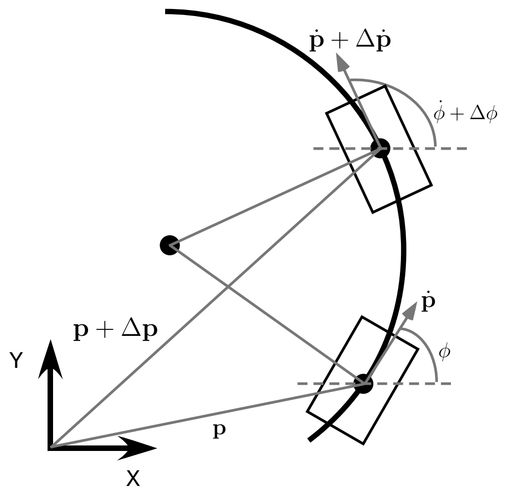
The final component of the vehicle tracker that must be discussed is the predictive filter. The filters used in 3D model-based trackers are typically just standard Kalman filters or one of its numerous variants. Chapter 2 provided a detailed description of Kalman filters. Thus in this section, only pertinent details regarding the predictive filtering that are implementation-specific will be provided.

In (Koller, Daniilidis & Nagel, 1993), the state vector for the filter consists of the position, orientation, and velocity magnitudes (translational and angular) of the car. Explicitly, the state can be written as $\mathbf{x}_t = [p_{x,t}, p_{y,t}, \phi_t, v_t, \omega_t]$. State estimation is achieved via a constant translational and angular velocity motion model. This circular motion model is shown in Fig. 3.3. The figure shows the three parameters that must be estimated to align the 3D car model with the image data — the vehicle's center position $(p_{x,t}, p_{y,t})$ and the orientation of its principle axis ϕ_t. Given that constant translational and angular velocities are assumed, the rate of change of position and orientation are defined as

$$\begin{aligned} \dot{p}_{x,t} &= v \cos \phi \\ \dot{y}_{y,t} &= v \sin \phi \\ \dot{\phi}_t &= \omega, \end{aligned} \tag{3.17}$$

where v and ω are the constant translational and angular velocities, respectively. It appears the significant benefit of adopting the circular motion model is to more accurately represent the motion of the cars for the videos that are under analysis. Specifically, the sequences considered in this paper primarily involve cars going around corners (e.g., cars turning at traffic lights and cars driving in circular trajectories in parking lots).

Continuing with the Kalman filter framework, the measurements are the final set of image lines obtained during correspondence and pose estimation. The measurement equation of the filter must transform state variables into measurement variables



$\dot{\mathbf{p}} + \Delta\dot{\mathbf{p}}$
$\dot{\phi} + \Delta\phi$
$\dot{\mathbf{p}}$
ϕ
$\mathbf{p} + \Delta\mathbf{p}$
Y
$\mathbf{p}$
X

Figure 3.3: Circular motion model for cars used in (Koller, Daniilidis & Nagel, 1993).

so that the two quantities can be compared. This is not a trivial task in (Koller, Daniilidis & Nagel, 1993) as the state represents the overall position, orientation, and velocity of the car model while the measurements are the optimal image lines! Thus, to compare the state and measurements, nonlinear transformations are used to map model points into the world coordinate system and subsequently into the image plane, necessitating the use of an extended Kalman filter or an IEKF (both of which are explored in this work).

With the main building blocks in place, the tracker of Koller et al. is essentially completely defined. The performance results of this system are qualitatively quite good. Videos showing realistic traffic footage from a parking lot and traffic intersection are shown. Even in the presence of some shadows and clutter, the models are aligned closely with the vehicles. There are, however, a number of downfalls to this approach. To begin, camera calibration must be performed beforehand so that mapping between model, world, and image points can be performed. Another important note is that throughout their experiments, the authors modified various system parameters. This tweaking makes it appear as if the parameters are not universal and need to be tuned for particular types of videos.

There are several other 3D model-based trackers that follow the same general approach as seen in Koller et al.'s work (Lowe, 1991, 1992; Simon & Berger, 1998; Verghese, Gale & Dyer, 1990; Shang, Jasiobedzki & Greenspan, 2007). The approach in (Lowe, 1991, 1992), for example, is very similar to that seen in (Koller, Daniilidis & Nagel, 1993). As a result, we will primarily discuss the correspondence algorithm, where the majority of the algorithmic differences arise. In particular, for this stage, rather than using the commonly-employed Mahalanobis distance for comparing predicted and measured values, Lowe develops a probabilistic matching scheme.

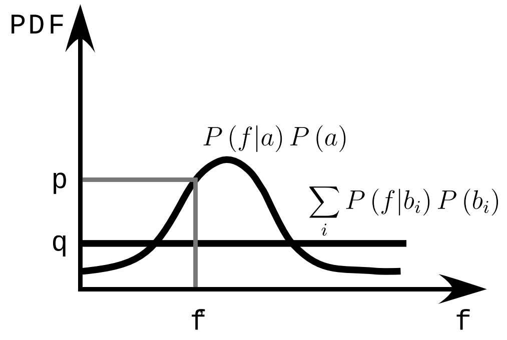


Figure 3.4: Pictorial representation of the probability distribution for a measurement, **f**, given a predicted model feature, **a**. This distribution is compared against a uniform background distribution for all other features, $\mathbf{b}_i$, that provide a false match. Adapted from (Lowe, 1992).

Lowe's algorithm for establishing correspondence between model features and image features is discussed primarily in (Lowe, 1992). Like many of the trackers seen in this section, Lowe also elects to use line features. Specifically, the author denotes **f** as the measurement feature vector obtained when a projected model line, **a**, is associated with a particular image line. All other projected model lines that are not assumed to correspond to **a** are denoted as $\mathbf{b}_i$. Using Bayes rule, an expression for computing the probability of the projected model line, **a**, under this assumed correspondence is derived. This probability is written as

$$\begin{aligned} P(\mathbf{a}|\mathbf{f}) &= \frac{P(\mathbf{f}|\mathbf{a})\,P(\mathbf{a})}{P(\mathbf{f})} \\ &= \frac{P(\mathbf{f}|\mathbf{a})\,P(\mathbf{a})}{P(\mathbf{f}|\mathbf{a})\,P(\mathbf{a}) + \sum_i P(\mathbf{f}|\mathbf{b}_i)\,P(\mathbf{b}_i)}. \end{aligned} \tag{3.18}$$

Figure 3.4 shows a pictorial representation of the probability distribution.

Figure 3.4 assumes that the feature vector, **f**, is a generic, one dimensional value. In this particular work, the feature vector is actually two dimensional. The first value of the feature vector is the perpendicular distance from the center of the image line to the projected model segment. The second component of the feature vector is the angular orientation difference between the two lines. To simplify the mathematics, these two values are assumed to be independent. It is worth noting one flaw with this matching technique. Orientation and perpendicular distance of lines are typically the most robust and consistent measured values of lines. However, these two values alone do not completely constrain the line's position. As Figure 3.5 shows, even if the perpendicular distance and orientation difference between each line is established, no constraints are placed on the amount of "overlap" between the lines. Accordingly,

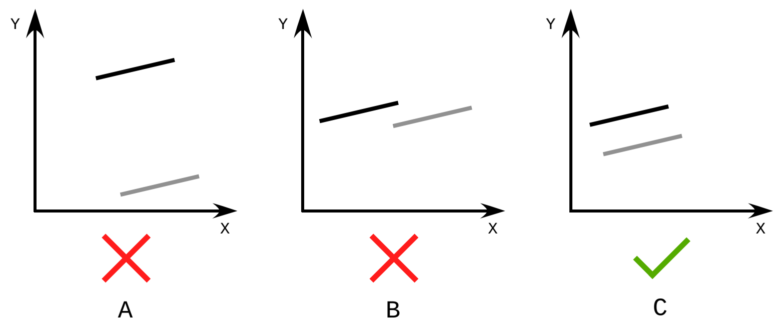


Figure 3.5: Line matching problem in (Lowe, 1991, 1992). A should be rejected because the lines are too far apart. B should be rejected because the lines barely overlap, even though they are spatially close and have a similar orientation. C illustrates good line correspondence. In Lowe's work, both B and C are accepted

even if two lines share a similar orientation and the perpendicular distance, they still may not be a good match.

Once the probabilities of associating each model line with every image line have been computed according to Eq. 3.18, the system selects initial correspondences from which the model parameters are updated. Lowe's work uses a novel "best-first" optimization procedure to attain accurate correspondences. The correspondences between model and image lines are first ordered from the most probable to least probable associations. At first, only the four most probable associations are used. With these four associations, the system calls the pose estimation algorithm that updates the 3D model parameters to bring the matched lines into closer correspondence. Once the optimization has been completed, additional line correspondences are added and pose estimation is repeated. The rationale behind this line correspondence strategy is to let the strongest matches guide the initial stages of optimization. In some sense, this correspondence algorithm is akin to coarse-to-fine strategies. At the outset, the strongest feature matches (often stemming from longer lines) are used to guide initial parameter estimation. Subsequently, the weaker correspondences (often stemming from shorter lines) are used to refine the parameters. The conservative approach provided by this best-first optimization procedure is argued by the author to operate very effectively.

There are several additional properties of Lowe's system that should be made clear at this time. The pose estimation algorithms employed are almost identical to those described for the work of Koller et al. However, it should be noted that Lowe proposed this optimization technique prior to the work of Koller et al. Furthermore, in (Lowe, 1991, 1992), the use of Kalman filters is avoided. The author argues that Kalman filters oversmooth parameters, making them less effective during changes in velocity. In place of a predictive filter, Lowe argues that more effective stabilization and prediction can be obtained by using prior variances on parameters.

The results of Lowe's tracker are quite impressive, overall. Although the results shown in (Lowe, 1991, 1992) all display in-lab tests, the most impressive aspect of the system is its ability to successfully estimate the model parameters that account for articulations. It is challenging to speculate how well this system would fare if it was applied in surveillance situations where lighting changes and objects may be moving more quickly. Since the features being used are discrete, the system should be somewhat resilient to illumination changes, although shadows could present some challenges. The lack of a predictive filter might cause this tracker to struggle if the target is moving quickly. Nonetheless, Lowe's work provides a principled approach to dealing with articulated objects which could presumably be applied to situations such as estimating the lengths of different vehicle sections for the tracker in (Koller, Daniilidis & Nagel, 1993).

Another example of a 3D model-based tracker is provided in (Gennery, 1992). In this work, a slightly different correspondence algorithm is used in conjunction with a Kalman filter prediction mechanism. We will begin the discuss of this system with the correspondence and pose estimation techniques. In more detail, the 3D vertices (junctions between two or more lines in the model) are projected into the image. Special care is taken to ensure that only visible edges have their vertices projected. Following projection, the vertices are used to compute the projected model lines. These lines are represented with their center point and length. Furthermore, metrics are computed that provide the perpendicular distance from the line

$$h = c_x\left(y - y_1\right) - c_y\left(x - x_1\right), \tag{3.19}$$

and the parallel distance along the line

$$g = c_x\left(x - x_1\right) + c_y\left(y - y_1\right). \tag{3.20}$$

In Eq. 3.19 and Eq. 3.20, the line's center point is represented by (c_x, c_y), (x_1, y_1) is one of the line's end points, and (x, y) is an arbitrary point in the image.

The correspondence strategy employed in this work is one point of significant difference from the 3D model-based trackers already described. Instead of comparing projected to image lines, Gennery et al. establish correspondences between the projected lines and the individual image edge pixels. During correspondence, the last five pixels at each end of the projected lines are not used because of their notoriety for being unstable. The remaining pixels along the projected line are sampled at a regular interval. For each of these sampled line pixels, a search is performed to find the closest image edge pixel. The search has a range of five pixels and is performed along the coordinate axis that is closest to being perpendicular to the projected line. Figure 3.6 illustrates this search algorithm. A point along the projected line is con-

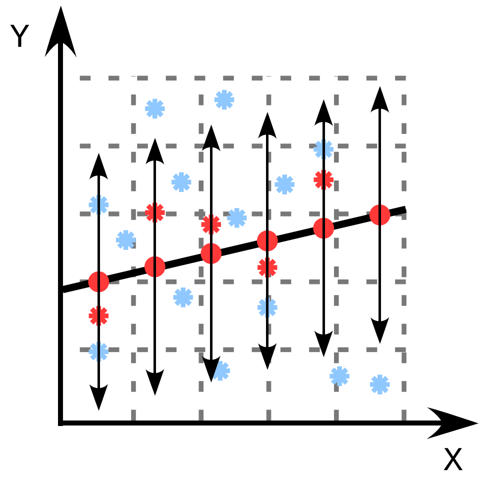


Figure 3.6: Search strategy used in (Gennery, 1992). The black line indicates a projected model line. "*"s indicate detected edges. Red edge pixels are matched to the discrete subsampled points along the projected model line.

cluded to correspond to the closest image edge pixel within the search range, if any edge pixels are found.

Measurements in this work are defined as being the perpendicular distance between the selected edge pixel and the projected line. These correspondences are also assigned a weight, which is set by the authors *a priori*. This weight is used to place more emphasis on projected model lines that should be easier to detect in the image, based on the reflectance and orientation characteristics of the 3D object.

The individual measurements and associated weights are then combined so that the model parameters can be updated. Conceptually, a straight line is fit through the identified image pixels that correspond to a particular projected model line. All obtained straight line fits are then used to update the 3D model parameters. The parameter updates are found using weighted linear least squares. Once the optimal parameter adjustments are found they are used to update a Kalman filter.

In this work, the objects being tracked included a hexagonal prism and a mock satellite. The hexagonal prism was painted a bright white and the scene backgrounds were typically quite dark. Accordingly, contrast edges between the foreground and background are distinct. The mock satellite had a similar shape to the hexagonal prism but it was larger and had more realistic face textures that might be seen on a real satellite. All experiments were performed within the lab, although the authors did note that various illumination conditions were considered. Overall, the results obtained by this tracking solution were qualitatively reasonable, but for arguably easier videos than those seen in (Koller, Daniilidis & Nagel, 1993).

Another 3D model tracker from several years ago was presented by Simon and

Berger (Simon & Berger, 1998). One of the primary differences with this tracker is that curve features in the image domain are tracked instead of points or lines. These curves are actually detected and tracked with a "snake" contour tracker (to be discussed at length in Chapter 4). For this reason, this tracker could be moved to Chapter 5, which will discuss trackers that use a combination of approaches. However, since the overall approach in this work is very similar to other 3D model trackers, we elect to discuss it here. Another unique aspect of this work is that robust metrics are used when matching the model and image features. Aside from these variations, the remaining components of the system in (Simon & Berger, 1998) are similar to the algorithms discussed previously.

The resulting algorithm is certainly unique due to its usage of curve features. It also appears to perform well on a single video sequence where the camera pans across a bridge. However, given the limited amount of data, it is hard to extrapolate its general level of performance. In this algorithm, model to image feature correspondences are assumed to remain constant. This design decision presumably requires the inter-frame motion to be reasonably small.

Another interesting early 3D model-based tracker was presented in (Verghese, Gale & Dyer, 1990). Verghese et al. argue that one could imagine two methods of attacking the 3D model alignment problem. The first approach would be to perform a local perturbation within the 3D model space to generate a number of hypothesized models. These models could then be projected into the image plane and compared with the available data. In (Verghese, Gale & Dyer, 1990), this algorithm is referred to as the "motion-searching paradigm". The second approach would be to perform feature tracking within the image domain. Upon the completion of image domain tracking, the result is back-projected into 3D space to determine the corresponding pose of the model. The authors term this technique the "motion-calculation paradigm". The papers discussed so far in this section all roughly follow the second paradigm.

In (Verghese, Gale & Dyer, 1990), possible implementations for each of these two approaches are described. High spatial and temporal sampling rates are assumed, making inter-frame motions very small. Accordingly, the authors argue that in either paradigm, a predictive mechanism, such as a Kalman filter, is not necessary. Furthermore, with this assumption, correspondences between model features and image features can be assumed to remain consistent from frame-to-frame. This approach was also employed in (Simon & Berger, 1998).

With regards to the motion-searching paradigm, the authors make the assumption that complete movements experienced by a target can be approximated in a piecewise fashion, by varying each model parameter in turn. Hence, rather than exhaustively exploring all possible permutations of the parameters, the authors make use of this approximation to solve a simpler problem. Furthermore, since the temporal sampling rate is very high, the parameters will vary little from frame-to-frame. Thus, the system considers each parameter in turn and generates two hypotheses, corresponding to a small positive and negative perturbation to the parameter. The two 3D models with their perturbed parameters are both projected into the image, one at a time. Normalized cross correlation is performed between the image edge pixels and the projections of the two hypotheses. If the normalized cross correlation score of one hypothesis is significantly better than that of the other, the parameter under analysis is updated to correspond to that of the best match. This process is repeated until all parameters have been considered.

In addition to tracking accuracy, another focus of the paper is implementing the proposed algorithms in a parallel system. One noteworthy point is that under the parallel architecture, the maximum speed of a target that can be tracked was derived and verified experimentally. In terms of tracking accuracy, limited results on fairly simple video sequences are presented, making it hard to compare this system against more recent works. However, given that this system was proposed in 1990, it is one of the first works in 3D model-based tracking, making it quite noteworthy.

With respect to the motion-calculation approach presented in (Verghese, Gale & Dyer, 1990), a very simple approach is taken. Given that a high temporal sampling rate is used, correspondences between model and image features are assumed to remain unchanged from frame-to-frame. Thus, the problem collapses to tracking the image features (linked edges are used) from frame-to-frame and then performing pose estimation using techniques similar to those seen in Lowe's work. The results illustrated for this approach in (Verghese, Gale & Dyer, 1990) are once again limited, but impressive, considering the year of publication

Most recently, systems that employ the abovementioned general three stages of processing to perform 3D model-based tracker are becoming more rare. Despite its decreased level of exposure in the research community, feature-based 3D model tracking does still receive some limited attention. One example of a recent work in this subfield was presented in (Vacchetti, Lepetit & Fua, 2004). The biggest contribution of this work is its attempt to derive a correspondence algorithm that includes multiple hypotheses.

Typically when matching between the image edges and projected model lines, a single match is established that provides the minimum cost/distance. The authors of (Vacchetti, Lepetit & Fua, 2004) argue that this approach is not always suitable as high-contrast edges may generate superior match scores based on the strength of the edge rather than its goodness of match. Clearly, this argument depends on the particular type of match metric that is being utilized. The proposed algorithm alleviates the problems associated with incorrect correspondences by making use of multiple measurement hypothesis when computing matches. More specifically, the matching strategy in (Vacchetti, Lepetit & Fua, 2004) begins in the standard manner of projecting a model line into the image domain. Points along the line are sampled and from these seed samples local searches are performed for corresponding edges. The searches are performed in the direction normal to the line.

Upon performing this search, an obvious strategy would be to associate each sampled point along the line with the strongest edge pixel. Under such an approach, the 3D model parameters could be updated to minimize the following overall cost

$$v_t = \frac{1}{N_e} \sum_i \sum_j \rho \left(\left\| e_{i,j} - e'_{i,j} \right\|^2 \right) \tag{3.21}$$

where $e_{i,j}$ is a sampled point along the projected line, $e'_{i,j}$ is the strongest edge discontinuity uncovered during the search operation, N_e is the total number of sample points, ρ is a robust estimator, i varies over all projected model lines, and j varies across all sampled points on a specific line.

In this work, the authors propose the following amendment to Eq. 3.21

$$\begin{aligned} v_t^* = \frac{1}{N_e} \sum_i \sum_j \rho^* \Bigg(\left\| e_{i,j} - e'_{i,j,1} \right\|^2 , \left\| e_{i,j} - e'_{i,j,2} \right\|^2 , \ldots \\ \left\| e_{i,j} - e'_{i,j,K_{i,j}} \right\|^2 \Bigg) , \end{aligned} \tag{3.22}$$

where $K_{i,j}$ is the number of image edge pixels hypothesized to correspond to a particular $e_{i,j}$. Thus, Eq. 3.22 considers numerous correspondences rather than just the single strongest edge. For Eq. 3.22, a modified robust metric has to be introduced, since robust metrics do not natively handle vectors. In this work, the authors define $\rho^* (x_1, \ldots x_n) = \min_i \rho (x_i)$. Intuitively, this function considers multiple matching hypothesis, but it is only the best match that contributes to the overall cost function of Eq. 3.22. The authors report that this modified objective function provides accurate tracking in situations where using Eq. 3.21 fails.

There are several additional systems that we would like to briefly make note of before concluding this section. Specifically, another fairly recent work that is worth mentioning was proposed in (Drummond & Cipolla, 2002). In this work, a fairly standard 3D model-based tracker was proposed whereby the SSD between sample points along the model lines and the nearest image edges (found along normal directions in a similar manner to (Gennery, 1992)) is minimized. However, in this work, an active camera component is also incorporated. Once the pose of the 3D model with respect to the image is determined, it is used to compute the necessary motion of the camera to obtain a specified view of the target object. This relocating of the camera to seek out a particular view of a target is successfully demonstrated for a stationary target and when a target is moving (slowly) in discrete steps. Additionally, in (Tsin, Genc, Zhu & Ramesh, 2007) when correspondences are established between projected model edges and image domain edges, intensity patches surrounding the edges are used to improve robustness. The proposed 3D model tracker also incorporates an on-line learning scheme to adapt to appearance changes of the abovementioned intensity patches. Finally, in (Brand & Bhotika, 2001), an extremely unique 3D model-based tracker is demonstrated for the problem of tracking deformable, non-rigid faces. Unlike the other 3D model-based trackers described in this review, the system by Brand and Bhotika directly makes use of intensity gradient features when tracking the 3D target. A second novel aspect of the work is that the system can be provided with only a crude model of the target. In addition to tracking, the system refines the 3D model to more accurately represent the specific target observed in the particular video sequence.

It is not a secret that since the early 1990s feature-based 3D model trackers have received less attention from the computer vision tracking community. Perhaps this decreased level of attention is due to the fact that the problem of aligning three dimensional models with image data is a very hard, especially as the models get more complex. There may be some truth in this explanation, but perhaps an alternate, more satisfactory argument is available. In particular, with these systems, the designer is locked in to tracking a particular class of object. Certainly, there are some applications where the types of objects to be tracked are known beforehand, but there are a wealth of situations where the types of objects to be tracked are unknown and continuously varying. Outside of designing a separate 3D model-based tracker for every type of object that will be viewed in a scene, there is no obvious method of making a "universal 3D model tracker". Another problem with these trackers is that obtaining 3D object models is not trivial and can often necessitate the need for specialized equipment.

Relations to 3D Model Detection

It is interesting to note the intimate relationship shared between 3D model detection and 3D model-based tracking. In particular, the central idea of tracking 3D models in images appears to have emerged from the earlier literature concerned with the detection of 3D models in 2D images (e.g., (Thompson & Mundy, 1987; Lowe, 1991)). In fact, the work by Lowe (Lowe, 1991), which we discussed previously, was primarily concerned with the problem of 3D model detection rather than tracking.

In addition to (Lowe, 1991) another example of an early system that performed 3D model-based detection is (Thompson & Mundy, 1987). In this work, pairs of vertices (a vertex is defined as the intersection between two lines) are defined as the features used for comparing model to image data. A unique aspect of the overall system is that correspondences are not established between image domain and model domain features. Instead, correspondences are exhaustively considered using a "binning" or "voting" strategy that is somewhat akin to the Hough transform in a 6D parameter space.

Another interesting point to note regarding the relationship between 3D model detection and tracking is that the field seems to have come full circle. Three dimensional model-based tracking stemmed from 3D model detection. Subsequently, many of the ideas used in 3D model detection were employed in tracking, with the additional constraints and techniques concerned with establishing temporal consistency between frames (e.g., predictive filters, assumed constant correspondences, etc.). Now that detection schemes are becoming more computationally efficient, some researchers are proposing to perform 3D model tracking using detection techniques on every frame and putting less emphasis on temporal consistency constraints (e.g., (Leng & Wang, 2004; Lepetit, Lagger & Fua, 2005))! We will briefly discuss two of these proposed works.

In (Leng & Wang, 2004) an offline database of the target's appearance is manually constructed. This database is typically employed when detecting the target in the current frame. However, in the event that the target's appearance changes (e.g., due to illumination changes or occlusion), the system can switch to frame-to-frame matching instead. Another more recent approach to tracking via 3D model detection eliminates the costly and time-consuming requirement of constructing a database of targets views (Lepetit, Lagger & Fua, 2005). Instead, interests point features are detected on the target and affine warps are used to artificially construct other views. However, one should not be fooled into believing that this approach of artificially constructing a database from a single object view will always be sufficient. Specifically, the generated views can only capture the fundamental target information that was present within the original image. Target features that were not visible in the original view will not be represented in such a database.

There are several benefits to be had by performing continuous detection instead of inter-frame tracking. In particular, when inter-frame tracking is performed, manual initialization is typically needed for the first frame. Furthermore, with tracking-based approaches, errors can accumulate, causing the tracker to drift off the center of the target and eventually loose the target completely. Finally, tracking-based approaches typically cannot recover from complete failures. Thus, if a sudden movement, accumulated error, or some other phenomena causes the tracker to fall off the target completely, recovery can be challenging. However, continuous detection-based approaches do have their own shortcomings. In particular, more pre-processing is required to construct the database of the target under various poses. Furthermore, poses of the target in the sequence can only be recovered if there is a close match in the database. As a result, care must be taken to construct a very complete database consisting of all expected target poses. Finally, given that the database is typically constructed under ideal conditions, tracking during illumination changes and occlusions can be problematic.

Matching 3D Models to 3D Data

Although this review is primarily concerned with visual tracking in monocular imagery obtained using CCD cameras, we will take a brief moment to discuss tracking in different types of imagery. In particular, the matching of three dimensional range data that can be obtained using stereopsis or light detection and radar (LiDAR) to 3D models. The rationale for mentioning these systems in this review is that these systems are an important recent offshoot to the standard 2D image features to 3D model trackers already discussed in this chapter.

A very popular technique in this field is known as the iterative closest point (ICP) algorithm (Besl & McKay, 1992). The idea here is that if correspondences between the 3D model points and 3D data points are known, then the rotational and translational parameters can be obtained by minimizing the sum of squared Euclidean distances. The main problem, therefore, lies in establishing the point correspondences. The ICP algorithm operates by iteratively making intelligent guesses regarding the point correspondences. Specifically, for each data point there will be a single model point to which it is closest, in the Euclidean distance sense. These points are assumed to correspond. In fact, this process is repeated so that correspondences are established between each model and data point based on the smallest Euclidean distances. Once preliminary correspondences have been established, the system estimates the transformation that brings the point sets into alignment. The process of recomputing the correspondences and realigning the point sets is iterated until convergence is reached.

On the positive side, the ICP algorithm monotonically approaches the nearest local minimum solution. However, a significant problem with the original ICP algorithm proposed in (Besl & McKay, 1992) is that its execution time is often several minutes in duration. Accordingly, researchers have attempted to make the algorithm more efficient. In (Simon, Hebert & Kanade, 1994), the algorithm's speed is increased via tree-based pruning algorithms and closest point caching for future frames. The authors report that after the proposed speed enhancements, the resulting algorithm only requires 1% of the processing time of the original ICP algorithm.

Another very recent and interesting work in the field of 3D data to 3D model matching was presented by Shang et al. (Shang, Jasiobedzki & Greenspan, 2007). The main idea in this work is to place assumptions on the inter-frame target transformations to restrict the number of feasible transformations. If sufficient assumptions are used, the transformations can be exhaustively considered. To reach this goal, three main assumptions were employed. First, it is assumed that the transformation is rigid. Secondly, it is assumed that the target can move with some maximum speed. Finally, it is assumed that the images are taken at a known, fast sampling rate. With these assumptions, maximum bounds can be placed on the inter-frame transformation, B.

In addition to limiting the search range, in (Shang, Jasiobedzki & Greenspan, 2007), the changes that can be incurred by each translational and rotational parameter are discretized. This discretization further reduces (in fact, from an infinite number to a small, finite number) the possible number of allowable transformations. After discretization, the number of transformations is low enough that they can be efficiently exhaustively considered. Another advantage of discretizing the transition space is that the problem of determining the optimal transformation can be transformed into a classification problem.

Several classification strategies are proposed in (Shang, Jasiobedzki & Greenspan, 2007), but they are all based upon a binning/voting strategy similar to the Hough transform or the method seen in (Thompson & Mundy, 1987). In (Shang, Jasiobedzki & Greenspan, 2007), the measured data is three dimensional range data. Thus, the first step of the most straightforward classification approach is to transform the measurements and models into the same "voxel" feature space. In fact, for *all possible* discrete transformations, B, the voxel feature representation is created for the transformed versions of the model. For the measured data, only a single voxel feature representation is generated. The system then identifies the particular transformation in which the transformed model and measurement voxels are in closest alignment using a voting procedure.

In addition to being fairly unique, it appears that the approach also performs well, in practice. Specifically, the system is shown to be capable of tracking simple block objects, a more complex dinosaur, and even a model of a molecule. However, in addition to the general disadvantages of 3D model based approaches, this system also assumes the object is rigid.

3.3 Recapitulation

In summary, in this chapter we have considered trackers that make use of discrete features. We began with a discussion of trackers that use simple point features and progressively considered more complex features, including lines, groups of edges, and even 3D models. Point feature trackers often rely on positional and velocity-based assumptions to match targets from frame-to-frame. Line-based trackers typically employ Kalman filters to predict and correct the various parameters that define a line. Finally, 3D model trackers face the challenges of relating model-based features to image-based features and subsequently bringing those two sets of features into closer alignment. In the next chapter, we will shift our focus to trackers that represent targets as curves during execution.

Chapter 4

Tracking With Contours

To begin the discussion for this chapter, let us first define what is meant by a contour tracker. For this review, a contour tracker is defined as any system that follows a target from frame-to-frame and represents said target with a curve that adheres to its outline. The curve representing the object can be closed or open, but typically closed curves will be described in this review. One might argue that in some ways, contour trackers (specifically, contour trackers using an open curve) are similar to the line trackers described in the previous chapter. In particular, both approaches can track the boundary lines of targets within the scene; however the line trackers discussed in Chapter 3 are limited to following straight line segments. Due to the fact that contour trackers can adhere to the boundaries of targets that are drastically different from a straight line (e.g., a circle) and the techniques used for these two types of trackers are quite different, we have elected to separate them into two disparate classes.

Unlike many trackers that are described in this review, contour trackers ideally provide very precise localization of the target and its boundary. Most contour trackers allow this boundary to be of any arbitrary shape and to be altered in a non-rigid manner. Thus, in each frame, the goal of this class of trackers is to determine the optimal shape and placement of the tracking contour in the given image frame. Most of these trackers assume that the framerate is sufficiently high, leading to small inter-frame changes of the target's shape and position. Accordingly, the task of locating the optimal location and shape of the target is reduced to that of modifying the optimal contour found at the previous time instant.

This chapter will be divided into four sections. The first section will focus on the earliest works in the area of contour-based tracking. Subsequently, a more recent formulation of contour trackers, one that employs the usage of level sets, will be detailed in conjunction with geodesic contour trackers. The chapter will continue with a discussion of trackers that represent the target as a contour, but use more than just gradient-derived information for promoting its evolution. Finally, the chapter will close with a discussion of several contour trackers that do not fall into any of the other categories.

4.1 Basic Snakes

The first examples of contour-based trackers started appearing in the late 1980s and early 1990s (Kass, Witkin & Terzopoulos, 1988; Terzopoulos & Szeliski, 1992; Leymarie & Levine, 1993). To our knowledge, the seminal work in this field was completed by Kass, Witkin, and Terzopoulos (Kass, Witkin & Terzopoulos, 1988), which was subsequently elaborated upon in (Terzopoulos & Szeliski, 1992). The authors termed this class of contour trackers as "snakes", presumably because the curve slithers around the image (not unlike a snake) as it attempts to find the target. It is with these snake systems that our discussion of contour-based trackers will begin.

In the earliest works (Kass, Witkin & Terzopoulos, 1988; Terzopoulos & Szeliski, 1992), the idea of contour evolution for tracking was posed as an energy minimization problem. Specifically, the energy represents the goodness of a contour with respect to the current frame and the goal is to identify the contour that minimizes this energy. To create a snake-based tracker, two components must be designed: the energy function and the method of optimizing the energy.

The intuitive rules that the authors of (Kass, Witkin & Terzopoulos, 1988; Terzopoulos & Szeliski, 1992) used to define the energy functions subsequently became standard in the snake literature. The first fundamental idea behind the energy of a snake is with regards to its shape. In general, it is assumed that targets of interest do not contain protruding edges and sharp, deep valleys. Rather, the designers of snake-based trackers believe that most objects are composed primarily of smooth edges. Furthermore, it is assumed that the length of an object's outline will not change significantly from frame-to-frame. These intuitive ideas are expressed more formally using

$$E_{int} = \int_0^1 w_1(s)\left|v_s\right|^2 + w_2(s)\left|v_{ss}\right|^2 ds, \tag{4.1}$$

where v is the contour and w_1, w_2 are weights. The contour, v, is actually represented as a mapping from the parametric domain, $s \in [0, 1]$, to the image plane. The variables v_s and v_{ss} represent the first and second derivatives of the contour with respect to the parameter, s. At a higher level, the first term in the equation controls the "tension" of the snake. In other words, this term makes the snake behave like a string, in that it attempts to maintaining the same overall length and resist stretching. The second term in Eq. 4.1 controls the "rigidity" of the snake. One can imagine this term as making the snake act more like a rod that resists bending. Together, these two terms indicate that snakes with relatively smooth shapes that remain roughly the same length are preferred.

At this point, the described energy is only dependent upon the shape of the curve itself — there has been no mention of reliance on the image data. Of course, image data must enter into the energy equation and it does so in a so-called external energy term, $E_{image}(v)$. The external energy of the snake is an integral of the form

$$E_{image}(v) = \int_0^1 P(v(s))\, ds, \tag{4.2}$$

where the variable of integration varies from zero to one as this represents a complete cycle around the target contour and P is the external image potential energy. A variety of functions can be used for computing the image potential energy and it is up to the designer to select an appropriate function that meets the needs of the desired application. The energy, P is typically based on image intensity or image gradient measurements. For instance, the use of the function $P(x,y) = \pm c\,[G_\sigma * I(x,y)]$ would result in a snake that is attracted to bright or dark lines within in the image. Note that $G_\sigma * I(x,y)$ simply indicates that the image is filtered by a Gaussian smoothing filter with standard deviation, σ, to ameliorate the effects of high frequency noise artifacts and that c is a constant. An alternative image potential energy function that is attracted to high contrast, intensity edges can be written as $P(x,y) = -c\,|\nabla [G_\sigma * I(x,y)]|$.

The most fundamental snake energy (Kass, Witkin & Terzopoulos, 1988) is merely the summation of the internal and external energy terms. Explicitly, the energy of a snake is

$$E_{snake}(v) = E_{int}(v) + E_{image}(v). \tag{4.3}$$

Naturally, this energy definition can be modified to meet the needs of the tracking system. However, in the original works on snakes (Kass, Witkin & Terzopoulos, 1988; Terzopoulos & Szeliski, 1992) and many subsequent works (Leymarie & Levine, 1993; Peterfreund, 1997, 1999), this was exactly the energy equation that was employed.

As alluded to earlier, in addition to defining a quantitative means of computing the goodness of a snake, the second major requirement when creating a snake-based tracker is developing an optimization scheme. If one imagines the energy minimization task as a static problem of finding the minimum contour, v, that satisfies Eq. 4.3, then a variety of standard optimization techniques could be used. For instance, Eq. 4.3 could be minimized using greedy methods, dynamic programming, gradient-based methods, etc. However, very shortly after the initial inception of snake trackers, it was realized that there were gains to be had by viewing snake energy minimization under a different light. Specifically, Terzopoulos and Szeliski argue in favor of treating a snake as a dynamic system, whose energy can be minimized using Lagrangian mechanics (Terzopoulos & Szeliski, 1992).

To imagine energy minimization from the perspective of a dynamic system, a

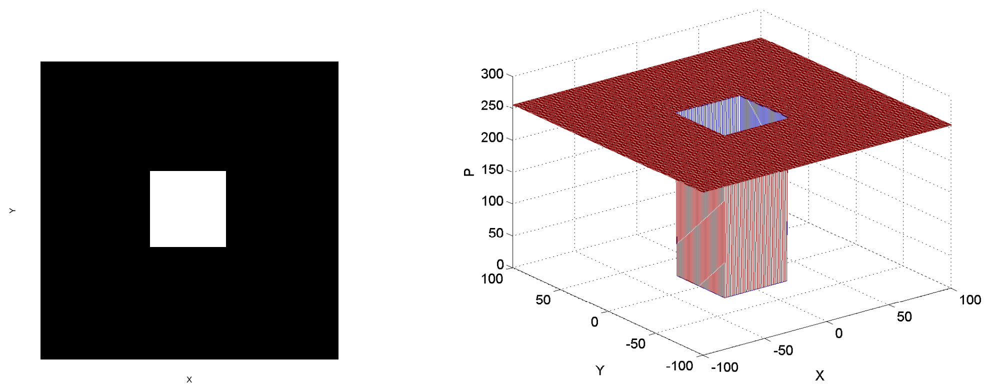


Figure 4.1: Synthetic image of a square target (left) with its corresponding potential energy surface (right).

reasonable visualization tool is that of a three dimensional potential energy surface. An example of such a image potential energy surface is shown in Figure 4.1. As can been seen, the image potential energy $P(v(s))$ is shown as a three dimensional surface (rather than as a 2D intensity image). The goal during energy minimization is to have the snake reach a state of low gravitational potential energy. This goal is achieved by moving the snake to the valleys within the potential energy surface. As the snake travels from high potential energy to low potential energy regions, potential energy is converted to kinetic energy. Of course, the conversion of potential energy to kinetic energy is never a lossless process. If the conversion were lossless, pendulums would swing indefinitely and a bouncing basketball would continually bounce back to the height at which it was dropped. During the conversion, a percentage of energy is dissipated, often in the form of heat or other "wasted" energy. Accordingly, when modeling snake energy minimization as a dynamic process, a damping factor is needed to ensure that the snake reaches a state of equilibrium.

The above description of Lagrangian dynamics provides an indication of the additional components that are needed to transform the static snake energy minimization task into this new framework. Firstly, the contour itself now becomes a function of time, $v(s,t)$. Furthermore, the system potential energy is defined as the overall energy of the snake, E_{snake}. However, we now also require a measure of kinetic energy. In (Terzopoulos & Szeliski, 1992), the kinetic energy of a snake is computed using $\int_0^1 \mu |v_t|^2 ds$, where v_t is the time derivative of the contour and $\mu(s)$ is the mass of the snake at point s. This equation is exactly analogous to the kinetic equations studied in simple physics ($\frac{1}{2}mv_t^2$, where m is mass and v_t is velocity).

Following the notation in (Terzopoulos & Szeliski, 1992), a Lagrangian for the snake can be constructed by combining the kinetic and potential energy terms

$$\begin{aligned} L(v) &= \frac{1}{2}\int_0^1 \mu |v_t|^2 - \frac{1}{2}E_{snake} \\ &= \frac{1}{2}\int_0^1 \mu |v_t|^2 - w_1(s)|v_s|^2 - w_2(s)|v_{ss}|^2 - P(v)\,ds. \end{aligned} \tag{4.4}$$

Generally speaking, the Euler-Lagrange equation can be used to optimize a functional of the form

$$J = \int_a^b F(t, f(t), f'(t))\,dt, \tag{4.5}$$

where $f'(t)$ is the first derivative of the function, f. The Euler-Lagrange equation that is used to optimize Eq. 4.5 is a differential equation of the form

$$F_f(t, f(t), f'(t)) - \frac{d}{dt}F_{f'}(t, f(t), f'(t)) = 0, \tag{4.6}$$

where F_f is the derivative of F with respect to f and $F_{f'}$ is the derivative of F with respect to f'.

The snake Lagrangian defined in Eq. 4.4 satisfies the general form required for optimization via the Euler-Lagrange equation. Accordingly, the partial differential equation for maximizing the snake energy can be written as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial v_t}\right) + \frac{\partial D}{\partial v_t} - \frac{\partial L}{\partial v} + \frac{\partial}{\partial s}\left(\frac{\partial L}{\partial v_s}\right) - \frac{\partial}{\partial s^2}\left(\frac{\partial L}{\partial v_{ss}}\right) = 0. \tag{4.7}$$

Note that the second term in the partial differential equation arises due to a dissipation term, D, that must be included to ensure that the snake converges to static equilibrium, as intuitively alluded to previously. Specifically, Terzopolous and Szeliski suggest using a Rayleigh dissipation functional, $D(v_t) = \int_0^1 \gamma |v_t|^2\,ds$, where $\gamma(s)$ is the damping density at point s.

If we assume that the the snake mass and damping densities are constant, with some manipulations, the force balancing equations for the snake are

$$\mu v_{tt} + \gamma v_t - \frac{\partial}{\partial s}(w_1 v_s) + \frac{\partial^2}{\partial s^2}(w_2 v_{ss}) = -\nabla P(v(s,t)). \tag{4.8}$$

Qualitatively, the terms on the left hand side of Eq. 4.8 correspond to kinematic, dissipation, stretching, and bending forces, respectively. These forces are balanced by the gradient of the image potential energy on the right hand side. Using Eq. 4.8, a contour can be identified that minimizes the overall snake energy.

Before Eq. 4.8 can be employed in a computer vision tracker, it must first be discretized. Thus, discrete approximations of the above equations must be derived. The works of Terzopolous et al. (Terzopoulos & Szeliski, 1992) and Leymarie et al. (Leymarie & Levine, 1993) both provide thorough discussions of the discretization of the snake energy equations. Here, the most important points will be highlighted. To begin, a continuous snake contour must replaced by a series of vertex points, u_i, along the continuous curve. The energy of this discretized curve is computed as

$$E\left(u\right) = \frac{1}{2} u^T \mathbf{K} u + P\left(u\right), \tag{4.9}$$

where $\mathbf{K}$ is the stiffness matrix that contains information concerning the weights w_1 and w_2. Additionally, $P\left(u\right)$ is a discrete version of the image potential energy.

Following the ideas presented in (Terzopoulos & Szeliski, 1992), a discrete version of the Lagrangian motion equation of Eq. 4.8 can be written as

$$\mathbf{M} u_{tt} + \mathbf{C} u_t + \mathbf{K} u = -\nabla P\left(u\right), \tag{4.10}$$

where $\mathbf{M}$ is a matrix containing mass information and $\mathbf{C}$ is the damping matrix. Terzopolous and Szeliski suggest computing the temporal derivatives in Eq. 4.10 using central difference approximations. The following derivation illustrates how one can compute an expression for updating the contour at time $t + \Delta t$ by inserting central difference approximations into Eq. 4.10

$$\begin{aligned} \mathbf{M} u_{tt} + \mathbf{C} u_t + \mathbf{K} u &= -\nabla P\left(u\right) \\ \mathbf{M} \frac{u^{(t+\Delta t)} - 2u^t + u^{(t-\Delta t)}}{\Delta t^2} + \mathbf{C} \frac{u^{(t+\Delta t)} - u^{(t-\Delta t)}}{2\Delta t} + \mathbf{K} u^{(t+\Delta t)} &= f \\ \left[\frac{\mathbf{M}}{\Delta t^2} + \frac{\mathbf{C}}{2\Delta t} + \mathbf{K}\right] u^{(t+\Delta t)} = \frac{2\mathbf{M} u^t - \mathbf{M} u^{(t-\Delta t)}}{\Delta t^2} + \frac{\mathbf{C} u^{(t-\Delta t)}}{2\Delta t} &+ f \\ \mathbf{A} u^{(t+\Delta t)} &= b^t. \end{aligned} \tag{4.11}$$

With Eq. 4.11, we have arrived at our required update equation for the contour. To summarize, we began with the continuous Lagrangian motion equation (Eq. 4.8) which was subsequently discretized (Eq. 4.10). This final step shows how to derive a contour update equation from the discrete version of the partial differential equation.

Up to this point, we have discussed the most basic snakes that were originally presented in the literature (Kass, Witkin & Terzopoulos, 1988; Terzopoulos & Szeliski, 1992; Leymarie & Levine, 1993). Following the introduction of these standard snakes, several extensions were made. For instance, in (Terzopoulos & Szeliski, 1992), the authors continue the discussion of snakes by describing how: (1) snakes can be derived in a probabilistic framework (2) Kalman filters can be used in conjunction with snakes. These additional contributions are actually very closely intertwined as the probabilistic derivation of snakes can be conveniently linked with the Bayesian perspective of Kalman filters. Introducing Kalman filters into the snake tracking paradigm can provide superior initial guesses for the solution in the current frame. As a result, larger inter-frame target displacements can be estimated and local minima pose less of a problem.

As mentioned previously, another independent work on snakes was presented by Leymarie and Levine (Leymarie & Levine, 1993). This work was published in the same year as (Terzopoulos & Szeliski, 1992) and focused primarily on the implementation aspects of snakes. Most of these implementation details are very similar to those presented in (Terzopoulos & Szeliski, 1992). Nonetheless, three interesting contributions of this work are as follows. (1) The authors apply snakes to cell tracking, an application area that had received only limited attention with snakes up until that time. (2) The authors perform snake optimization in a coarse-to-fine framework. In a similar manner to other coarse-to-fine schemes, snake optimization for a particular frame is first completed at the coarsest scale of the pyramid. The end result obtained at the coarser levels provides initial estimates for the next finest scale. In their implementation, optimization is performed up to and including level one of the pyramid. Optimization is not performed at level zero because the authors argue the highest frequency level is corrupted with too much noise. (3) The authors propose a novel new stopping criterion for optimization. This stopping criterion will be elaborated upon next.

In (Leymarie & Levine, 1993), it is argued that when optimization iterations are performed until the standard snake energy (Eq. 4.3) is minimized, the snake vertices will tend to collapse to small clusters of points that lie within the lowest valleys of the potential energy surface. To circumvent this problem, one could imagine altering the snake energy equation by incorporating additional terms or adjusting weights that penalize the snakes from collapsing in this fashion. However, the authors of (Leymarie & Levine, 1993) propose a different solution. Specifically, a new stopping criterion is suggested.

In their work, Leymarie and Levine describe a stopping condition that is based on the average energy of the contour. As soon as the contour's average image potential energy stabilizes, the authors argue that further optimization iterations are uncalled for, and in fact, may actually cause the snake to skip over a preferred solution. Specifically, a solution where all snake vertices are at approximately an equal height on the potential energy surface, is much preferred over a snake where groups of vertices are clustered in the various deepest valleys of the surface. Thus, the length-based stopping condition proposed in (Leymarie & Levine, 1993) is

$$E_{length}\left(v\right) = \frac{\int_0^1 P\left(v\left(s\right)\right)}{\int_0^1 |v_s| ds}, \tag{4.12}$$

where $P(v)$ is the image potential energy described previously in Eq. 4.2. The numerator is the sum of the heights of the snake vertices on the potential energy surface while the denominator is the length of the snake. By using the average potential energy of all the vertices as the stopping condition, the authors argue that the bias toward shrinking snakes is reduced. It should be emphasized that during optimization itself, the complete energy (internal energy plus potential image energy) as described in Eq. 4.3 is used. The average potential energy, Eq. 4.12 is used exclusively as a stopping condition.

After these initial exploratory works (Kass, Witkin & Terzopoulos, 1988; Terzopoulos & Szeliski, 1992; Leymarie & Levine, 1993) laid out the theoretical and implementation foundations of snakes, subsequent papers proposed extensions to this class of contour tracker. An example of snake extensions is provided by the work of Peterfreund (Peterfreund, 1997, 1999). In this work, the author includes velocity information in the snake energy. The general rationale is that the velocity of the snake should mimic the velocity of the contour in the image. For example, even if there is a very strong intensity edge in the image, we probably would not want our snake to be attracted to this edge if it was traveling in the complete opposite direction to the snake's previous velocity. Peterfreund incorporates this intuitive idea into the snake formulation and terms the resulting mechanism a "velocity snake".

In his formulation of velocity snakes, Peterfreund recommends using the internal and image potential energies exactly as they were proposed in prior works. The addition of velocity information arises in the damping function. Rather than using the damping function suggested earlier ($D(v_t) = \int_0^1 \gamma |v_t|^2 ds$), the author incorporates velocity information using the following equation

$$D\left(v_t, v_t^b\right) = \frac{\gamma}{2} \int_0^1 \left|\left|L^T\left(v_t - v_t^b\right)\right|\right|^2 ds + \frac{\beta}{2} \int_0^1 \left|\left|\frac{\partial}{\partial s} v_t\right|\right|^2 ds, \tag{4.13}$$

where v_t is the snake contour velocity and v_t^b is the velocity of the image boundary. The first term on the right hand side of the dissipation function measures the difference between the snake velocity and the velocity of the boundary within the scene. The second term is a smoothness constraint added by the author for the purpose of adding additional numerical stability.

Altering the dissipation function clearly will have an impact on the Euler-Lagrange equation. In particular, the new motion equations are

$$\begin{aligned} \mu v_{tt} + \gamma C\left(v_t, v_t^b\right) - \beta \frac{\partial}{\partial s}\left(\frac{\partial}{\partial s} v_t\right) - \frac{\partial}{\partial s}\left(w_1 v_s\right) + \frac{\partial^2}{\partial s^2}\left(w_2 v_{ss}\right) & \\ = -\nabla P\left(v\left(s,t\right)\right), & \qquad (4.14) \end{aligned}$$

where $C\left(v_t, v_t^b\right) = LL^T\left(v_t - v_t^b\right)$, and is termed the "velocity control". Notice that it is only the second and third terms of Eq. 4.14 that differ from the original Euler-Lagrange equation for snakes (Eq. 4.8).

Previously, we have discussed how one can compute estimates of the contour velocity, v_t by using central difference approximations. The problem that remains to be solved in (Peterfreund, 1997, 1999) is that of computing the image boundary velocity, v_b. One method that the author proposes is to merely compute image optical flow at the snake vertex points. These velocity estimates can be derived using Horn and Schunk's method of computing optical flow by minimizing

$$\int_R \frac{1}{2}\left(\nabla I\mathbf{w} + I_t\right)^2 + \frac{\beta^2}{2}\left(||\nabla p||^2 + ||\nabla q||^2\right)dx, \tag{4.15}$$

where the integral considers all pixels within image region, R, β is a weighting term, and $\mathbf{w} = (p, q)$ is the velocity vector with horizontal, p, and vertical q components.

Instead of explicitly computing the optical flow at the image snake points, Peterfreund does propose an alternative (Peterfreund, 1997, 1999). When the variable L in the velocity control is assigned a particular value, the necessity of explicitly computing optical flow is removed. Specifically, assuming $L = \nabla I\left(v\left(s,t\right)\right)$, the velocity control term reduces to

$$\begin{aligned} C\left(v_t, v_t^b\right) &= \nabla I\left(\nabla v_t - \nabla I v_t^i\right) \\ &= \nabla I\left(\nabla v_t + I_t\right), \end{aligned} \tag{4.16}$$

by using Horn and Schunk's gradient constraint equation. With this velocity control term, Peterfreund's modified Euler-Lagrangian equation can be solved without explicitly computing optical flow. However, Peterfreund does note that this solution is more sensitive to noise and therefore superior results are typically attained by explicitly computing flow. Finally, it should be noted that the primary contribution of (Peterfreund, 1999) is that it extends velocity snakes to include a Kalman filtering mechanism.

The overall results of (Peterfreund, 1997, 1999) are arguably the most advanced of all the snake-based trackers considered in this review. The earlier works (Kass, Witkin & Terzopoulos, 1988), considered fairly simple videos and often incorporated user interactive mechanisms. The latter work by Terzopoulos et al. (Terzopoulos & Szeliski, 1992) increased the difficulty level by tracking cells, non-rigid facial features, and objects in a laboratory environment. Nonetheless, most of the sequences dealt with fairly clear, high-contrast edges in minimal clutter. Leymarie and Levine's work (Leymarie & Levine, 1993) appeared to provide adequate qualitative performance, but only in the limited domain of cell target tracking. Peterfreund's work extends beyond the prior performance demonstrations and illustrates results across a wide range of environments, many of which involve difficult tracking situations. Specifically, both indoor and outdoor video sequences are shown where rigid and non-rigid objects are tracked. Although the easiest sequence Peterfreund presents involves a white car on gray asphalt, the degree of difficult in his additional experiments is quite high. For instance, one sequence illustrates the tracking of a human hand in an office environment where there is significant levels of clutter in the background. The improved version of Peterfreund's tracker that includes the Kalman filter does indeed offer superior performance to the one that omits this component. In general, the Kalman filter-enhanced tracker provides tighter and more precisely centered object contours. More importantly, however, the velocity snake coupled with the Kalman filter has the ability to track through occlusion events. Furthermore, when Kalman filtering is used, the tracker also provides a confidence measure for each of its vertices. Partial or full occlusion can be detected by analyzing the vertices' confidences.

In summary, this section has discussed some of the earliest contour trackers, known as snakes, and their subsequent extensions. Since these initial contour trackers, other contour-based systems have been derived which rely on different mathematical concepts such as level sets. These more recent contour trackers will be discussed in the next section.

4.2 Level Set and Geodesic Methods

The snake trackers discussed in the previous section provided a starting point for tracking objects via curves. In any research area, seldom is it the case that the first solution to a problem is the optimal one. Initial techniques that are proposed take time to be refined and improved upon. In fact, many research solutions are perpetually works in progress. It is no different with respect to contour-based tracking. Not long after snake trackers started becoming more popular, did researchers begin to take note of the limitations of the snake framework. The problem of snakes, as many researchers suggest, is that the parameterization of a curve is too cumbersome for most tracking problems. Fortunately, a solution to this challenge has been proposed, namely, representing and propagating a curve with level sets. As level sets will be the topic of great discussion in this and subsequent subsections, we shall begin with a description of level sets themselves.

4.2.1 Level Set Theory

Level sets can provide a representation of a curve or interface and can be used to track their movement. In essence, level sets provide an alternative to representing a curve as a parameterized function. Avoiding contour parameterization by using level sets offers numerous advantages, including the ability to easily track curves with corners and cusps as well as allowing topological changes within the contour. Before we go

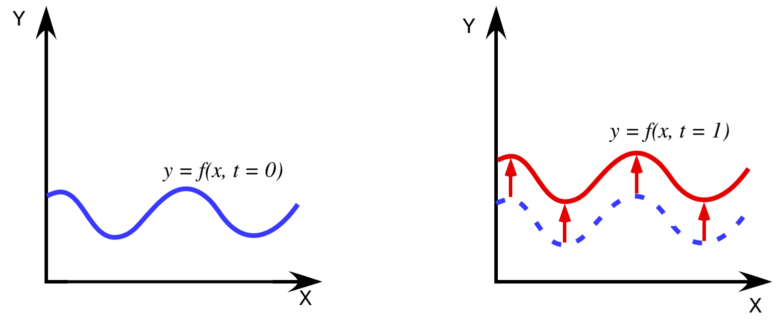


Figure 4.2: Functional representation of a sinusoidal-type curve in the XY plane.

into more details regarding level sets themselves, we will first take a high level look at the parameterized representation used for snakes so that its shortcomings can be viewed more clearly.

The key idea behind contour tracking/evolution is to be able to determine how the shape of the contour will change in future time instants. When considering potential solutions, one might initially visualize this problem using a "functional representation". Consider Fig. 4.2, which shows a sinusoidal curve in the XY plane. One could describe this image as a function, where every x value maps to a single y value. The graphs in Fig. 4.2 can be considered as curves and it may be desired to determine how that contour changes over time. Thus, on the left of Fig. 4.2, the contour is described as the function $y = f(x, t = 0)$ whereas on the right, we have the functional form of the contour at a later time, $y = f(x, t = 1)$.

The aforementioned functional representation of a contour seems like a reasonable approach until one categorical failure is noticed. Consider Figure 4.3. In this case, the contour could be considered to be the grass and the outline of the house. However, what is highlighted with this image is that, in general, an image curve can not be represented as a function. Along the highlighted column, for example, where $X = x$ there are three associated y values. These three values correspond to the grass, as well as the bottom and top of the roof. Given this limitation of the functional representation, an alternative method of describing a curve is needed.

Undoubtedly this problem was noted when researchers were developing the first snake trackers. In an attempt to identify a solution to this coordinate system challenge, the parametric representation of a curve was proposed. With a parametric representation, one maps each (x, y) value of the Cartesian coordinate system to a different parametric variable, s. Different mappings between the Cartesian system and the parametric domain can be used, but this only effects the velocity at which one traces around the curve. In other words, regardless of the parameterization selected,

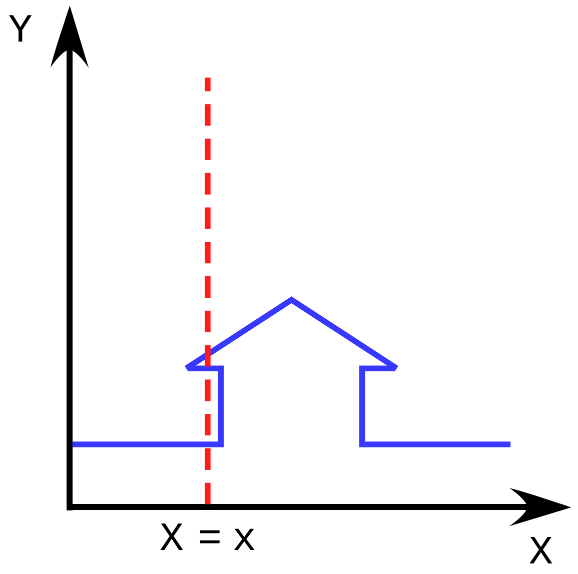


Figure 4.3: Functional representation of a house image in the XY plane.

the shape of the contour will remain consistent. The parameterized representation removes the explicit dependence on a coordinate system and not surprisingly, initial contour based trackers (e.g., snakes) made use of this representation.

Figure 4.4 shows an example of a curve represented as a discrete snake. In particular, there are discrete points sampled around the curve and these points are connected via “strings” that run between them. The arrows in Fig. 4.4 show the computed velocity of the snake vertices. For the sake of argument, these velocities have been chosen based on the contour curvature. Points in convex regions will have velocities that point toward the center of the contour whereas points in concave regions will move away from the center. Furthermore, the velocity magnitude of these vertices depends on the amount of curvature at each point. Although this example seems straightforward enough with a parameterized representation, problems are soon encountered. Using the velocities suggested in Fig. 4.4, the vertex points will quickly overlap and it becomes somewhat challenging to keep track of the connections. Furthermore, maintaining equal spacing between discrete vertices is another clear challenge that will arise. One method to alleviate these problems is to occasionally resample the curve. However, this solution is not overly elegant.

Another even more significant challenge when using parameterized curves can be seen in Fig. 4.5. Here, two objects are shown whose contours are expanding. Under a parametric representation, these two curves would be independently tracked from the first frame onward. However, at a later time, these two contours may overlap with one another. As the right-hand side of Fig. 4.5 shows, discrete snake points from the left-

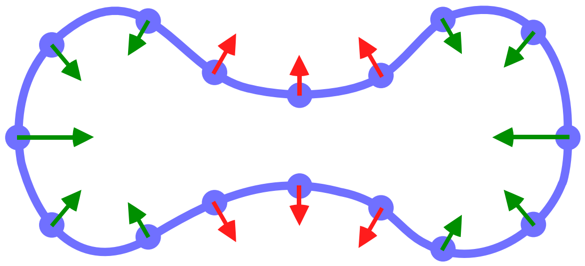


Figure 4.4: Curve represented as discrete points. Velocity of the discrete points is based on contour curvature.

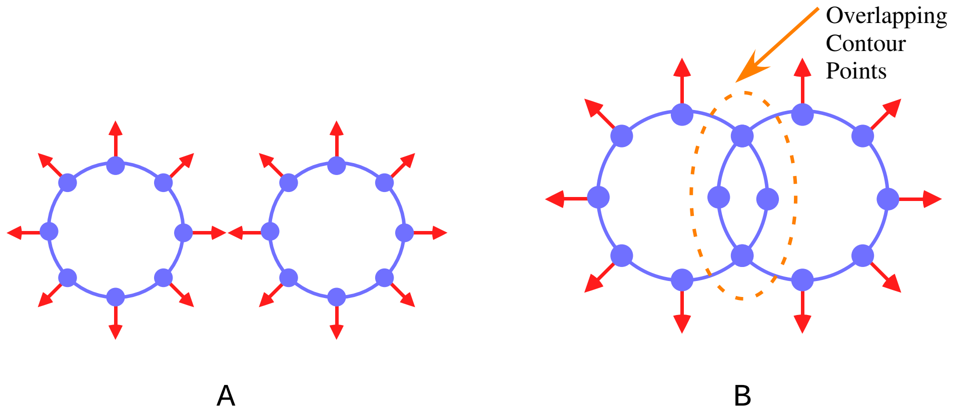


Figure 4.5: Two discrete contours merging. Discrete contours cannot easily handle topological changes.

hand circle are now inside the contour on the right and vice versa. In most situations, the ideal solution would be to somehow combine the two snakes into a single snake and only maintain the vertices that appear at the exterior of the combined region. Thus, with snakes, the process of merging multiple contours (or splitting a single contour) is not a natural process. Implementing these features with snakes is neither trivial nor computationally efficient. One could imagine that, at the very least, mechanisms would be needed for detecting when snakes overlap, determining which snake vertices are overlapping, and resampling the combined contour. In summary, if topological changes occur when tracking with snakes, a higher level system component would be needed for monitoring the individual snakes and managing how they interact with one another.

It appears that both of the immediate and obvious curve representations (functional and parametric) have their significant, individual shortcomings. Thankfully, level sets provide a third representation. Level sets once again return to a fixed coordinate system in favor of an explicit parameterization. The trick is to use a space with a dimensionality that is one higher than the true dimension of the curve. For example, in this paper, we are only concerned with two-dimensional curves in the image domain. Accordingly, with a level sets representation, a three dimensional coordinate frame for tracking the contour of interest is used. Furthermore, rather than explicitly tracking the motion of a contour, a 3D surface will be explicitly moved.

Conceptually, level sets are quite straightforward. The first aspect of level sets that must be considered is: "how should this 3D surface be constructed?" Essentially, to construct a surface, each (x, y) point needs to be assigned a height. Mathematically, the process of assigning heights can be written as the function $z = \phi(x, y, t = 0)$. A standard way of computing the height at each (x, y) point when creating a level set surface is to use the signed perpendicular distance from the 2D contour. Image points that are inside the closed contour will receive negative distances while those outside garner positive values. Naturally, (x, y) points along the contour itself are assigned a height of $z = 0$. Once we have the level set surface at any time instant, it is trivial to determine the 2D contour within the image. Specifically, all we have to do is slice a plane through the level set surface, keeping the plane level at a height of zero. The plane $z = 0$ is known at the zero-level set.

Having defined the level set surface and the correlation between the surface and an image contour, the method of moving the surface must be explained. In particular, the surface must be propagated in such a manner that the zero-level set always corresponds to the two dimensional contour. It has been shown (Osher & Fedkiw, 2003b,a) that the so-called "level-set equation" can be used for moving the surface

$$\phi_t + F\left(\phi_x^2 + \phi_y^2\right)^{\frac{1}{2}} = 0, \tag{4.17}$$

where the subscripts represent the partial derivatives of the surface in space and time and F is the speed of the contour that will be determined based on the application. For example, in image segmentation and tracking, typically high contrast edges are used to determine the contour velocity. Accordingly, the velocity, F, can be computed such that the surface will move quickly when the image gradient is small and more slowly (or stop) when the gradient is large.

Two final points that should be made with respect to the theory of level sets. Oftentimes the level set equation is modified to include a viscosity term that helps to ensure that cusps and sharp curves in the contour do not arise. Such image structures are undesirable because they lead to undefined derivatives and numerical instability. The modified level set equation is often written as

$$\phi_t + F\left(\phi_x^2 + \phi_y^2\right)^{\frac{1}{2}} = 0.1\kappa, \tag{4.18}$$

where κ is the local curvature of the surface. It should be noted that the there is nothing special about the "0.1" multiplicative factor. Other small, positive multiplicative factors could be used. For instance, a factor of "0.01" could be used, which would provide slightly less smoothing.

Another implementation detail regarding level sets is that of efficient computation. If the complete level set surface is re-estimated at each and every time instant, the system will be very slow. Various techniques have been proposed for efficiently computing the level set surface. Two of the most popular solutions are narrow band and fast marching methods. The narrow band approach simply updates the surface in the area immediately surrounding the zero-level set (both inside and outside of the contour). This solution can still be computationally expensive. As a result, when one knows that the surface will always be propagating in a particular direction (either inward or outward), the fast marching method can be used. This second algorithm achieves its efficiency by taking advantage of the fact that the contour is only moving in one direction and by using an effective sorting strategy.

4.2.2 Level Set Contour Trackers

With the above level set theory in place, we can now begin to elaborate upon trackers that have employed a level set curve representation. Although their system was designed for single frame segmentation rather than tracking, the work of Caselles et al. (Caselles, Kimmel & Sapiro, 1995) is one of the first and most influential examples of applying level sets to the image processing domain. This work makes two significant contributions. (1) The contour energy employed in most snake trackers (Kass, Witkin & Terzopoulos, 1988; Terzopoulos & Szeliski, 1992; Leymarie & Levine, 1993) is shown to be equivalent to finding the geodesic (minimum distance) curve in a particular space. (2) One of the first examples of applying a level set representation to the problem of curve evolution for segmentation is shown.

The work by Caselles et al. begins by drawing an equivalence between the standard energy formulation used for snakes, Eq. 4.3 and a new energy formulation based on the ideas of geodesic curves. In particular, it is shown that under either formulation, the optimal curve uncovered will be equivalent. To draw this equivalence, the authors first set w_2 (in Eq. 4.1) to zero and assume that the image potential energy is $P(v) = g(|\nabla I(v(s))|)^2$. The function, g, is based on the image gradient and has the property of asymptotically tending to zero as the input goes to infinity (e.g., a Gaussian function). This function has the desirable property of assigning the lowest energies to the strongest intensity edges. Substituting the selected values of w_2 and $P(v)$ into the snake energy equations, we obtain

$$E(v) = w_1 \int_0^1 |v_s|^2 ds + w_3 \int_0^1 g(|\nabla I(v(s))|)^2 ds. \tag{4.19}$$

where w_1 and w_3 are general constants. By assuming particular constant values and performing some algebraic manipulations, the authors show that the minimum of Eq.

4.19 is equivalent to the following

$$\min \int_0^1 g\left(\left|\nabla I\left(v\left(s\right)\right)\right|\right)\left|v_s\right| ds. \tag{4.20}$$

Intuitively, the problem of minimizing the internal and image energy of the contour has been transformed into the problem of finding the shortest path between two points in a non-Euclidean space. Specifically, each Euclidean element of length is weighted by $g\left(\left|\nabla I\left(v\left(s\right)\right)\right|\right)$ — information pertaining to the curve's boundary. Thus, in Eq. 4.20 we are not finding the classical minimum length between two points, but the minimum length that takes into account image properties.

The next step is to optimize the objective function described by Eq. 4.20. In a similar manner to snake-based systems, the Euler-Lagrange equation is employed. The Euler-Lagrange equation for Eq. 4.20 can be shown to be

$$\begin{aligned} \frac{\partial v\left(t\right)}{\partial t} &= g\left(I\right)\kappa\mathbf{N} - \left(\nabla g\mathbf{N}\right)\mathbf{N} \\ &= \beta\mathbf{N}, \end{aligned} \tag{4.21}$$

where κ is the curvature, $\mathbf{N}$ is the unit inward normal, and $\beta = g\left(I\right)\kappa - \left(\nabla g\mathbf{N}\right)$. It is important to stress that Eq. 4.21 describes the rate of change of the contour with respect to time, i.e. the contour's velocity. Note also that for each point along the contour, the velocity is in the direction of the unit inward normal, weighted by the factor β.

Under the level sets formulation, as discussed in the previous subsection, we want to relate the evolution of the curve to the evolution of a surface. To compute the motion of the corresponding surface, the authors use the following theorem. Assume that the velocity of the curve is known and is of the form $v_t = \beta\mathbf{N}$. Note that this is in the same form as was just derived in Eq. 4.21. Given this velocity of the planar curve, it can be shown (Caselles, Kimmel & Sapiro, 1997) that the velocity of the level set surface that corresponds to contour, v, is

$$u_t = \beta\left|\nabla u\right|. \tag{4.22}$$

Thus, by employing the aforementioned theorem and performing a few algebraic manipulations, the motion of the corresponding surface can be shown to be

$$\frac{\partial u\left(t\right)}{\partial t} = g\left(I\right)\kappa\left|\nabla u\right| + \nabla g\nabla u. \tag{4.23}$$

By employing standard implementation techniques as described in (Osher & Fedkiw, 2003a,b), the above equation can be used for segmenting via the zero level set curve.

Although the authors do not use their level set-based system for tracking, they do present some single frame segmentation experiments. The results shown are not plentiful or overly impressive. However significant benefits of the level sets technique become immediately apparent. In one example, the authors segment two tools in a simple, high contrast situation. The beauty of this example is that it illustrates that when representing curves with level sets, the topology of the curve can change. At the beginning of optimization, the curve encompassed almost the entire image. However, as optimization proceeds, the single large contour breaks into one contour for each of the two tools. Even more impressive is the fact that one of the tools has a hole inside of its outer boundary. The level set formulation not only uncovers the outer boundary of the tool, but also the boundary of the inner hole. A secondary benefit seen in these examples is that, unlike snakes that require the initialized position to be very close to the optimum, when using level sets, initial contours can be placed far away from the true target. Of course, in order to find the correct edges, it is assumed that there are no other good image features ("good" being defined by the particular optimization function) between the initialized contour and the desired image boundary. In other words, level sets are not a panacea; they will still be trapped in local minima.

Arguably, the pinnacle of the basic level sets contour trackers was presented by Paragios and Deriche (Paragios & Deriche, 2000). In this work, three significant contributions were made. To begin, the authors presented a new algorithm for updating the level set surface in an efficient manner. Additionally, geodesic active contours were applied to the problem of detection and tracking. This is in contrast to the work of (Caselles, Kimmel & Sapiro, 1995) that purely focused on segmentation. Finally, in (Paragios & Deriche, 2000), simulations were performed on a number of illustrative video sequences.

In this work, the authors decompose the problem of tracking into separate detection and tracking algorithms that are performed on every frame of the video. The detection and tracking components each contribute their own geodesic energy terms. Furthermore, the two modules operate on two different forms of data-derived information. We will begin our discussion with a description of the detection component.

The main idea of the detection system is to bring the contour into close alignment with the target using very simple motion information. Specifically, let $D(x, y, t) = I(x, y, t) - I(x, y, t-1)$ be the temporal difference image between the current and previous frame. The authors' idea is to first attract the contour to this inter-frame change region. Instead of directly basing contour evolution on $D(x, y, t)$, a probabilistic motion boundary detector is first deployed to identify more precise motion boundary regions. As shown in Fig. 4.6, a pixel is deemed to exist on a motion boundary if its neighbors on one side are not moving, while on its other side its neighbors are in motion. To proceed, these neighborhood regions must be explicitly defined. To that end, the authors elect to quantize two dimensional space into four orientations (horizontal, vertical, and diagonals at 45 and 135 degrees). Four lines (corresponding to the four orientations) are drawn through the current pixel, as shown in Fig.

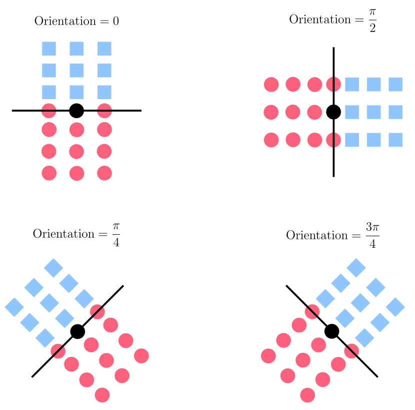

Orientation = 0
Orientation = $\frac{\pi}{2}$
Orientation = $\frac{\pi}{4}$
Orientation = $\frac{3\pi}{4}$

Figure 4.6: The four neighborhoods considered for the current point. The point under consideration is in the black. In each sub-figure one neighborhood is represented by pink circles while the other is denoted using blue squares.

4.6. With respect to a particular line, a pixel's two neighborhoods are defined as the surrounding pixels, on either side of the line.

For each orientation, the probability that the pixel is on a motion boundary (i.e. one neighborhood is moving and the other is non-moving) is computed. Among the four orientations, the one that provides the maximum probability is retained. This probability is then stored within a map at the (x, y) coordinates corresponding to the current pixel. This probability map is denoted as $I_D(x, y)$ and it shows the pixels that have the highest probability of being on a motion boundary.

The detection component of the system in (Paragios & Deriche, 2000) uses a geodesic energy that is directly based on the probability map, I_D. Although the detection component will not provide overly accurate contour delineation, the goal is simply to bring the curve into close alignment with the target. Subsequently, later stages of processing can then refine the initial solution. Mathematically, the detection geodesic energy integral is

$$E_D(v(s)) = \int_0^1 g(I_D(v(s)))\,|v_s|\,ds. \tag{4.24}$$

where g is a Gaussian function.

The tracking component of the system in (Paragios & Deriche, 2000) is simply an image gradient-based geodesic energy integral. This tracking energy is equivalent to that seen in (Caselles, Kimmel & Sapiro, 1995). For completeness, we will express this energy once again

$$E_T(v(s)) = \int_0^1 g(|\nabla I(v(s))|)\,|v_s|\,ds. \tag{4.25}$$

The detection and tracking equations are then combined into a single energy integral

$$E_T(v(s)) = \int_0^1 \gamma g(I_D(v(s)))\,|v_s| + (1-\gamma)\,g(|\nabla I(v(s))|)\,|v_s|\,ds, \tag{4.26}$$

where γ is a weight that adjusts the relative contribution of the detection and tracking components. Using level set theories, the corresponding update equation for the level set surface can be shown to be

$$\begin{aligned} u_t = \quad & \left[\gamma\left(g(I_D)\,\kappa + \nabla g(I_D)\frac{\nabla u}{|\nabla u|}\right) + \right. \\ & \left.(1-\gamma)\left(g(\nabla I)\,\kappa + \nabla g(|\nabla I|)\frac{\nabla u}{|\nabla u|}\right)\right]|\nabla u|\,. \end{aligned} \tag{4.27}$$

It should be noted that during optimization, a coarse-to-fine framework is employed. During coarse stages, almost all emphasis is placed on the detection component of the system (i.e. $\gamma \approx 1$). When processing the finer levels of the pyramid, the target's location has already been coarsely identified. Thus, during this refinement stage, the weight is updated to put more emphasis on the tracking component (i.e. $\gamma \approx 0$).

A second contribution of the system in (Paragios & Deriche, 2000) revolves around the actual implementation details of propagating the level set surface. Two of the most common surface update equations, fast marching and front propagation, each have their own shortcomings. The authors introduce a method they term "Hermes Algorithm" that attempts to combine the two methods to overcome their individual shortcomings. The algorithm proceeds by iteratively identifying the fastest moving points along the contour and updating the level set surface in a small region around that point. The algorithm continues until the surface no longer moves or a certain number of iterations have been performed.

The authors show that their proposed front propagation algorithm is both efficient and widely applicable. In fact, the algorithm is only slightly slower than fast marching methods and does not restrict the contour motion in any fashion. The one downside of this algorithm is that it only provides an approximate solution for the level set surface, u_t. The authors argue that the Hermes algorithm will still obtain a final solution that is equivalent to that obtained through other methods. However, Hermes algorithm violates constraints during intermediate stages of optimization for the sake of efficiency. In practice, it appears the intermediate approximations do not cause any difficulties.

As was mentioned at the outset of the discussion for Paragios and Deriche's work, one of its significant contributions was the varied and impressive results. A wide variety of results are shown, ranging from simple shapes to tracking vehicles, soccer players, and pedestrians. The ability of the system to operate on a wide variety of realistic sequences is quite impressive. Another positive aspect of the system is that the initial contour need not be placed in the immediate vicinity of the target. In the examples shown in (Paragios & Deriche, 2000), the initialized contour subtends almost the entire image frame. The system can handle such coarse initializations because the detection component operates on temporal difference motion information. Assuming that the target is the only moving object within the scene, it is straightforward for the system to converge to a coarse outline of the object (as the background is essentially zero under a temporal difference representation). Should multiple moving objects be present in the scene and the designer wishes to only follow one of them, a more precise initialization would certainly be required.

Even though the results in this work are quite impressive qualitatively, there are certainly a number of limitations to the system. Further inspection of the selected videos reveals some shortcomings. In particular, in the videos used, the target of interest is usually seen in front of a uniform background. In other words, there are no strong intensity gradients in the background within the immediate vicinity of the target. If there were high intensity background edges near the target, the system's performance may suffer. In particular, the authors' system makes the assumption that there are no background edges between an object's motion outline (found through the detection process) and its true border. Any large intensity gradients in this region have a chance of attracting the contour away from the true edge. The system also experiences difficulties with large motions, as it relies on the temporal differencing detection component to provide an initial contour that is completely outside of the true object boundary. When large motions occur, the contour recovered through the detection process may lie partially inside and partially outside the true target boundary. Another limitation with this system is that it has no ability to handle occlusion events. Nonetheless, this is overall quite a respectable paper, that substantially advanced the field of contour-based tracking.

In (Bertalmio, Sapiro & Randall, 2000), another approach was presented that provides very comparable results to those seen in (Paragios & Deriche, 2000). Specifically, the system of Bertalmio et al. can identify very accurate boundaries when the target is viewed in front of a nearly uniform background. However, in this work, the resulting system is also shown to be capable of locating the outline of a target with slightly more complex backgrounds so long as the initial contour is sufficiently close to the optimal solution. The most novel aspect of this work is that a pair of coupled partial differential equations are used during contour evolution. This pair of differential equations will be discussed next.

The first differential equation used in (Bertalmio, Sapiro & Randall, 2000) is termed the morphing equation. This equation deals directly with image domain features in two consecutive frames. The difference between the features in these two frames is measured and the features in the first frame are morphed toward the features in the second frame. Let $L\left(I\left(x,y,t\right)\right)$ be a function of an image I. This function is essentially a mask that eliminates the influence of all points outside of a three pixel band around the current contour. However, the intensity values of pixels within the band are left unchanged. This "thick contour" comprises the features used for the first differential equation. Specifically, $F\left(x,y,t\right) = L\left(I\left(x,y,t\right)\right)$, where F are the features in the current frame. The features of the previous frame are iteratively morphed toward the features in the current frame. To denote this iterative process, subscript notation is used, whereby, $F_0\left(x,y,t-1\right)$ indicates the initial features from frame $t-1$ before any morphing has occurred.

To derive a differential equation of the desired form, the authors need to compute the velocity of these image features. This velocity is computed as the pixel-wise difference between the two sets of features, $\beta\left(x,y,t\right) = F\left(x,y,t\right) - F_n\left(x,y,t-1\right)$, where n is the current iteration. The desired morphing differential equation can be written as

$$\frac{\partial F\left(x,y,t-1\right)}{\partial t} = \beta\left(x,y,t,\right)\left|\nabla F_1\left(x,y,t-1\right)\right|. \tag{4.28}$$

This differential equation leads to an estimate of the feature velocity $\beta\left(x,y,t,\right)$, which will subsequently be used in the second differential equation for evolving the contour

To obtain an approximation for the velocity of the tracking contour, the authors simply project the morphing velocity, β, onto an appropriate direction vector. Having obtained the velocity of the tracking contour $\hat{\beta}$, an expression for propagating the contour's corresponding level set surface is

$$\frac{\partial u\left(x,y,t\right)}{\partial t} = \hat{\beta}\left(x,y,t,\right)\left|\nabla u\left(x,y,t\right)\right|. \tag{4.29}$$

Thus, given an initial surface, Eq. 4.29 is used to propagate the level set surface.

The differential equations in this work given by Eq. 4.28 and Eq. 4.29 are coupled and work together in an iterative fashion to locate the optimal contour. Once the differential equation of Eq. 4.28 has completely morphed the image features from the previous frame into alignment with the image features in the current frame, the velocity β will be zero. At this point, the zero level set of the second differential equation should correspond to the desired image contour.

As mentioned, the results presented in this work are very similar in style to those seen in (Bertalmio, Sapiro & Randall, 2000). An interesting challenge that is unique to the level set formulation of contour trackers becomes apparent in some of the results presented in this paper. In particular, a downside of allowing natural topological changes is that oftentimes, spurious, unwanted contours will split off from the main contour. These “noise contours” may persist in the image even after the true target has moved further away. In general, it may be argued that these spurious contours can be considered as “shot-noise” and can be removed with an appropriate filter. However, this solution is not ideal.

Another interesting limitation of the system in (Bertalmio, Sapiro & Randall, 2000) is that the image features used are based on intensity matching within a thick contour region. Most other contour trackers discussed previously have relied more heavily on intensity gradients. Although there are potentially gains to be had by making use of intensity information in a contour tracking framework, there are also certain advantageous to using gradient information. Specifically, gradients should be more resilient to changes in illumination. Although it does not appear to be a problem in the simulations presented, the reliance on intensity information would suggest that the system in (Bertalmio, Sapiro & Randall, 2000) will not be well equipped to deal with changes in illumination. Nonetheless, with this work we begin to foreshadow a change in approaches with respect to contour trackers. Specifically, rather than just relying on gradient image information, researchers studying contour trackers at this time were beginning to look into making use of other modalities, including those that are derived from regions of larger support. This type of contour tracker will be the major theme of the next subsection.

4.3 Contour Trackers Using Regional Information

At the conclusion of the last section, we discussed a system that made use of information directly around the contour of interest (Bertalmio, Sapiro & Randall, 2000). In essence, the contour had a width that was greater than a single pixel and information within that thick contour was utilized during its propagation. This technique could be argued as starting to include regional information into the contour tracking process. This subsection continues with that same line of thought, only it takes the concept of regional information even further. For instance, some contour trackers make use of all the information that exists on either side of the contour. Other trackers use regional information collected along discrete normal lines to the contour. A number of these trackers will be presented and discussed.

One of the earlier works that attempted to make use of regional information in the contour tracking paradigm was seen in (Ronfard, 1994). In this work, it was realized that only so much can be achieved when using only edge-based features. Information contained within the target or background region can be invaluable when used in combination with intensity edge information. Furthermore, in some cases, a high-contrast edge may not exist between the foreground and target region, making the use of other modalities imperative for success. Noticing this limitation of purely edge-based contour trackers, Ronfard set out with the goal of outlining initial methods for including regional information in contour-based trackers (Ronfard, 1994).

In his work, Ronfard develops two trackers — a so-called "depth-adapting algorithm" and an "adaptive fusion" algorithm. The two algorithms are very similar, with the adaptive fusion method extending the depth-adapting algorithm to include more global information. Unlike the contour trackers discussed previously, Ronfard's tracker is more heuristics-based. We will begin with a discussion of Ronfard's depth-adapting algorithm (Ronfard, 1994).

As was mentioned during our discussion of standard snakes, one of the main components of a contour tracker is its energy function. Contour trackers that make use of regional information have to determine methods of incorporating this new source of information into an energy function. In (Ronfard, 1994), an approach similar in spirit to that of classical snakes was proposed. Notably, the energy of a region is defined according to

$$W^{region} = \int_{R_k} \left|\left| I(x,y) - \bar{I}_k \right|\right|^2 dxdy, \tag{4.30}$$

where $\bar{I}_k$ denotes the average intensity of that image region. In other words, the energy of the region is its intensity variance.

The energy of a region must somehow be related to the energy of a contour. To that end, the following formula is introduced

$$\begin{aligned} W^{contour}(v) &= D[R_{in}, R_{out}] \\ &= W^{region}(R_{in} \cup R_{out}) - W^{region}(R_{in}) - \\ &\quad\; W^{region}(R_{out}), \end{aligned} \tag{4.31}$$

where R_{in} is the target region (inside the contour), R_{out} is the background region (outside the contour), and $\cup$ denotes the union operator. The contour energy can be intuitively thought of as the energy required to merge two regions.

With region and contour energies defined, we now move on to discussing the method of moving the contour. In Ronfard's depth-adapting algorithm, a local view of contour evolution is taken. Specifically, the contour is sampled at a number of discrete points. At each of these sample points, a rod is defined that is oriented normal to the contour. This rod has a thickness of L and a length of $2P+1$. The sampled contour point under consideration is found at the very center of this thick rod. Figure 4.7 illustrates the rod/contour set-up. To evolve the contour, the author considers an image region that is much smaller than the area of the thick rod and that immediately surrounds the current contour point. This small, local region is defined as $\delta v(s)$. Given that a particular sampled contour point lies at the center

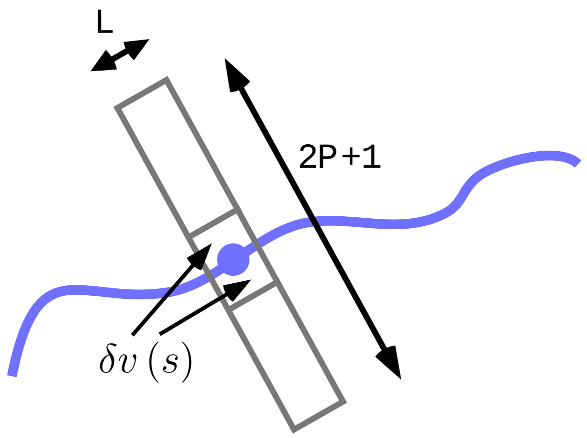


Figure 4.7: A contour and the thick rod representation around a discrete contour point used in (Ronfard, 1994).

of the rod and along the contour itself, we know that half of the rod lies outside the current target region while the other half of the rod lies inside. The rod region that lies outside of the current contour will be denoted as $R_{out}(v(s))$. Similarly, $R_{in}(v(s))$ denotes the area of the rod inside the target.

Keeping in mind the above definitions and Fig. 4.7, the change in energy experienced with a local perturbation of the contour is

$$\frac{\partial W}{\partial C} \approx \int_0^1 \left(D_{in}[v(s)] - D_{out}[v(s)]\right) \frac{ds}{|\Delta t(s)|}, \tag{4.32}$$

where $D_{in}[v(s)]$ is the so-called Ward distance between $R_{in}(v(s))$ and $\delta v(s)$. Naturally, $D_{out}[v(s)]$ is defined in an analogous manner. Furthermore, the Ward distance is defined as (Beaulieu & Goldberg, 1989)

$$D[R_A, R_B] = W^{region}(R_A \cup R_B) - W^{region}(R_A) - W^{region}(R_B), \tag{4.33}$$

where R_A and R_B are two image regions. So long as the local contour perturbation results in a positive change in energy (i.e. Eq. 4.32 stays positive), the current contour point will continue to move inward or outward in discrete steps until a maximum displacement is reached. This process is then repeated for all contour points.

A negative aspect of the resulting algorithm is that there are a number of parameters and they are not determined in a principled manner. Another downside of this depth-adapting algorithm is that all contour points are completely independent of one another. Ideally, more information would be shared between the contour points so that a more global perspective could be taken. Ronfard's extended version of the system, the adaptive diffusion algorithm, attempts to address this second issue.

The proposed solution for incorporating more global information is actually very straightforward. In Eq. 4.30, when the energy of a region is computed, the mean intensity of the region, $\bar{I}_k$, is used. Subsequently, when computing Eq. 4.32, the average intensity of the inner and outer regions of a rod are required. In the standard version of the algorithm, these mean intensities are computed using just the current rod. However, in the extended version of the algorithm, more global measures are obtained by averaging the inner (outer, respectively) regions of numerous neighboring rods. Contributions to the mean intensity used for a particular contour point are weighted such that its own rod has the most influence and the influence decreases as the distance increases. These averaged inner and outer region intensities, $\bar{I}^*_{in}$ and $\bar{I}^*_{out}$, are then used during the computation of Eq. 4.32. The authors report that this modification improves convergence properties around the contour boundaries.

Overall, the results shown in (Ronfard, 1994) are fairly comparable to other approaches of the time. Specifically, results are shown for segmenting and tracking that are quite similar to the work by Kass et al. (Kass, Witkin & Terzopoulos, 1988). It appears that there are some gains to be had by using the adaptive diffusion algorithm as it helps to maintain consistency and incorporate global information. However, the algorithm is more complex and harder to control than the simpler method. Additionally, the extended algorithm is more computationally intense.

Another interesting method that attempts to incorporate region information into a contour tracker is presented in (Jehan-Besson, Barlaud & Aubert, 2001). This work was published much later than Ronfard's and attempted to define a common framework for this class of contour trackers. Since a general framework was being proposed, attempts were made to make the approach as broadly applicable as possible. For instance, for the purpose of generality, the paper makes use of so-called "descriptor functions". Essentially, descriptor functions can be used to characterize a region or contour. The actual functions employed are application-dependent and can be defined by the designer.

Like many other contour-based trackers, this work defines an energy function that measures the goodness of a contour. The proposed energy is a function of the contour as well as the regions outside and inside (i.e. the background and foreground, respectively). Mathematically, the energy is written as

$$E\left(R_{in}, R_{out}, v\right) = \int\int_{R_{out}} k^{out} dxdy + \int\int_{R_{in}} k^{in} dxdy + \int_{v} k^{b} ds, \tag{4.34}$$

where k^{out}, k^{in}, and k^b are the descriptor functions for the background, foreground, and contour, respectively.

It is claimed that another major contribution of this work is the fact that time-dependent descriptors can be used. In other words, the contour energy itself is modified to include a dynamic variable that represents the optimization iteration number.

To fill this roll, the evolving parameter, τ, is introduced to keep track of the iteration number. Thus, the modified version of Eq. 4.34 is

$$\begin{aligned} E(\tau) &= \int\int_{R_{out(\tau)}} k^{out}(x,y,\tau)\,dxdy + \int\int_{R_{in(\tau)}} k^{in}(x,y,\tau)\,dxdy + \\ &\quad \int_{v(\tau)} k^{b}(x,y)\,ds. \end{aligned} \tag{4.35}$$

With standard snakes, the energy integral is formed and then is subsequently transformed using the Euler-Lagrange equation. At that point, an equation can be derived for updating the snake contour for each iteration (e.g., Eq. 4.11). However, in the proposed system of (Jehan-Besson, Barlaud & Aubert, 2001), the iteration variable is within the energy function itself. The direct consequence of this modification is that the change of the contour with respect iteration number, $\frac{\partial v}{\partial \tau}$, can be computed directly.

By consider the region-based and contour-based terms independently, the partial derivative of the energy with respect to τ can be shown to be,

$$\begin{aligned} \frac{\partial E}{\partial \tau} &= \int\int_{R_{out(\tau)}} \frac{\partial k^{out}}{\partial \tau} dxdy + \int\int_{R_{in(\tau)}} \frac{\partial k^{in}}{\partial \tau} dxdy + \\ &\quad \int_{v(\tau)} \left(k^{out} - k^{in} - k^{b}\cdot\kappa + \nabla k^{(b)}\cdot\mathbf{N}\right)(v\cdot\mathbf{N})\,ds. \end{aligned} \tag{4.36}$$

where $\frac{\partial k^{out}}{\partial \tau}$ and $\frac{\partial k^{in}}{\partial \tau}$ indicate the partial derivatives of the background and target descriptors with respect to the dynamic variable, respectively. Note that if the descriptor functions do not depend on τ, then the first two terms of Eq. 4.36 vanish.

The authors provide specific derivations of $\frac{\partial v}{\partial \tau}$, when particular descriptors functions are used. For example, in the simplest case where the regional descriptors are time-independent the energy derivative reduces to

$$\frac{\partial E}{\partial \tau} = \int_{v(\tau)} \left(k^{out} - k^{in} - k^{b}\cdot\kappa + \nabla k^{(b)}\cdot\mathbf{N}\right)(v\cdot\mathbf{N})\,ds. \tag{4.37}$$

Under these assumptions, the curve velocity partial differential equation can be shown to be

$$\begin{aligned} \frac{\partial v}{\partial \tau} &= \left(k^{in} - k^{out} + k^{b}\cdot\kappa - \nabla k^{(b)}\cdot\mathbf{N}\right)\mathbf{N} \\ &= \beta\mathbf{N}. \end{aligned} \tag{4.38}$$

Note that this is the same form as seen in other level sets contour trackers.

In (Jehan-Besson, Barlaud & Aubert, 2001), contours are represented and optimized using a level set formulation. In the standard fashion, the level set surface update equation is derived using the velocity of the contour, β, as defined above. The authors argue that the additional terms included in β (as a result of including the iteration variable directly in the energy equation) help to ensure that the minimum energy solution is determined in the fewest number of iterations.

The theoretical aspirations of (Jehan-Besson, Barlaud & Aubert, 2001) are quite noble and sound. The introduction of a general framework for introducing regional information into contour-based trackers is certainly a worthwhile contribution to the field. However, the experimental results presented in this work are not convincing of the overall utility of this framework. One might have expected the results to illustrate that the incorporation of the dynamic variable directly into the energy equation does indeed provide gains in efficiency, as suggested by the authors. Unfortunately, no such qualitative or quantitative evaluation is performed. In fact, the results presented in this work are based on very simplistic video sequences, especially for the time of publication. Furthermore, the descriptors selected are very simple — essentially reducing the resulting system to a contour-based background subtraction scheme. The selected energy function attempts to find the contour that maximizes the similarity between the model background and the region outside the contour while simultaneously minimizing the number of foreground pixels and the length of the contour. Although this approach appears to provide adequate results for the simple videos considered, it is difficult to speculate as to how well it would perform on more challenging videos. Additionally, the utility of the proposed framework is not clear, given the unconvincing results presented.

Another contour tracker that used regional information was presented by Chen et al. in 2001 (Chen, Rui & Huang, 2001). In a similar manner to the two works already discussed, a very inventive approach was taken when incorporating this new source of information. Clearly, researchers during this time had the desire to utilize regional information to improve contour trackers' performance; however, the best strategy for accomplishing this task was still very much undecided. In (Chen, Rui & Huang, 2001) a contour tracker was developed that includes region information by using a hidden Markov model (HMM) (Poritz, 1988). In this work, the idea of setting up an energy integral is avoided in favor of defining the behavior of the contour purely through the HMM's *observation model* and *transition probabilities*.

In brief, the contour tracker performs the following overall operations from frame-to-frame

1. Predict the position of the contour in the current frame.

2. Sample the predicted contour with discrete junction points.

3. Define normal line segments that pass through each discrete contour point.

4. Compute the observation likelihoods for each pixel along every normal line.

5. Compute the transition probabilities
6. Identify the optimal contour using the Viterbi algorithm (Forney, 1973) in conjunction with the computed probability distributions.
7. Fit an ellipse to the contour to increase stability.

The first three steps in the HMM-based tracker are very similar to other contour trackers that have already been discussed previously (e.g., (Ronfard, 1994)). To define the observation model, color and edge-based information are used and considered to be independent. For edge based features, it is possible that more than one edge will be detected along a particular normal line. Accordingly, the observation model expresses the probability of observing a particular set of edges, given that the system has selected a particular pixel along the normal line as its state. With respect to the color observation model, the authors make use of foreground and background color distributions. These two distributions represent the probability of observing a particular color pixel if it lies within the foreground and background regions, respectively.

When designing the proposed tracker, the authors of (Chen, Rui & Huang, 2001) face a similar predicament to that of Ronfard in his work (Ronfard, 1994). In particular, the observation probability provides a way of locally determining the optimal contour point along each normal line. However, in considering only a single normal line, no global information is considered. Since the tracker is based on a HMM framework, the transitions probabilities provide a natural mechanism for including global constraints. A very simple global constraint would be to encourage neighboring discrete contour points to be close to one another, in a Euclidean distance sense. Specifically, one simple state transition density function proposed in this work is

$$p\left(s_{\phi}|s_{\phi-1}\right) = c\exp\left(-\frac{\left(s_{\phi}-s_{\phi-1}\right)^{2}}{\sigma_{s}^{2}}\right), \tag{4.39}$$

where c is a normalization constant, σ_s^2 is the variance of the distribution, s_ϕ denotes the state of a particular normal line, and $s_{\phi-1}$ represents the neighboring normal line.

A more elaborate state transition function is also proposed in an attempt to make use of not only the global positional information (i.e. smoothness), but also global color information. Intuitively, the idea with the modified transition probabilities is to encourage the color patterns along neighboring normal lines to be similar or, at minimum, to vary in a smooth manner. For example, consider Fig. 4.8 which shows one normal line that has three blue pixels on the far left, followed by two black pixels in the middle, and four red pixels on its right end. As Fig. 4.8 shows, the new transition probabilities encourage neighboring normal lines to have similar color orderings. With this state transition function, multiple measurements along the current normal line must be matched to multiple measurements along the next normal line. Matching

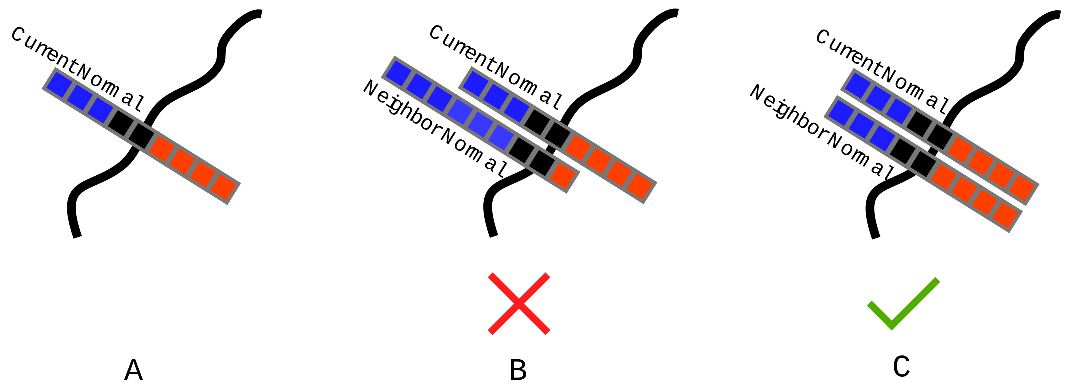


Figure 4.8: Global normal line color constraint used in (Chen, Rui & Huang, 2001).

sets of measurements is not a trivial problem and is something that was discussed in the context of statistical point trackers in Chapter 2. Borrowing ideas from that literature, the authors of (Chen, Rui & Huang, 2001) use a JPDAF to match the measurements in conjunction with a dynamic programming optimization procedure.

The proposed system is demonstrated in the application area of head tracking. The results indicate some success with moderate appearance changes, quick motions, and out of plane rotations. Furthermore, the authors claim that the system is moderately robust to changes in illumination, which it should be, given that discrete edge features are employed. Overall, the sequences show fairly typically head tracking performance in an office environment. The more interesting aspect of this work is the novel usage of hidden Markov models for contour based tracking.

The three trackers just described provide three very unique perspectives on how to incorporate regional information into the contour tracking paradigm. Several additional contour trackers will now be discussed (Mansouri, 2002; Yilmaz, Lin & Shah, 2004; Li, Miller, Weiss, Campbell & Kanade, 2006; Zhou, Hu, Chen & Hu, 2007). These works still provide quite disparate perspectives on the optimal method in which regional information should be incorporated. Nonetheless, there are two common themes that unite these papers: they all use levels set curve representations and they all wish to incorporate regional information when propagating the tracking contour. These are some of the more modern works in the field of contour tracking and two current trends are quite clear. The first trend is that representing contours with level sets is currently considered the best method, in general. Secondly, modalities other than gradient edge information must be incorporated into contour trackers in order to improve their reliability and robustness in realistic situations. With this motivation in mind, we will begin with a discussion of the tracker by Li et al. (Li, Miller, Weiss, Campbell & Kanade, 2006).

In (Li, Miller, Weiss, Campbell & Kanade, 2006), the goal is to simultaneously track numerous microscopic human cells. There is little about this tracker that is completely unique; however, the authors do combine a variety of standard tracking tools in a novel fashion, a process that yields an effective cell tracker. The four stages of the proposed tracker are as follows:

1. Cell detection.
2. Model tracking via level sets.
3. Prediction with motion filtering.
4. Track arbitration.

We will now discuss each of these stages in turn.

The process of cell detection in Li et al.'s tracker is accomplished with a pixel-wise classification process. Histograms representing the intensity distributions of the foreground and background regions are learned using training videos. Subsequently, pixel-wise maximum *a posterior* classification decisions can be made using these learned distributions. The result of this classification process is a binary map of the image. Each connected region within the binary map is considered to be a "cell candidate".

As mentioned, contour evolution is performed using a level set representation. For cell tracking, the usage of level sets appears to be almost necessary as there can be literally hundreds of cells within the field of view at any one time. Attempting to keep track of each contour individually with another method, such as standard snakes, would surely prove far too complicated and computationally intense. However, with the level sets formulation, topological changes are natural. As always, to use the level sets formulation, the contour energy must first be defined. Li et al. elect to define the energy of the contour based on three terms: region energy, E_{region}, which is based on pixel intensities; edge energy, E_{edge}, which is based on image gradients; and motion energy, E_{motion}, which is based on the contour's predicted motions.

In the cell tracking application considered in (Li, Miller, Weiss, Campbell & Kanade, 2006), level sets are beneficial for allowing topological changes in all but one situation. In particular, according to current knowledge, it is physically impossible for two cells to merge into one. As a result, the authors require a method of detecting and preventing cell fusion. An elegant solution to this problem is not provided in (Li, Miller, Weiss, Campbell & Kanade, 2006). Instead, a heuristic mechanism is used that relies on neighborhood functions and connected components. It appears that allowing and disallowing certain types of topological changes in a principled manner is an area of level set methods that can be further explored.

The third component of the system in (Li, Miller, Weiss, Campbell & Kanade, 2006) is motion filtering. A standard, off-the-shelf Kalman filter is used where the state of a cell is defined by the its centroid position, speed, and average intensity.

Measurements for this motion filter involve only the centroid position and mean intensity. The single unique aspect of the Kalman filter used in this work is that the motion model is learned offline using training data. Assuming and defining a constant, standard motion model (e.g., constant velocity, constant acceleration) is a much more popular approach than learning the dynamic model. However, the learning of dynamic models has certainly been explored before, particular by Blake, Isard, et al. in works such as (Blake, Isard & Reynard, 1994) and applied in the popular CONDENSATION tracker (Isard & Blake, 1996). These works will be discussed in more detail in the following section.

The track arbitrator is the final component of the proposed tracker from (Li, Miller, Weiss, Campbell & Kanade, 2006). This system module deals with high level system problems such as the initialization of new tracks and the removal of stale tracks. Additionally, this module will attempt to associate unmatched candidate cells (found through detection) with unmatched predicted cell regions (found using motion filtering). Most of the techniques employed in this system module are rule-based and incorporate cell tracking domain knowledge.

The experimental results reported in this paper are quite impressive given the number of cells being tracked simultaneously. Truthfully, the field of contour based tracking has come a long way since the work of Kass and Terzopoulos! This advancement is largely thanks for the level set formulation. In this work, quantitative performance analysis is also provided. However, these quantitative results are not overly useful as, to our knowledge, no other researchers use the same data. One shortcoming of the proposed approach is that it is quite computationally expensive, taking up to twenty seconds per frame for processing.

The remaining level set contour trackers in this category (Mansouri, 2002; Yilmaz, Lin & Shah, 2004; Zhou, Hu, Chen & Hu, 2007) take a similar, probabilistic approach to the problem. We will begin with the work of (Yilmaz, Lin & Shah, 2004), which provides extremely impressive overall tracking results. In this work, the authors elect to combine color features with the filter responses of spatial orientation selective filters. This work is actually somewhat analogous to the tracker from the previous section (Bertalmio, Sapiro & Randall, 2000) that made use of a thick contour representation. Despite this similarity between (Bertalmio, Sapiro & Randall, 2000) and (Yilmaz, Lin & Shah, 2004), it was felt that the work by Yilmaz et al. was better suited for this section as it incorporates features that are definitely region-based and their approach is quite similar to (Mansouri, 2002).

Under the probabilistic formulation presented in (Yilmaz, Lin & Shah, 2004), the goal is to discover the contour, v that will optimally divide the image into two regions corresponding to the foreground and the background. Mathematically, we want to find the contour that maximizes the probability $P_v = P\left(\phi\left(R^t\right)|I^t, v^{\bar{t}}\right)$ where ϕ is a function that divides the image pixels into two disparate sets of foreground and background pixels. Furthermore, R^t is a particular image region and $v^{\bar{t}}$ are all prior contours that have been found. By making use of Bayes' rule, the contour probability is approximated as

$$P_v \approx P\left(I^t|R^t_{target}, v^{\bar{t}}\right) P\left(I^t|R^t_{background}, v^{\bar{t}}\right) P\left(\phi\left(R^t\right)|v^{\bar{t}}\right) \tag{4.40}$$

where R_{target} and $R_{background}$ are the target and background regions, respectively. The first two terms on the right hand side can be computed via the target and background priors as well as the measured features. The third term considers the shape of the target, which is learned over time. To simplify notation, we will denote the three terms on the right hand side as $P_{R_{tar}}\left(I^t\right)$, $P_{R_{back}}\left(I^t\right)$, and P^t_s, respectively.

Following (Yilmaz, Lin & Shah, 2004), a maximum *a posterior* estimate of the curve is computed. This contour estimate is attained using the following

$$\hat{v}^t = \arg\max_v \prod_{x_1}\left[\prod_{x_2} P_{R_{tar}}\left(I^t\left(x_2\right)\right)\prod_{x_3} P_{R_{back}}\left(I^t\left(x_3\right)\right)P^t_s\right] \tag{4.41}$$

where x_1 are points along the contour. Note that in this work, the set of points, x_2, are points within a thick contour band inside of the current contour. Similarly, the set of background points, x_3, are defined within a thick band on the outside of the true contour. By taking the negative log-likelihood of Eq. 4.41, the maximum a posterior contour estimate can be changed into the problem of energy minimization. The end result of the aforementioned transformation is written as

$$\begin{aligned} E \quad = \quad & \int_{x_1}\left[\int\int_{x_2} -\log P_{R_{tar}}\left(I^t\left(x_2\right)\right)dx_2 + \right. \\ & \left. \int\int_{x_3} -\log P_{R_{back}}\left(I^t\left(x_3\right)\right)dx_3 - \log P^t_s\right]dx_1. \end{aligned} \tag{4.42}$$

With this equation, the standard level set machinery can essentially be employed to arrive at the surface update equation. In particular, minor modifications are made to Eq. 4.42 to parameterize the energy in terms of the distance along the contour, s. Subsequently, the Euler-Lagrange equation is used to compute the contour velocity from the energy integral. The contour velocity is then utilized in the standard manner in propagating the level set surface.

Thus, not surprisingly, one of the the biggest contributions of this work by Yilmaz et al. is the proposed energy functional that is derived using probability theory. The remaining mechanics have been derived previously. Another unique aspect of this work is that an explicit occlusion handling mechanism is introduced. Occlusion detection is handled in an ad hoc manner, using rules based on the targets' sizes to determine if an occlusion event has occurred.

Overall, the results displayed in this work are excellent. The tracker provides extremely tight boundary contours and can be used in a variety of situations including with moving cameras. Perhaps the most impressive results are those that illustrate the system's occlusion handling abilities. Although occlusion detection and handling is largely heuristics-based, in practice, it appears to perform extremely well. For humans who are partially or almost fully occluded, the system extrapolates very reasonable contours for the occluded region. It is challenging to ascertain the exact correctness of the contours in these occluded regions (as the true boundary is not visible) but visually, the results are very pleasing. The tightness of the target contours and the occlusion handling ability of this tracker are striking and are probably the most impressive results of all contour trackers considered in our review.

Although the results obtained in (Yilmaz, Lin & Shah, 2004) are qualitatively very good, it is worth noting that one significant assumption made in the proposed approach is not overly accurate (Zhou, Hu, Chen & Hu, 2007). In Eq. 4.40, it is assumed that the pixels in foreground and background regions are independent of one another. This assumption is typically sound as, generally speaking, the foreground and background have different distinguishing features (e.g., different color, texture, etc.). However, in Eq. 4.41 it is further assumed that pixels within the two regions are independent from one another. Typically, this assumption is not true. Noticing this shortcoming, an alternative probabilistic level sets approach was proposed in (Zhou, Hu, Chen & Hu, 2007). In this work, pixels within the foreground (background, respectively) are not assumed to be independent. Instead, a Markov Random field-based approach is used that considers the foreground/background states of a neighborhood of pixels during likelihood computations. During empirical evaluation, the proposed method is shown to outperform a contour tracker that makes the independence assumption between pixels.

In (Mansouri, 2002), another probabilistic contour tracker that uses regional information is proposed. Here, the general idea of the tracker is to estimate the target region in the current frame, R_1, given the previous and current images, I^n and I^{n+1}, as well as the previous target region R_0. Consider the equation

$$I^{n+1} \cdot \phi\left(x, y\right) = I^{n}\left(x, y, \right) + \mu\left(x, y\right), \tag{4.43}$$

where μ is zero-mean Gaussian noise and ϕ is a function that aligns the target region in the previous frame to the target region in the current frame. In other words, $\phi\left(R_0\right) = R_1$. Region warping trackers, which will be discussed more in the next chapter, attempt to estimate the transformation, ϕ. In this tracker by Mansouri, a different approach is explored. In this case, the author attempts to estimate the target region in the current frame, R_1, directly.

In a probabilistic maximum *a posterior* framework, the goal of estimating R_1 can be written as

$$\begin{aligned}\hat{R}_1 &= \arg\max_R P\left(R_1 = R | I^n, I^{n+1}, R_0\right) \\ &= \arg\max_R P\left(I^{n+1} | I^n, R_0, R_1 = R\right) P\left(R_1 = R | I^n, R_0\right) \qquad (4.44)\end{aligned}$$

where the second line of the above equation is obtained using Bayes' rule. Note that the first probability on the right hand side (second line) is the likelihood probability while the second term is the prior. In order to evaluate Eq. 4.44, a number of assumptions are required.

In this work, two major assumptions are made for the purpose of defining and simplifying the aforementioned likelihood and prior. Firstly, a conditional independence assumption is made which has the effect of limiting the allowable transformations that can take place on the target region. The second, partial dependence, assumption has the affect of asserting that the foreground and background have differing luminance and chrominance properties.

Given these assumptions, the likelihood probability is defined as $P\left(I^{n+1} | I^n, R_0, R_1 = R\right)$. This distribution is defined in a piecewise fashion, depending on whether the pixel under consideration lies within the foreground or background. The distributions are based on the pixel-wise squared intensity difference between the pixels in the current frame and the corresponding pixels in the previous frame (which are brought into alignment using the current transformation function, ϕ). By assuming that a maximum range of motion for the contour is known *a priori*, the set of allowable transformations can be reduced to the problem of performing a local neighborhood search around the current pixel.

In addition to determining the posterior, in order to estimate the optimal region in the current frame, the prior distribution must be determined. The prior distribution is defined as the length of the contour enclosing R_1, multiplied by a constant scaling term. Combing the log probabilities of the prior and posterior, an energy integral can be created. This energy integral can then be used in the level set formulation using the standard procedures.

The results presented in this work are good, but perhaps not as impressive as some of the other contour trackers considered in this review (e.g., (Yilmaz, Lin & Shah, 2004)). One of the experiments used the same car sequences as in (Koller, Daniilidis & Nagel, 1993). It could be argued that the performance of the system in (Mansouri, 2002) is drastically superior to Koller's work because, in this case, much less prior information is employed. Specifically, Mansouri's system does not make use of a 3D model during tracking. However, with reference to other contour-based trackers, the results on this sequence are somewhat disappointing. Throughout the experiments, it is clear that the extracted contour is quite ragged relative to other works. Additionally, in the car sequence the tracking contour tends to only capture the upper $\frac{1}{2}$ to $\frac{1}{3}$ of the car. The portion of the car that is successfully tracked is that which is not effected by shadows. The car sequence presented by Mansouri also clearly illustrates the problem that the level sets formulation has of generating small, spurious closed contours. Other results shown in (Mansouri, 2002) are more impressive than the car sequence used by Koller. In particular, the system is shown to be able to track successfully in cluttered environments. Furthermore, in another vehicle sequence, the tracker is not confused by an elongated cast shadow which would tend to trick other contour trackers, such as the one proposed in (Paragios & Deriche, 2000).

In summary, this section has focused on contour trackers that have elected to use more than just curve-derived information while tracking a target. This progression of contour trackers is certain a logical one as previously, there was significant regional information that was being discarded by this class of trackers. Arguably the biggest obstacle when using regional information with contour trackers is that of integrating the regional information into the paradigm. As evidenced by several papers in this section, a popular approach is to define an energy integral that includes regional information and employ level sets for curve propagation.

4.4 Other Contour Trackers

No matter how hard we try and what classes are use to divide the various contour trackers, there will be unique systems that simply do not fit properly into our defined categories. There is something to said for such systems, as it is does indicate that the researchers' work is very unique. However, at the same time, if a paper presents a truly ground-breaking tracker, it would be expected that a number of works would be subsequently published that are minor variations on the original idea. In this section, we will consider works from two different research groups. The first system arguably does not make significant contributions outside of its unique approach. The latter is a truly groundbreaking series of work that helped to redefine contour tracking and visual tracking, in general.

The first, unique work considered in this section is by Xu et al. (Xu & Ahuja, 2002) and considers the design of a contour tracker using graph cuts. Given an initial contour, the significant steps performed by this system are as follows:

1. Dilate the contour into a thick contour.

2. Represent the pixels within the thick contour as an adjacency graph.

3. Group the nodes along the inner boundary of the thick contour into a single graph source.

4. Group the nodes along the outer boundary of the thick contour into a single graph sink.

5. Find the minimum graph cut that separates source from sink for the graph created in Steps 2-4. The resulting graph cut optimally separates the inner boundary of the thick contour from the outer boundary.

6. Iterate Steps 1-5 until convergence.

The system deploys the above algorithm on two types of input data — frame differenced images and intensity images. The graph cut contour finding algorithm is first used on frame difference imagery to obtain a coarse contour estimate. Frame differencing eliminates the background clutter and allows the contour to converge to the interesting (i.e. moving) target within the scene. The authors argue that the resulting contours at this stage are neither precise nor well-centered, partially due to the problem with two frame differencing, discussed in Chapter 2 of this review. As a result, the initial contour is refined using the original intensity image. It is assumed that the initial contour will be close enough to the true target boundary that it will not become trapped by local minima caused by the background clutter. The authors argue that the contour would be distracted by background clutter if the first stage of processing with the frame difference image was not used. This two-stage approach is reminiscent of the work by Paragios et al. (Paragios & Deriche, 2000) where detection and tracking stages were used.

The end results of this system are not overly impressive, especially given other papers that were published in that same time frame. Consequently, few papers have extended this proposed approach. Nevertheless, this work is somewhat interesting in the context of this literature survey. Specifically, this work is yet another example of a contour tracker that takes the middle-ground approach of a “thick contour” (similar to (Bertalmio, Sapiro & Randall, 2000; Yilmaz, Lin & Shah, 2004)).

A series of works on the opposite end of the spectrum from that of (Xu & Ahuja, 2002) revolves around the CONDENSATION tracker by Isard and Blake (Isard & Blake, 1996). This system is the culmination of numerous proceeding works (e.g., (Blake, Isard & Reynard, 1994)). Furthermore, several extensions have been made, most notably by researchers affiliated with the same research group (e.g., (Maccormick & Blake, 2000)). The CONDENSATION tracker’s most prominent contribution was the mainstream introduction of particle filtering to visual tracking. Prior to this work, Kalman filtering was almost the exclusive predictive filter of choice in vision research. The advantages of particle filters were highlighted in (Isard & Blake, 1996) with respect to tracking in clutter and when multiple targets with similar appearance are present. Thanks to this work by Isard and Blake, particle filters have become almost as popular as Kalman filters for visual tracking.

We have already described most of Isard and Blake's CONDENSATION algorithm indirectly in Chapter 2 during our discussion of particle filters. The authors follow the particle filtering method exactly as described in our previous chapter. Accordingly, the main system components that must be defined are the target representation as well as the specific dynamic model and observation model that are employed.

In their work (Isard & Blake, 1996), the target is represented using parameterized, B-spline curves, $r(s,t)$. Furthermore, the space of contours is restricted by only allowing the contour to undergo certain types of transformations. This set of transformations can be determined *a priori* using key frames of a training sequence, or through other methods.

In this work, the dynamic model is represented using a second order linear difference equation that can be written as

$$x_t - \bar{x} = A\left(x_{t-1} - \bar{x}\right) + Bw_t \tag{4.45}$$

where x_t is the state at time t, $\bar{x}$ is the mean value of the state, A is matrix describing the deterministic motion properties, w_t is a vector of Gaussian random variables, and B represents the stochastic aspects of motion. At this stage, many trackers in the literature would simply assume a particular motion model (e.g., constant velocity) and that the stochastic component of the model is represented by zero-mean, Gaussian noise. In this case, a different approach is taken. Using training data of similar targets with similar dynamical properties, accurate motion parameters for A and B are learned. This learning of a target's dynamical properties was discussed in prior work by Blake et al. in (Blake, Isard & Reynard, 1994), where it was shown that learned dynamical models do indeed achieve superior performance than simple assumed dynamic models.

With respect to the observation model used in (Isard & Blake, 1996), an approach reminiscent of other works discussed in this chapter is used. In particular, a one dimensional observation model is first defined using normal lines at discrete contour points. Along each line, the probability distribution $p(\mathbf{z}|x)$ is defined. This distribution expresses the probability of observing a particular set of edges, $\mathbf{z}$, along the normal line, given that the system is in a particular state, x. In particular, $\mathbf{z}$ is a collection of one dimensional edge position measurements $\mathbf{z} = (z_1, z_2, \ldots, z_M)$, where M is the number of measurements. In this work, the one dimensional observation model is decomposed into two parts — one which represents the probability that the target is not visible, while the other considers the probability that one of the edge measurements corresponds to the true contour. Mathematically,

$$p_1(\mathbf{z}|x) = qp(\mathbf{z}|\text{clutter}) + \sum_{m=1}^{M} p(\mathbf{z}|x,\phi_m)\,P(\phi_m)\,, \tag{4.46}$$

where q is the probability that the target is occluded, and ϕ_m indicates that the correct measurement (i.e. that which is generated by the true contour) is z_m.

To precisely define Eq. 4.46, a number of assumptions are made. To begin, all measurements are assumed to be equally probable, making $P(\phi_m)$ constant for all m. Furthermore, clutter along the line is assumed to arise from a Poisson process with spatial density, λ. Finally, a true target measurement, if one is present, is assumed to be normally distributed with a standard deviation of σ. Combining these assumptions, the one dimensional observation distribution is

$$p_1(\mathbf{z}|x) \propto 1 + \frac{1}{\sqrt{2\pi}\sigma\alpha} \sum_m \exp\left(-\frac{(z_m - x)^2}{2\sigma^2}\right), \tag{4.47}$$

where $\alpha = q\lambda$. To further simply matters, it is assumed that only the closest measurement to the predicted position defined by the state will be included. This final assumption leads to

$$p_1^*(\mathbf{z}|x) \propto \exp\left(-\frac{1}{2\sigma^2} f(v_1;\mu)\right), \tag{4.48}$$

where $f(v;\mu) = \min(v^2, \mu^2)$ and μ is a spatial scale constant. Additionally, v is the difference between the state position and the closest measurement.

As was seen in other works, (Ronfard, 1994; Chen, Rui & Huang, 2001) the one dimensional observation probability must be extended to encompass all normal lines. To do so, the authors compute the product of the one dimensional densities that are evaluated independently for each of the normal lines. This product yields the required two dimensional observation probability used by the tracking system. Thus, having defined the dynamic model and observation model, the CONDENSATION particle filter proposed in (Isard & Blake, 1996) is essentially completely defined.

It should be noted that there have been a number of extensions to the original CONDENSATION tracker, one of which is presented in (Maccormick & Blake, 2000). In this particular extension, the authors make two contributions. First, they update the observation model so that it can more accurately deal with multiple targets. Additionally, a "partitioned sampling" scheme is proposed to reduce the number of particle samples required (and thus, processing time). Most of the extensions that have been reported on the CONDENSATION algorithm have been evolutionary refinements rather than revolutionary new achievements.

Returning to the original CONDENSATION tracker (Isard & Blake, 1996), the results shown in this work were striking for their time. Tracking results were shown on targets that move quickly and erratically in front of cluttered backgrounds. From the examples shown, the benefits of maintaining multiple hypotheses when performing visual tracking with distractors is clear. The authors report that standard Kalman filters obtained nowhere near the level of performance that can be attained via their particle filter tracker.

There are definitely significant downsides of the approach presented in (Isard & Blake, 1996) as well. The authors elected to incorporate a lot of prior information into their final CONDENSATION tracker. Hand-drawn templates are used at the outset. The dynamic models for the object are learned using training data. This process often involves iteratively running a standard Kalman filter to learn and refine the target dynamics. In short, a great deal of sequence-specific work was performed to obtain the high quality final results that are shown in the publication. Although the framework of the tracker allows for the tracking of any arbitrary object, it appears that the CONDENSATION tracker requires a significant amount of domain knowledge to achieve the best performance. This requirement limits the usefulness of the tracker somewhat as its ease of deployability is reduced. In line with that thought is the fact that it appears the CONDENSATION tracker is more often cited in the literature for its introduction of particle filtering to tracking, rather than papers by other researchers attempting to improve upon that particular contour-based implementation.

4.5 Recapitulation

To summarize, this chapter has presented trackers that propagate contours that delineate precise target boundaries. Early works include snake-based systems that used discrete curve parameterizations. Subsequently, a more elegant contour tracking framework involving level sets was introduced. Among other advantages, level set contour trackers allow for natural topological changes. As researchers began to see the limitations of using only information along the contour itself, techniques were proposed for introducing regional descriptors into the contour tracking framework. In the next chapter, we will explore trackers that forgo the use of a contours altogether and simply represent targets as contiguous regions.

Chapter 5

Region-Based Trackers

In this chapter, the various flavors of region-based trackers will be discussed. For the purpose of this review, a region-based tracker is defined as a tracker that maintains feature information across an area that represents the target support. The types of features that are used in conjunction with region trackers includes: color, texture, gradient, spatiotemporal energies, filter responses, and even combinations of the above modalities. The more significant difference between these trackers is the manner in which they represent and capture the target motion and move the tracking window from frame-to-frame. These methodological differences led to the categorization of region-based trackers presented here. Namely, we separate this class of trackers into the following subclasses: blob-trackers; pixel-wise template trackers; kernel and histogram trackers; and miscellaneous trackers.

5.1 Blob Trackers

In the space of all region-based trackers, one could argue that blob trackers are at one extreme whereas pixel-wise template trackers are at the other end of the spectrum. In particular, blob trackers will often collapse all information regarding a target into very basic summary statistics. For example, a blob tracker might only use the centroid position and average color of the target that is being followed. Pixel-wise trackers, on the other hand, retain some type of feature that describes the target at every pixel across the target support.

The most basic blob trackers follow a fairly intuitive series of steps. These steps can be summarized as follows:

1. Assume a stationary background (or that techniques can be performed beforehand that remove the background motion)

2. Detect changes in the foreground (e.g., moving objects) between the current and previous frame

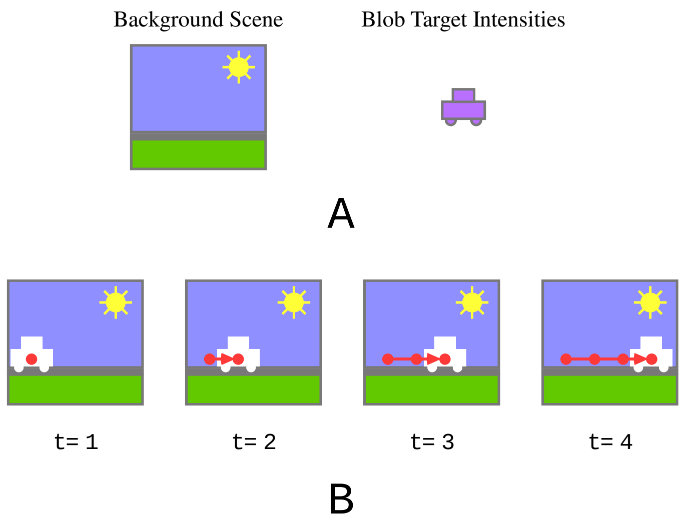


Figure 5.1: Illustrative example of using blob tracking to follow a car. In A, the background model and the intensities of the detected foreground blob are shown. In B, the foreground blobs are shown in white, the blob centroids are represented as red circles that are linked via red lines.

3. Associate the currently detected foreground objects with the tracked object(s) from previous frames

In other words, using the terminology and theory developed in Chapter 2, the tracking problem is essentially simplified to that of foreground detection and data association. Thus, the significant differentiating factor between blob-trackers in the literature are the tools used to solve each of these sub-problems and the manner in which these two tools are combined. Reference to specific blob-trackers, their contributions, and how they extend beyond this overly-simplified model will now be described.

Blob trackers rely very heavily on having an effective background subtraction scheme in place. As a result, it is hardly surprising that this sub-field of tracking has resulted in a number of very effective foreground detection schemes being developed (Wren, Azarbayejani, Darrell & Pentland, 1997; Stauffer & Grimson, 2000; Yin & Collins, 2007a). The background subtraction schemes used in (Wren, Azarbayejani, Darrell & Pentland, 1997; Stauffer & Grimson, 2000) have become standard solutions (both of which were described in Chapter 2) that are commonly used with minor modifications, even today. In fact, most of the other blob-trackers discussed in this section make use of one of the background schemes from (Wren, Azarbayejani, Darrell & Pentland, 1997; Stauffer & Grimson, 2000).

Given that fairly standard techniques for foreground detection were developed in conjunction with blob trackers, the other major differentiating factor between trackers of this class is the means by which data association is performed. Perhaps the simplest data association method was in (Yin & Collins, 2007a), where nearest neighbor association was utilized. To be fair, this work was more concerned with presenting a novel foreground detection scheme using belief propagation and Markov random fields. Obviously, this approach will be limited in its success when multiple moving targets are moving in close proximity.

Another more complex blob tracker made use of heuristic rules to track multiple targets in a closed environment (a children's playroom) (Intille, Davis & Bobick, 1997). Not unexpectedly, this system used a background subtraction scheme inspired by (Wren, Azarbayejani, Darrell & Pentland, 1997) and morphological operators to obtain blob regions. In addition to using positional information, the system in (Intille, Davis & Bobick, 1997) also utilizes blob color and size. Matrices for each type of feature modality are developed that indicate the matching scores between each object being tracked and the blobs located within the scene with respect to that particular feature. Subsequently, an overall matching matrix is obtained by taking the weighted contribution of matrices for each modality.

The matching process in (Intille, Davis & Bobick, 1997) is performed using nine heuristic stages, during which the weighting factors for each type of feature matrix are modified. During each stage, an overall matching matrix is computed and specific correspondences are established using a matching algorithm similar to that described in (Rangarajan & Shah, 1991). The overriding nine stage scheme is based on so-called "closed-world" assumptions. For example, the first stage attempts to connect known targets with detected blobs. The second stage links known targets with blobs that have already been assigned a target. In other words, multiple targets can be assigned to the same blob; the rationale being that oftentimes the targets will move close to one another causing the merging of blobs. However, when performing matching during this second stage, color feature information is ignored because the blob is assumed to consist of two or more targets and therefore, the average color is no longer meaningful. Similar heuristic strategies are used through the remaining matching stages.

Another interesting component of the system in (Intille, Davis & Bobick, 1997) is its initialization scheme and ability to analyze videos where targets enter and leave the scene. Since the system is being deployed in a specific children's playroom, the authors define a "door" area where entries and exits to the environment can be monitored. This mechanism allows the system to keep track of the number of targets within the environment, information which is then used during the matching process.

Although the system presented in (Intille, Davis & Bobick, 1997) is unique, it certainly has its limitations. The system's theoretical basis is not overly sound, relying on more heuristic strategies for performing matching. More importantly, their work points out a significant limitation of the blob-tracking strategy, especially in tight, indoor environments. Blob extraction, generally speaking, is not overly accurate in delineating targets and does not lead to consistent blobs across frames. Shadows, motion of individual parts, and noise can cause blobs to split and merge erratically, making it challenging to accurately track and maintain the state of the targets within the scene. The authors themselves admit that their system is very sensitive to threshold selection. For further improved results, a methodology change is suggested.

More recently, two systems (Yang, Li, Pan & Li, 2005; Takala & Pietikainen, 2007) were proposed that follow a very similar approach to that seen in (Intille, Davis & Bobick, 1997). Specifically, background subtraction and a similar heuristic matrix-style data association scheme are employed. In (Takala & Pietikainen, 2007), multiple modalities (color, texture, and motion) are used in an attempt make matching more robust. However, it appears that neither (Yang, Li, Pan & Li, 2005) nor (Takala & Pietikainen, 2007) advance this type of blob tracker significantly beyond the performance of the system by Intille et al.

Using (Intille, Davis & Bobick, 1997; Yang, Li, Pan & Li, 2005; Takala & Pietikainen, 2007) as an indication, there is a performance limit that can be reached when using heuristic techniques to perform data association. As one might expect, some blob-trackers have utilized more statistical approaches when performing data association (Wren, Azarbayejani, Darrell & Pentland, 1997; Rosales & Sclaroff, 1999; Stauffer & Grimson, 2000). In particular, (Wren, Azarbayejani, Darrell & Pentland, 1997; Rosales & Sclaroff, 1999) used the KF and the EKF, respectively, while (Stauffer & Grimson, 2000) incorporated an MHT-based approach. We will proceed with a discussion of these statistical-based methods, beginning with (Wren, Azarbayejani, Darrell & Pentland, 1997).

The work of (Wren, Azarbayejani, Darrell & Pentland, 1997) is quite famous for deriving a novel background subtraction scheme that made use of pixel color. However, it is equally as famous for developing an impressive (for the time), color-based tracker. A blob-model is developed for the target whereby a Gaussian blob is initialized for each portion of the human body that has a different color. Models typically included a separate blob for the head, hands, legs, feet, and torso. Prediction of the blobs' locations in the current frame are obtained using Kalman filters. Subsequently, a support map defining which pixel belonged to what blob (or the background) is generated by computing likelihood measures. The blob models are updated by this measurement information and the process repeats in the next frame. Although initially impressive, this tracker has its limitations. The tracker assumes a stationary background and that only a single target can be present in the scene. Although initialization is semi-automatic, it does require that the user strikes some pre-determined poses so that contour and color-based methods can be used to locate the position of the head and hands.

Another tracker that made use of the Kalman filter framework was presented in (Rosales & Sclaroff, 1999). Again, background subtraction is performed in a similar fashion to (Wren, Azarbayejani, Darrell & Pentland, 1997) and morphological operators and connected components analysis are used to group the foreground pixels into blobs. The authors employed a temporal coherency scheme to further improve the accuracy and consistency of foreground blobs across frames. To link the tracks between consecutive frames, an extended Kalman filter is used. A standard Kalman filter is not sufficient as the measurements consist of two bounding box corners and the system state is the target crop box's position in three dimensional space. Thus, this relationship between measurements and state is non-linear, necessitating the use of the extended Kalman filter. The remainder of their tracker development is consistent with the standard extended Kalman filtering approach.

One limitation of the system in (Rosales & Sclaroff, 1999) is that it assumes a stationary camera. Furthermore, the authors claim that their system can track multiple objects simultaneously, but no mention is made as to how data association is performed. Of course, the extended Kalman filter provides a means of predicting the target's location in subsequent frames which can be used to disambiguate in some situations. However, in more complex circumstances (e.g., targets in close proximity for a number of frames), the EKF's ability to deal with data association in the proposed framework is limited. The inclusion of shape or appearance information could potentially assist in offsetting this problem.

An attempt to provide even more robust data association was presented in (Stauffer & Grimson, 2000). This work is most commonly known for the background subtraction technique that it proposed, but its multiple hypothesis-based tracker is impressive as well. Specifically, the authors use shape and size information when linking targets with foreground blobs via a pool of Kalman filters. Their multiple-hypothesis tracker first links each Kalman filter with every blob that lies within its validation sphere. Information contained within the previous two frames is used to determine if new Kalman filters should be deployed for any of the unmatched blobs in the current frame. Finally, to ensure that the number of tracking hypothesis does not grow out of hand, tracks with low probability are deleted.

Before closing this section, we should briefly mention once again the system developed in (Ye, Tsotsos, Harley & Bennet, 2000). As mentioned in Section 2.7, this work essentially presented a blob tracker that also included active camera controls. A database of background images is maintained for a number of discrete camera settings that are used to completely monitor the environment. While tracking, background subtraction is performed between the current image frame and the database background image that corresponds to the camera's current settings. Morphological operators are subsequently used to group disparate foreground blobs into contiguous human targets. Although the tracking techniques in this work are crude, the main focus of monitoring an entire 3D space using a minimal number of discrete camera settings is a unique and substantial contribution.

Thus, to summarize, blob tracking can be very effective at tracking targets in a scene where the camera is stationary (or can be properly aligned) and the targets are far removed from one another. The technique begins to break down when targets move in close proximity. The use of statistical data association techniques can help alleviate this problem. However, the foreground regions obtained by blob trackers are typically not very consistent from frame-to-frame. This lack of consistency is another significant disadvantage of these trackers that must be overcome.

5.2 Pixel-Wise Template Tracking

For the purpose of this paper, a pixel-wise template tracker is defined as a tracker that maintains some type of feature descriptor at each pixel across the target's support. The most basic feature descriptor would simply be the intensity value. We will further divide this class of trackers into two subsections. The first sub-class has a rigid template of the target that can be moved around to locate the region in the current image that most closely resembles the target. The second sub-class also involves the movement of the target region around the image, but these trackers incorporate an explicit parametric motion model for moving the tracking window. Furthermore, these parametric models can be made more sophisticated than a simple translational model such that more complex target deformations can be performed, including rotation, shear, and scaling.

5.2.1 Rigid Template Tracking

This sub-class of trackers is a logical place where one might begin if he or she had no prior knowledge of computer vision. In this approach, the pixel-wise intensities (for simplicity, at this stage of the discussion) of an area within a reference image form a template that represents the target and the goal is to find the image region in the next frame that provides the best match. Again, it is important to emphasize the difference we are drawing between features and templates. In this example, the features used are pixel intensities. This is in contrast to the template, which is the collection of specific, numeric intensity values that define the appearance of the target. The relationship between features and templates can be connected to ideas seen in computer programming. In particular, one could imagine that features are analogous to the data types of variables (e.g., int, double), while templates are similar to the numeric values variables are assigned.

The above simplified, proposed template tracker is somewhat analogous to matched filtering. In particular, matched filtering theory addresses the problem of detecting a target within a signal of unknown origin. The optimal linear filter that maximizes the signal to noise ratio when detecting the target is a matched filter. A matched filter is defined as the signal representing the target that one is trying to detect. Thus, matched filtering is the process of detecting a target within a signal of unknown origin by convolving the signal by the target signal.

Keeping the ideas of matched filtering in mind, the steps that a very basic template matching tracker might perform are as follows:

1. Initialize the template on the target region within the first frame
2. Predict where the target will appear in the subsequent frame
3. Load the next frame
4. Match the template to image regions centered on the predicted target position and within a surrounding neighborhood search region
5. Select the location that provides the highest matching score as the current target center
6. Optionally update the target template
7. Repeat Steps 2 - 7

Initialization (Step 1) can be performed either manually or in an automated fashion, as was described in Chapter 2. The prediction mechanism utilized in Step 2 can range from using no prediction (i.e. using the position of the target in the previous frame) to using a scheme such as Kalman/particle filters. Prediction techniques were also discussed at length in Chapter 2. The matching method employed in Step 4 is a design choice, but standard sum of squared differences (SSD) or cross-correlation are commonly employed (Brown, Burschka & Hager, 2003; Derpanis, 2006). To make the matching algorithm more robust to changes in illumination, both of the above matching methods have normalized forms that can be used. In (Burt, Yen & Xu, 1982) a comparative analysis of some of the most common matching metrics is performed. Furthermore, Table 5.1 was graciously provided by Derpanis and provides a summary of the mathematical equations used for computing numerous matching metrics.

Although a template matching tracker could be developed that exactly follows the steps outlined at the beginning of this section, most modern algorithms incorporate additional and more sophisticated processing. The reason for this advancement is that, quite simply, the above algorithm will not work very well in general application settings. To begin with, the proposed algorithm will experience problems with illumination changes. Although using a normalized matching method (e.g., normalized SSD or normalized cross-correlation) will help to reduce the effects of illumination changes, the problem is not eliminated completely. For example, illumination changes that alter the pixel intensities globally will be accounted for by using a normalized matching method. However, point lights, shadows, or other lighting changes that affect only part of the scene (specifically, part of the target) will still lead to errors. Furthermore, this approach is not robust to partial or full occlusions. The proposed template matching tracker will also struggle in situations when there are multiple targets that share a similar appearance (e.g., tracking a person's head or body on a subway platform). Finally, the approach is not robust to rotations (in-plane or out-of-plane) or changes in scale. Incorporating a template-update mechanism will help to ameliorate some of these problems if the frame-rate is fast enough, but it is not a complete solution. The question then becomes: "how can template matching trackers be improved?" We will now discuss additional intensity template trackers whose complexity extends beyond that of the simplistic tracker mentioned above.

In (Nguyen, Worring & Boomgaard, 2001) a method was proposed to make template-based tracking more robust to changes in illumination and occlusions. When tracking the template between frames, the authors make use of a very simple warping scheme (warping will be discussed at length later in this chapter) that allows for scale changes. However, at the core, the system in (Nguyen, Worring & Boomgaard, 2001) is essentially performing a robust SSD template matching method. The contribution of this work is that of a unique method for updating the template such that illumination changes and occlusions can be accommodated. Specifically, rather than using Kalman filters as they are typically applied to tracking (for predicting the target position and smoothing the track), they are used for updating the template intensities in a principled manner.

A Kalman filter is deployed for each pixel in the template and the template is updated such that it is a combination of the previous template and the region in the current frame that provides the best match. A robust metric is also employed during the SSD computation which allows for the detection of outlier pixel matches. For outlier pixels, the template is not updated. Such a scenario arises during partial occlusion. Full occlusions are dealt with in a similar manner when a certain percentage of pixel matches are flagged as outliers. Although Kalman filters are used for updating the templates, the inter-frame tracking mechanism is very simplistic — the entire image is exhaustively searched for the best matching region.

Another template tracker, presented in (Beymer & Konolige, 1999), incorporates stereo information to overcome some of the abovementioned limitations with the standard template matching approach. The authors make use of disparity information to help guide and refine the correlation matching process. Once disparity images are obtained, background subtraction is performed within the disparity domain. Performing background subtraction using disparities has its advantages because disparities are less influenced by changes in illumination. Accordingly, there is less of a need for background model adaptation (unless it is expected that foreign objects will be placed in the scene and subsequently will become part of the background). Once foreground regions have been obtained, they are segmented into areas of roughly constant disparity. Binary correlation is then performed to identify humans within each disparity layer. The above steps all describe a detection module that is utilized on every frame.

Match Measure	*Definition*
Sum of Squared Differences (SSD)	$\sum_{j=-n}^{n}\sum_{i=-n}^{n}(I_1(x+i,y+j)-I_2(x+i+u,y+v+j))^2$
Normalized SSD	$\sum_{j=-n}^{n}\sum_{i=-n}^{n}\left(\frac{I_1(x+i,y+j)-\bar{I}_1}{\sqrt{\sum_{j=-n}^{n}\sum_{i=-n}^{n}(I_1(x+i,y+j)-\bar{I}_1)^2}}-\frac{I_2(x+i+u,y+v+j)-\bar{I}_2)^2}{\sqrt{\sum_{j=-n}^{n}\sum_{i=-n}^{n}(I_2(x+i+u,y+v+j)-\bar{I}_2)}}\right)^2$
Sum of Absolute Differences (SAD)	$\sum_{j=-n}^{n}\sum_{i=-n}^{n}\lvert I_1(x+i,y+j)-I_2(x+i+u,y+v+j)\rvert$
Cross-Correlation	$\sum_{j=-n}^{n}\sum_{i=-n}^{n}I_1(x+i,y+j)\cdot I_2(x+i+u,y+v+j)$
Normalized Cross-Correlation	$\frac{\sum_{j=-n}^{n}\sum_{i=-n}^{n}(I_1(x+i,y+j)-\bar{I}_1)\cdot(I_2(x+i+u,y+v+j)-\bar{I}_2))}{\sum_{j=-n}^{n}\sum_{i=-n}^{n}(I_1(x+i,y+j)-\bar{I}_1)^2\cdot(I_2(x+i+u,y+v+j)-\bar{I}_2)^2}$

Table 5.1: Common region-based match measures. $\bar{I}_1$ and $\bar{I}_2$ are the local average intensity. Adapted from (Derpanis, 2006).

The tracking module in (Beymer & Konolige, 1999) links the tracks together by using intensity correlation guided by disparity information computed during detection. In particular, a Kalman filter is used to predict the location of a human target in world coordinates. A weighted intensity template is convolved around the predicted position of the target. As the intensity template does not exactly adhere to the outline of the target (like a contour-based tracker would) the disparity-based weights used during convolution help to decrease the influence of background clutter. Finally, the position of the target is further refined by ensuring that the target remains within a single layer within the disparity map. Improving standard intensity-based template matching by incorporating disparity information is a novel idea that can help add some robustness to illumination and help deal with occlusion problems. On the other hand, the system setup requires a stereo rig, which is not quite as accessible as standard monocular cameras. Furthermore, disparity estimation is a challenging, unsolved problem itself (Brown, Burschka & Hager, 2003), meaning that the depth information obtained may not always be reliable.

In (Cham & Rehg, 1999), a template matching tracker was used as a simple tool in a much larger multiple hypothesis tracking framework. As discussed in Chapter 2, multiple hypothesis tracking can be useful when tracking multiple targets or when tracking an object through a cluttered environment where there are numerous false positive targets. The traditional MHT operated on targets that are described by discrete points. However, the authors of (Cham & Rehg, 1999) correctly point out that there is no simple and efficient "human figure detector" that can take an image as input and provide the accurate position of the human body. As a result, the authors incorporate multiple tracking hypotheses in a different manner. Their approach lies between that of a Kalman filter, where the state distribution, $p\left(\mathbf{x}_t|\mathbf{z}_t\right)$, is Gaussian and particle filtering, where samples are drawn over the entire distribution. Specifically, the authors represent the state distribution as a piecewise Gaussian distribution. With that major modification in mind, the algorithmic steps are only slightly varied from either Kalman or particle filtering. In particular, their system

1. Performs Kalman filter prediction to obtain $p\left(\mathbf{x}_t|\mathbf{z}_{t-1}\right)$ from the posterior distribution at the previous time step $p\left(\mathbf{x}_{t-1}|\mathbf{z}_{t-1}\right)$. Prediction is performed for the peak of each mode in the posterior distribution and a constant velocity model is used.

2. The measurement likelihood, $p\left(\mathbf{z}_t|\mathbf{x}_t\right)$ is computed by sampling from the predicted distribution and by computing the product of squared differences between the image regions and the template. To improve upon the predicted location within the image, a locally optimal match between the intensity template and the image is found using an iterative Gauss-Newton method. This optimal location is used to compute $p\left(\mathbf{z}_t|\mathbf{x}_t\right)$.

3. Using the already computed distributions, $p(\mathbf{x}_t|\mathbf{z}_t)$ is found, with special attention paid to ensure that the number of modes stays within an acceptable level.

Hence, as can be seen, in the more complex tracking machinery presented in (Cham & Rehg, 1999), template matching is used to provide an estimate of the measurement likelihood. The obvious benefit of allowing for non-Gaussian distributions during tracking was discussed in Chapter 2 when comparing the particle and Kalman filters. However, what is not immediately obvious is why the proposed approach is superior to particle filters. The authors argue that particle filtering would become too computationally intense in their particular application domain of tracking humans with high degree of freedom models. Particle filtering would require a great number of samples to properly represent the distributions in their entirety whereas the proposed approach only represents the distinct modes. In the results that are shown, it is not immediately clear that the centroid of the tracked object is improved by the more sophisticated tracking prediction machinery. However, the placements of the parts of the target (i.e. the limbs) indicate fewer errors, thus making pose estimation more accurate.

Another work that should be mentioned is (Olson, 2000). This research focuses less on designing a template matching tracker and more on deriving a fast template matching scheme with a superior, subpixel matching method. The proposed method can work with either binary edge or intensity templates. When considering edge pixel features, the proposed approach proceeds as follows. Let the target template of edge features be denoted as M. Furthermore, the edge features detected in the image are termed N. We will also define the current placement of the template within the image as the random variable $\mathbf{p}$. The goal of this template matching scheme is to determine the optimal value of $\mathbf{p}$.

To that end, a distance measure, D, is defined. This distance function measures the distance between each template edge pixel to the closest edge pixel in the image, given the current position of the template, $\mathbf{p}$. The notation for representing this set of distances between template and image features is: $\{D_1(\mathbf{p}), \ldots, D_m(\mathbf{p})\}$, where m is the number of edge features in the template. The likelihood that the target template corresponds to location p within the image is thus, computed according to

$$L(\mathbf{p}) = \prod_{i=1}^{m} f(D_i(\mathbf{p})), \tag{5.1}$$

where f is a probability density function. The PDF is composed of the summation of two terms — a term for inliers (a pixel/edge in the template is present within the new image) and one for outliers (a pixel/edge in the template is not present within the new image). The goal of (Olson, 2000) is to search throughout each image for the region that yields the maximum likelihood. To accomplish this task, a unique search strategy is employed.

The search strategy operates in a coarse-to-fine manner whereby the image is divided into a number of cells. Each cell is considered in turn and an upper bound on the maximum likelihood that can be obtained for that cell is computed. To begin this search process, a maximum allowable movement measure is computed. In particular, the template is first positioned at the center position, $\mathbf{c}$, of a cell. The system then determines the maximum distance any template pixel can move when the template is offset from $\mathbf{c}$ to any other position within the same cell. This maximum distance is denoted as $\Delta\mathbf{c}$. In essence, this distance is the maximum distance any template edge pixel can move while the template remains within the cell region.

Once $\Delta\mathbf{c}$ has been determined, a modified distance function is defined as follows

$$D_i^C = \max\left(D_i\left(\mathbf{c}\right) - \Delta\mathbf{c}, 0\right), \tag{5.2}$$

where C is the current cell. Equation 5.2 basically computes the best case scenario for an template edge pixel — that it moves the maximum distance (or less) to come into alignment with the nearest image edge pixel. Applying this best-cases scenario to all template edges, an upper-bound can be placed on Eq. 5.1

$$\max_{\mathbf{p}\in C} L\left(\mathbf{p}\right) \leq \prod_{i=1}^{n} f\left(D_i^C\right). \tag{5.3}$$

Thus, at a particular scale, all cells are pruned except for those that surpass a threshold based on $\prod_{i=1}^{n} f\left(D_i^C\right)$ (alternatively, only the cell with the maximum upper bound is kept). The remaining cells are divided into smaller cells and the process repeats at this finer scale. The process is iterated until the size of the cells is the same as the template for which the search region is being performed. At the conclusion of the search, the authors obtain a single point within the image that corresponds to the maximum likelihood location of the target. Furthermore, it is suggested that a normal distribution is subsequently fit to this location based on the maximum likelihoods of the selected pixel as well as its neighbors. Fitting a normal distribution allows for sub-pixel accuracy and provides a measure of variance for the selected target location.

Advantages of the matching metric and search strategy presented in (Olson, 2000) are as follows. To begin, the matching method is cast in a probabilistic, maximum likelihood framework and is fairly well-founded in theory. Perhaps more importantly, the method provides a means of measuring the certainty of the measurements. The variance that is computed for the selected target position provides an indication of how confident we are in the exact pixel location of the target. Furthermore, the search strategy employed allows for the computation of an estimate of the likelihoods outside of the neighborhood surrounding the selected pixel. If the sum of the likelihood scores in other regions of the image is high, it is an indication that there are other regions that also match the template quite well.

Thus, this section has presented a number of trackers that represent the target with pixel-wise features. Matching between frames is performed by translating the template and finding the image region in the current frame that matches best. However, what if the target's motion is not only translational in nature? What if the target experiences scaling, shearing, or rotations? The trackers in the next section consider such transformations.

5.2.2 Template Tracking with Parametric Warping

Warping-based trackers, in a similar fashion to the rigid template matching trackers, make use of a target template or a reference image during the tracking process. For the purposes of this paper, an image warp (or image registration) is defined as a mapping from the pixels in some reference image to the corresponding/matching pixels in a second image. If no restrictions are placed upon this warping process, each pixel within the reference image is free to make a match association with any pixel in the second image. Typically, restrictions are placed on the movement of the pixels to constrain the problem. For example, a common constraint is to limit all pixels within a small spatial region to undergo the same motion. This constraint is one of the most common used in motion estimation. Another constraint enforces the motion of the pixels to follow a parametric model. Common parametric warps include pure translational warps and affine warps. As affine warping appears to be the most popular among the tracking community; this chapter will place its primary emphasis on this method. Affine warps are popular as they provide a first order approximation to observed image motion for 3D moving objects that have shallow relief (Derpanis, 2003).

Warping Basics

When performing image warps, the most common method of determining if two pixels correspond is via the intensity values. Although intensity-matching is by far the most popular, warping using other forms of pixel-wise descriptors is certainly possible (e.g., phase response of orientation selective filters (Jepson, Fleet & El-Maraghi, 2003)). For simplicity, we will generally assume that matching is based on intensity unless otherwise stated.

The modern formalization of image warping as it is commonly described today was first presented in (Bergen, Anandan, Hanna & Hingorani, 1992). In this seminal work, the authors realized that model-based motion estimation techniques could be cast as an image registration problem. Specifically, the authors realized that the model-based motion estimation techniques could be formulated in terms of the same

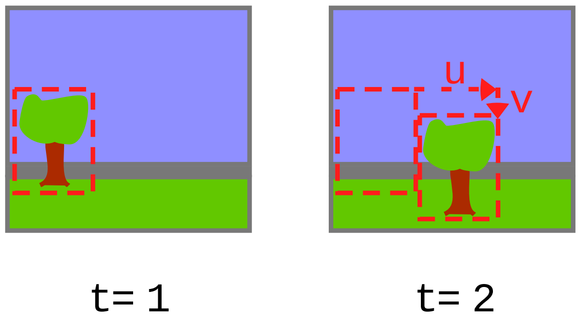


Figure 5.2: Computation of a translational warp that brings the tree into alignment.

SSD optimization problem (where different parameters are being optimized). We will return to a discussion of the work of Bergen et al., but first let us consider some of the prior image registration work.

One of the works that surely inspired Bergen et al. was presented in (Lucas & Kanade, 1981). This work focused on recovering translational warps that matched a target region in a reference frame to the optimal location in a second frame. The image warps primarily considered in this work were translational; thus, the parameters being estimated were the horizontal u and vertical v displacement of a target region in two images. Furthermore, the fact that this work primarily considers rigid translations of the target template makes it eerily similar to the rigid template trackers discussed above. However, the difference is that this work formalizes the problem in an image registration framework and thus, computes explicit translational warping parameters. A pictorial example of warping a target crop box that is tracking a tree is shown in Fig. 5.2.

Prior to the work of Lucas and Kanade, image registration was generally performed in an inefficient or ineffective manner. In particular, inefficient approaches included that of performing a brute-force search over the entire image to find the translational parameters. Ineffective approaches included those that performed gradient ascent style optimizations at full resolution. Although gradient ascent-based approaches appear fruitful, they frequently experience problems with local minima, especially when the desired displacement is large. In (Lucas & Kanade, 1981) an iterative Newton-Raphson method was proposed for performing translational registration efficiently and effectively. Results are improved by performing the optimization in a coarse-to-fine pyramid structure. In particular, with this coarse-to-fine approach, the registration is first performed at the coarse scales and the results at these scales are propagated as an initial estimation for the next finer scale. Coarse-to-fine strategies help to ameliorate the problems of large displacements and false positives due to high-frequency aliasing.

This work by Lucas and Kanade was definitely an excellent starting point for efficiently computing warping parameters. The ideas of image alignment presented in (Lucas & Kanade, 1981) certainly sparked researchers' collective interest and promoted significant research in the field. An excellent summary of the Lucas and Kanade algorithm (Lucas & Kanade, 1981) and its most noteworthy extensions can be found in (Baker, Gross & Matthews, 2004). However, instead of jumping ahead to the summary of 2004, we will take a more historic perspective, by making reference and elaborating upon some of the most significant works that expand upon the original ideas of Lucas and Kanade (Bergen, Anandan, Hanna & Hingorani, 1992; Black & Anandan, 1996).

The purely translation model presented in (Lucas & Kanade, 1981) severely limits the types of motions that can be properly recovered. It was a full 10 years later when Bergen et al. developed the common framework for model-based motion estimation techniques that has become a standard formulation for computing image warps (Bergen, Anandan, Hanna & Hingorani, 1992). The two primary contributions of this work are: 1) To show that model-based motion estimation techniques can be formulated in a standard SSD framework. 2) To propose a hierarchical, four step method where motion could be accurately computed using this framework.

The foundation upon which all motion estimation techniques presented in (Bergen, Anandan, Hanna & Hingorani, 1992) are based is the brightness constancy constraint:

$$I\left(\mathbf{x},t\right) = I\left(\mathbf{x}-\mathbf{u}\left(\mathbf{x}\right),t-1\right), \tag{5.4}$$

where $\mathbf{u}\left(\mathbf{x}\right) = \left\{u\left(x\right), v\left(x\right)\right\}$ is the horizontal and vertical displacement of the pixel, $\mathbf{x}$. In words, this assumption states that between frames, a target pixel may change position, but its brightness will remain constant. Of course, due to noise, varying perspectives, changing illumination, etc. the pixel values, in general, will not be identically equal. Accordingly, Eq. 5.4 can be restated as a problem of SSD error minimization

$$E\left(\mathbf{u}\right) = \sum_{\mathbf{x}} \left[I\left(\mathbf{x},t\right) - I\left(\mathbf{x}-\mathbf{u}\left(\mathbf{x}\right),t-1\right)\right]^2. \tag{5.5}$$

Ideally, this SSD term would yield a result of zero when the pixels from the reference target become perfectly aligned with the corresponding region in the new image. In realistic situations, zero-errors will not be attained, but the assumption is that if correct correspondence is made, then the SSD error will be minimized.

Also, note that as currently defined, the pixels within the target region are completely independent in the manner in which they move. The difference between the various model-based methods to motion estimation, arise from the different restrictions placed on the motion vector, $\mathbf{u}\left(\mathbf{x}\right)$. Many of the motion models define the motion vector parametrically whereas other methods will constrain the problem by assuming that all points within a small region must have a constant flow. Here, we will consider the addition of constraints involving a parametric motion model. To make the relationship between the motion vector and constraint parameters more explicit, we can write $\mathbf{u}(\mathbf{x};\mathbf{a})$, where $\mathbf{a}$ is the parameter vector.

A popular method for determining the optimal motion parameters is using so-called gradient methods. Under such an approach, a first order approximation of $I(\mathbf{x}-\mathbf{u}(\mathbf{x}), t-1)$ is obtained by assuming the time differential between subsequent frames is very small (thus, the motion is also very small) and using a Taylor series approximation where only first order terms are retained. After some additional algebraic manipulations, the gradient-based formulation for motion estimation can be written as:

$$\begin{aligned} E(a) &= \sum_{\mathbf{x}} \left(I_x(x,y,t)\,u(x,y;\mathbf{a}) + I_y(x,y,t)\,v(x,y;\mathbf{a}) + I_t(x,y,t)\right)^2 \\ &= \sum_{\mathbf{x}} \left((\nabla I)^T \mathbf{u}(\mathbf{a}) + I_t\right)^2, \end{aligned} \tag{5.6}$$

where I_x, I_y, and I_t denote the spatial and temporal image derivatives.

As mentioned previously, for this chapter we are primarily concerned with motion estimation using an affine motion model (due to its prevalence in the tracking literature). The affine motion model is defined as:

$$\mathbf{u}(x,y;\mathbf{a}) = \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} a_0 + a_1 x + a_2 y \\ a_3 + a_4 x + a_5 y \end{bmatrix}. \tag{5.7}$$

An affine transformation allows for the following transformations: translation, rotation, scale, and shear. Examples of these transformations are shown in Fig. 5.3. In (Bergen, Anandan, Hanna & Hingorani, 1992) various other motion models are detailed, but for the sake of brevity, they will be omitted from this discussion.

Hence, the above formulations have illustrated how the motion estimation problem can be cast in a common SSD minimization framework, independent of what motion model is used. The second significant contribution (Bergen, Anandan, Hanna & Hingorani, 1992) was that of presenting a hierarchical method for accurately computing the motion within this framework. The authors propose the following four steps:

1. Compute a pyramid representation for the frames under analysis
2. Compute a motion estimate using the abovementioned SSD framework
3. Warp the second image toward the first image using the computed motion parameters
4. Propagate the current motion estimates to the next finer level of the pyramid and refine the estimate based on this higher frequency information

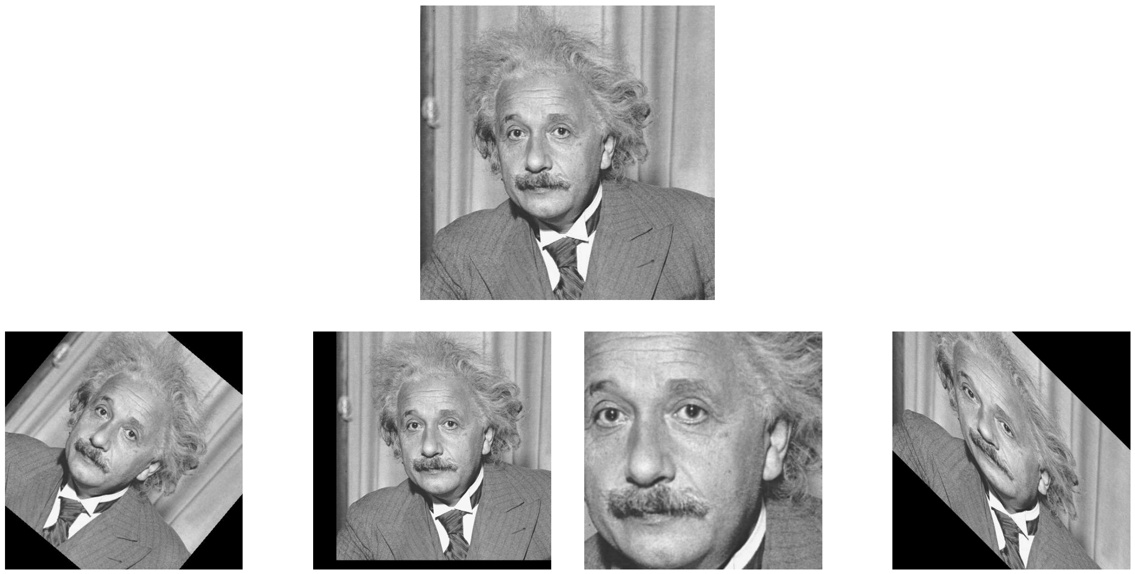


Figure 5.3: Allowable affine transformation. The first row shows the original image. From left to right, the second row shows examples of rotating, translating, scaling, and shearing, respectively

As can be seen, many of the same ideas that were proposed and shown to be effective in (Lucas & Kanade, 1981) (e.g., pyramid formulation) are once again used in (Bergen, Anandan, Hanna & Hingorani, 1992). These four major steps continue to be employed with little modification in many of the modern tracking works.

Although the founding framework for model-based motion estimation was laid out in (Bergen, Anandan, Hanna & Hingorani, 1992), there were limitations to the approach. One of the most glaring issues with the proposed formulation was the use of the SSD error metric. It is well-known that the SSD error measure is greatly affected by data outliers. Figure 5.4 shows an example of fitting a straight line to a series of points. When a non-robust metric (e.g., SSD) is used, the single outlier has a significant influence on the computed line. When a robust measure is utilized during line estimation, a much more appropriate solution is obtained. Thus, when using an approach such as that proposed in (Bergen, Anandan, Hanna & Hingorani, 1992) for estimating motion, the number of outliers must be minimal. Furthermore, if two dominant motions are present within the scene under analysis, an approach based on (Bergen, Anandan, Hanna & Hingorani, 1992) could yield unexpected results. Under this approach, one might expect that the system would compute an average motion that lies between the two dominant motions of the image region. This is essentially the worst result that could be attained as neither motions is properly modeled. However, in (Burt, 1991), it has been shown that using a coarse-to-fine strategy, such a motion estimation scheme can be designed that essentially "locks onto" a single image motion

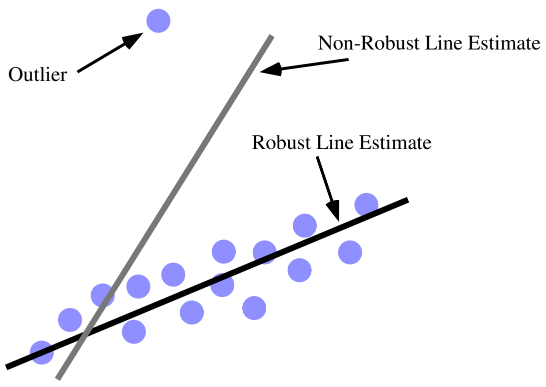


Figure 5.4: Line fitting example with a robust and non-robust error metric.

when two motions are present. Nevertheless, the SSD-based error function appears to have some significant shortcomings and accordingly, must be used with care.

To address many of these issues, a slightly modified approach was proposed (Black & Anandan, 1996) whereby the SSD measure is replaced with a robust estimator such as the truncated quadratic or Geman-McClure metric. The idea is quite intuitive, but the improvement can be used to great advantage. More formally, the parametric motion estimation equation, Eq. 5.6, is made more robust as follows:

$$E(\mathbf{a}) = \sum_{\mathbf{x}} \rho\Big((\nabla I)^T \mathbf{u}(\mathbf{a}) + I_t, \sigma \Big), \tag{5.8}$$

where σ is a scale parameter of the robust estimator, ρ. The Geman-McClure is a very popular robust estimator and will be the primary estimator considered in this paper. This estimator is mathematically defined as

$$\rho(x, \sigma) = \frac{x^2}{\sigma^2 + x^2}. \tag{5.9}$$

As mentioned previously, one advantage of using a robust framework for estimating motion is that the effects of outliers are greatly reduced. With a quadratic or squaring error function, errors rapidly increase in an unbounded fashion. Robust estimators do not exhibit this unbounded behavior. A truncated quadratic error measure simply reaches a plateau whereby all errors beyond some point all result in the same output. The Geman-McClure function is quite similar in shape, but the plateau is approached in an asymptotic fashion, rather than having a discrete cut-off point. Because of its smoother shape (leading to its differentiability), the Geman-McClure metric is often a desirable choice. The rationale of bounding the error output in this manner is that beyond a certain point, the actual error value is not relevant. In other words, once a point has been established as an outlier, it should have an equal effect on estimation as any other outlier.

A second advantage of the robust formulation is that outliers can be detected in a more principled fashion. Once outliers have been detected, they can potentially be removed from the estimation process so that a more accurate motion estimate for the inliers can be found. Moreover, if multiple strong motions within an image region exist, iterative estimation can be performed whereby the outliers at one estimation stage become the inliers of the subsequent estimation. In this manner, multiple motions within an image region can be estimated.

Trackers Using Image Warps

With the general warping machinery in place, we can now move on to discussing some specific tracking systems that use image registration techniques to follow a target of interest. The problems of image alignment, motion estimation, and tracking are very closely related and are certainly not always mutually exclusive. Although purely translation warping, as presented in (Lucas & Kanade, 1981), could feasibly be used to construct a simple tracker, in that particular work, it was used for the purpose of stereo vision, which is outside the scope of this review. Furthermore, (Bergen, Anandan, Hanna & Hingorani, 1992) is more concerned with the problem of global "motion estimation" (or camera motion), which is also not the core idea of this paper. It could be argued that global motion estimation is, in essence, tracking. Certainly this is true in the strictest sense, as motion estimation provides a mapping of pixels in one image to the corresponding pixels in a different image. However, "tracking" as it is defined for this review is more concerned with following a target which subtends only a portion of the entire image. Thus, we want to trace the movement of a particular subregion of interest within the image. Of course, these global motion estimation techniques could conceivably be applied to a target region within a larger image and perform motion estimation for only said target region. This idea is quite true and is in fact, the methodology used by most of the region warping trackers that will be described next.

Many trackers have been developed that directly make use of the robust motion estimation techniques proposed in (Black & Anandan, 1996). Three example trackers that fall into this category and also incorporate some aspect of training/learning are (Black & Jepson, 1998; Hager & Belhumeur, 1998; Avidan, 2001). In (Black & Jepson, 1998), a so called "EigenTracker" was developed. Eigen representations are popular in various areas of computer vision with the most famous example being the Eigenfaces system (Turk & Pentland, 1991). The general idea with an Eigen-based approach is to develop a basis set that captures the majority of the variability in the object class that is being modeled. For example, with Eigenfaces, a multitude of face examples are observed and these examples are used to construct a basis set that captures the general properties and variations amongst faces. Any given face in the training set can be approximately reconstructed by using a subset of these basis images. Eigen-based approaches have their advantages as the majority of the important information that is contained within a vast database of examples can be captured by keeping a small number of basis images. Thus, the space savings can be tremendous.

The EigenTracker in (Black & Jepson, 1998) follows the same general approach used in other Eigen-based works. Specifically, the tracker is trained to follow pop cans (specifically a Coke$^{\text{TM}}$ and 7-Up$^{\text{TM}}$ can). The authors train the system using a number of images of these cans from various viewpoints. The different viewpoints allow all the various paint patterns on the two types of cans to be observed and learned. Subsequently, a basis set is computed that captures the majority of information contained within the training examples. When an unknown pop can is observed, the authors use a robust method to perform reconstruction. Specifically, the follow optimization function is used, which has a from that is strikingly similar to Eq. 5.8,

$$E(\mathbf{c}) = \sum_{\mathbf{x}} \rho(I(\mathbf{x}) - [\mathbf{Uc}](\mathbf{x}), \sigma), \tag{5.10}$$

where

$$\mathbf{Uc} = \sum_i c_i U_i. \tag{5.11}$$

In words, **Uc** is the reconstructed image that is computed by summing each of the basis images U_i, weighted by the coefficients, c_i. Thus, the goal of Eq. 5.10 is to determine the optimal set of coefficients, $\mathbf{c}$, that minimize the error between the image and the pop can reconstructed from from basis vectors.

Another interesting component of Eq. 5.10 that should be noted is the fact that rather than using a SSD error measure, a robust metric is employed. The robust metric allows the authors to isolate outlier pixels. As alluded to previously, once a set of reconstruction coefficients, $\mathbf{c}$, have been computed, the error between the original image and the reconstructed version can be evaluated using the robust metric. Outliers are identified as the pixels that do not fit well with the current reconstructed version of the image. An outlier/inlier binary mask is generated with respect to the current reconstruction. With this mask in hand, the authors refine the reconstruction by using only inliers (previously, the outliers would have been detrimental to the estimation process). Furthermore, this mask enables a second round of reconstruction to be performed for the outlier pixels.

In (Black & Jepson, 1998) many convincing examples show how this iterative approach to estimation can be beneficial. For example, an artificial example is shown where the bottom portion of the pop can has paint from a Coke can whereas a small amount of the upper portion has paint from a 7-Up can. As the paint from the Coke can is dominant, during the first round of reconstruction, an image of a Coke can is produced from the basis vectors. Of course, the pixels in the image corresponding to 7-Up paint are outliers with respect to the Coke can reconstruction. As a result, these pixels can be used as the inliers for a second round of reconstruction. When this procedure is performed, the basis vectors correspondingly reconstruct a 7-Up can. Although this example is somewhat artificial, one can imagine other situations where it would be very powerful. For example, if a shadow covers only a part of the Coke can, this iterative approach could prove useful.

The portion of the system in (Black & Jepson, 1998) described thus far is not a tracker — it only focuses on the reconstruction problem. To achieve tracking using a Eigen-based approach, the authors include both the affine warping parameters as well as the Eigen coefficients into the same optimization framework as follows

$$E\left(\mathbf{c},\mathbf{a}\right)=\sum_{\mathbf{x}}\rho(I\left(\mathbf{x}+\mathbf{u}\left(\mathbf{x},\mathbf{a}\right)\right)-\left[\mathbf{Uc}\right]\left(\mathbf{x}\right),\sigma). \tag{5.12}$$

In essence, this equation represents the error between the variable image region, subject to a warp, and the reconstructed Eigen image. The problem now becomes that of optimizing both the warping parameters, **a**, and the Eigen coefficients, **c**. To perform this optimization, the authors iterate back and forth between holding one set of parameters constant and optimizing the other until the updates become negligible.

Another similar approach was presented in (Hager & Belhumeur, 1998). In this particular work, the robust estimator was once again included in the interest of adding robustness to partial occlusion. Instead of determining the image warp based on Eigen images, this work primarily used pixel intensities. However, a learning component was included in an attempt to reduce the affects of illumination changes in a manner reminiscent of the work in (Black & Jepson, 1998). Notably, in (Hager & Belhumeur, 1998), a basis set is constructed from a set of examples that view an object under an array of illumination conditions. Hence, this basis set can approximately reconstruct the object under any linear combination of illumination changes observed during training. These basis vectors are used to compensate for illumination using the following equation

$$E\left(\mathbf{b},\lambda\right)=\sum_{\mathbf{x}}\rho(I\left(\mathbf{x}+\mathbf{u}\left(\mathbf{x},\mathbf{a}\right)\right)-T\left(\mathbf{x}\right)-\mathbf{b}\lambda,\sigma), \tag{5.13}$$

where **b** are the basis vectors, λ are the corresponding weights, and T is the intensity-based template of the object. This solution is very similar to that proposed in (Black & Jepson, 1998), but rather than modeling the entire target as an Eigen image, only the illumination changes are modeled using a basis set.

The trackers proposed in (Black & Jepson, 1998) and (Hager & Belhumeur, 1998) were quite revolutionary at the time, but they do have limitations. Both are learning-based approaches and with regard to illumination changes, only linear combinations of the changes observed during training can be explicitly handled during tracking. Thus, although the systems in these works propose methods to deal with some changes in illumination, it is only a partial solution. Arguably, it would be ideal if illumination changes could be dealt with at the feature-level rather than incorporating an off-line training process.

Another work that is conceptually somewhat similar to the above two approaches was presented in (Avidan, 2001). Again, a learning-based system is deployed in a motion estimation framework. Here, motion is estimated not using the brightness constancy constraint, but by maximizing the classification score of a support vector machine (SVM) classifier. Mathematically, the equation used is

$$E(\mathbf{u}) = \sum_j y_j \alpha_j k\big(I\left(\mathbf{x} + \mathbf{u}, \mathbf{q}_j\right)\big), \tag{5.14}$$

where $\mathbf{q}_j$ are the support vectors, α_j are the support vector Lagrange multipliers, and y_j are the support vector signs. Thus, maximization is performed to find the flow, u, v that results in the highest SVM classification score. It would be possible to use a parametric motion model similar to those seen previously in this section; however, this difference does not alter the conceptual contribution of this paper.

The warping trackers discussed so far have all shared one common theme — using learning techniques to accommodate tracking. Such a strategy certainly has its place in the field of tracking, but the severe limitation of this strategy is that one must have knowledge of what will be tracked beforehand and develop a specific classifier that recognizes this class of object. There are an interesting array of problems in the field of tracking where one does not have the luxury of such information. For instance, in typical surveillance settings, the variety of moving objects that could be observed is staggering (e.g., vehicles, animals, humans, man-made objects). How can one effectively develop classifiers for all possible objects that might enter a general surveillance scene? To develop a tracking system that relied on classifying an object in order to perform tracking would not be feasible. The classification algorithms available presently are simply not sufficiently reliable. Moreover, it would be extremely difficult and time-intensive to construct classifiers for each different type of object. Although learning-based, object-specific trackers have their place, there is clearly a need for trackers that can follow general targets.

There do exist warping trackers in the literature that do not require offline training components (Jepson, Fleet & El-Maraghi, 2003; Enzweiler, Wildes & Herpers, 2005; Ross, Lim & Lin, 2008). In (Ross, Lim & Lin, 2008), for example, an Eigen representation, similar to that seen in the work of Black et al. is employed. However, in this more recent system, learning of the Eigen representation is performed in an online fashion, allowing for the tracking of arbitrary objects. A further difference between the two Eigen-based approaches is that in (Ross, Lim & Lin, 2008), a particle framework is used to estimate the affine motion parameters. Impressive results are provided for the problem of face tracking where the target undergoes significant pose, appearance, and illumination changes.

In (Enzweiler, Wildes & Herpers, 2005), a tracker was developed that estimated affine motion parameters with a slightly modified version of Eq. 5.8. Specifically, a pixel-wise factor, w is included that weights the result of the robust estimator. This weighting factor is computed using spatiotemporal oriented energies. Specifically, each weight measures the maximum coherent motion energy of a pixel in the horizontal or vertical direction. The rationale for including this weight is that pixels that are moving coherently are typically part of the target being tracked (i.e. they are not part of the background clutter) and should have more influence on the motion estimate. Although the matching between the template and the current image is performed using pixel intensities, this work showed that new types of features are gradually making their way into tracking.

The work of Jepson et al. in (Jepson, Fleet & El-Maraghi, 2003) is another example of a region warping tracker that makes use of image features other than image intensities. In particular, this work performs image warping using phase domain filter responses. These features are computed using a multi-scale, steerable filter formulation where two scales and four orientations are used. It should be noted that these filter orientations are purely in the spatial domain and capture purely spatial structure. One of the advantages the authors argue for using their phase domain filter responses is that they are more illumination invariant than other features such as intensity.

In addition to using a novel feature set in the warping-based framework, the authors also developed a very sophisticated target model. The target model consists of stable, short-term, and noise-components. One of the long-standing issues in tracking has been how to properly update the target template (as was mentioned in Chapter 2). If a template is updated quickly, it can adapt to fast changing targets. However, if the target is temporarily occluded or a tracker slips slightly off the target, the occluder or the background will quickly be inherited as part of the target. On the other hand, if template adaption is performed slowly, the tracker may have difficulties properly representing a target whose appearance is changing relatively quickly. The tracking framework proposed in (Jepson, Fleet & El-Maraghi, 2003) deals with this challenge by explicitly modeling both fast and slow changes that a target may experience. Typically, the stable component of the target model will carry the majority of the influence when tracking is being performed. However, during partial occlusion or quick changes in target appearance, the short-term component provides the most influence. This flexible and complete method for updating the target model is a problem that does not appear to be well-studied in the tracking literature. Because of the advanced target model, a unique warping mechanism is employed whereby both the motion parameters and the target model parameters are estimated simultaneously in an EM-style framework (Dempster, Laird & Rubin, 1977).

Another system that uses image warping for the purpose of tracking is presented in (Wong & Spetsakis, 2006). The overall goal of this work is to perform motion segmentation and tracking in a video sequence. To accomplish this goal, seed regions are selected either manually or automatically. If seeding is performed automatically, an interest point operator (as discussed in Chapter 2) is used to select effective feature points for tracking. Having selected a seed point, a three stage motion estimation procedure is performed between the current and previous frame using only the region immediately surrounding the seed. The motion parameters obtained for this subregion are then used to warp the entire previous image (not just the seed region) toward the current image. In an ideal world, the parts of the image that move in a consistent manner to the seed region could be computed by taking the squared difference between the warped previous image and the current image. Hence, the target associated with the seed region could be segmented by merely applying a threshold to the squared difference image. Unfortunately, the problem is not quite this easy due to noise, illumination changes, and the fact that a seed region may be incorrectly matched to an entirely different arbitrary region that shares similar texture. As a result, more sophisticated techniques for performing segmentation in this manner are required and (Wong & Spetsakis, 2006) presents two such statistical-based techniques. The ability to segment an image sequence can be extremely beneficial for an application such as computing optical flow. With optical flow, two significant challenges are that of dealing with large motions and motion at boundaries. Since the technique in (Wong & Spetsakis, 2006) can segment the image sequence, motion boundaries are no longer an issue as optical flow can be computed for each image region in turn. Furthermore, the warping parameters computed during the seeding process provide an approximation of the true motion. Thus, optical flow will only be computing small refinements to the original motion estimates which, in general, will not involve large motions. Both of these properties are significant benefits provided by the segmentation system in (Wong & Spetsakis, 2006).

There are numerous additional warping-based trackers that are also worth mentioning in brief (Meyer & Bouthemy, 1994; Uhlin, Nordlund, Maki & Eklundh, 1995; Wixson, Eledath, Hansen, Mandelbaum & Mishra, 1998; Matthews, Ishikawa & Baker, 2004; Birchfield & Pundlik, 2008). We would certainly be remiss if we did not make note of the tracker by Meyer and Bouthemy as it is one of earliest examples of where the image warping paradigm was explicitly applied to the problem of visual tracking. In a similar manner to most of the subsequent works, the proposed system uses a coarse-to-fine framework. The authors also elect to maintain a regional shape descriptor that consists of a number of vertices on the boundary of the target. This regional descriptor is useful in detecting occlusion events if the area of the enclosed region changes much more than expected. Detection is performed with motion-based segmentation and is used for the purpose of tracker initialization and identifying the regions of interest in the current frame. The results in (Meyer & Bouthemy, 1994) are extremely impressive, especially given the date of publication. The proposed system is shown to be relatively successful for the challenging problems of partial occlusion and multi-target tracking in realistic environments. Another system that is worth mentioning in brief was proposed in (Matthews, Ishikawa & Baker, 2004). In this case, the authors address the problem of template updates. Matthews et al. show that superior template updates can be attained by selecting the optimal target region in the current frame and subsequently warping this region back toward the original template of the first frame. This warped version of the optimal target region is used as the template for the next frame. Finally, in (Birchfield & Pundlik, 2008), a tracker was developed that combined ideas from the work of Lucas and Kanade (i.e. that local regions experience the same motion) with Horn and Schunk's optical flow estimation (i.e. that scene motion varies smoothly). These constraints are combined, allowing for the joint tracking of interest points. An added benefit is that this approach allows for the tracking of edge interest points instead of just corner or well-textured interest points.

Other warping trackers have considered the incorporation of active vision components (Uhlin, Nordlund, Maki & Eklundh, 1995; Wixson, Eledath, Hansen, Mandelbaum & Mishra, 1998). In (Uhlin, Nordlund, Maki & Eklundh, 1995) a two-stage warping procedure is performed whereby the background motion is computed first. Subsequently, a parametric warp is estimated for those pixels that do not move consistently with the background. A novel aspect of this work is that disparity information is used to detect target occlusions when computing the foreground region and its corresponding motion. The motions of the foreground and background regions are then used to update the settings of an active camera. The system in (Wixson, Eledath, Hansen, Mandelbaum & Mishra, 1998) also incorporates a warping-based tracker with active camera controls. In this particular work, the fixation of ground targets from an airborne platform is achieved using a unique combination of image warping and active camera control. Because stabilization through image warping is included, a simple PI controller is sufficient for retaining fixation. Furthermore, with the camera considered in this work, only translational shifts can be rectified by altering the pan and tilt parameters. By incorporating an affine warp, transformations beyond simple translations can be resolved.

Before continuing on with kernel histogram methods, we would like to stress to the reader that there is nothing special about warping using image intensities and hopefully the literature that has been described in this section have emphasized this point. During this discussion, pixel-wise warping trackers have been described that match: intensity values to intensity values (e.g., (Black & Anandan, 1996; Enzweiler, Wildes & Herpers, 2005)); phase domain filters responses to phase domain filter responses (e.g., (Jepson, Fleet & El-Maraghi, 2003)); intensity values to learned Eigen-basis sets (e.g., (Black & Jepson, 1998; Ross, Lim & Lin, 2008)); and intensity values to an SVM classification system (e.g., (Avidan, 2001)). In theory, many other types of features could also be used within the warping framework. Features as simple as edge pixels could be used. In fact, we discussed such a work in Chapter 3 (Huttenlocher, Noh & Rucklidge, 1993), but the transformations considered in that case were only translational.

Although we did not discuss such systems in this review, it is conceivable that vector-based features could be incorporated into the warping paradigm. For instance, RGB components of a color pixel or perhaps even vectors of spatiotemporal oriented energies (Adelson & Bergen, 1985) could be used in warping trackers. An initial, naïve approach to incorporating vector-valued features into the framework would be to treat each of the components as independent images and accumulate the results across all components. Mathematically, this naïve approach could be implemented according to

$$E\left(\mathbf{a}\right) = \sum_{c}\sum_{\mathbf{x}} \rho\Big(\left(\nabla I^{c}\right)^{T}\mathbf{u}\left(\mathbf{a}\right) + I_{t}^{c}, \sigma\Big), \tag{5.15}$$

where the first summation is over all components, c, of the vector features and the superscripts on I indicate which feature set is being used. With such advancements on the immediate horizon, clearly there is much left to explore within this tracking paradigm.

5.3 Kernel Histogram Methods

The third class of region-based tracker to be discussed here is kernel histogram methods. By kernel histogram methods, we are essentially referring to trackers that use some combination of a weighting kernel and a histogram to represent the target and attain frame-to-frame object tracks. The most prominent kernel histogram tracker is the original mean shift tracker (Comaniciu, Ramesh & Meer, 2000) along with its numerous variants and extensions (Comaniciu, Ramesh & Meer, 2003; Collins, 2003; Collins & Liu, 2003; Hager, Dewan & Stewart, 2004; Birchfield & Rangarajan, 2005; Leung & Gong, 2006; Parameswaran, Ramesh & Zoghlami, 2006; Cannons & Wildes, 2007; Fan, Yang & Wu, 2007). Accordingly, the primary focus of this section will be on these works. It has been stated in this review and in the literature (Hager, Dewan & Stewart, 2004) that region-based trackers have two extremes — the blob-based trackers and the pixel-wise template trackers (rigid or warping). Kernel histogram methods can be viewed as a middle ground between these two extremes. In particular, point-wise features are accumulated across the support in a manner that typically retains more information about the target than is used in a blob tracker. However, such practices discards spatial information that is utilized in template trackers. In the mean shift formulation, kernel histogram methods arguably share more similarities to blob trackers (due to the lack of explicit shape information), but it has been shown (Hager, Dewan & Stewart, 2004) that with some modifications, kernel histogram methods can operate more akin to template trackers.

5.3.1 Standard Mean Shift Tracking

To begin, we will discuss the standard mean shift tracker that was original presented in (Comaniciu, Ramesh & Meer, 2000). At a high level, the basic components of a mean shift tracker that need to be defined are as follows: the histogram representations for the template and candidate regions; the matching metric between histograms; and the method of locating the best candidate match in the current frame. These components will now be explained, in turn.

Histogram Definition

The vast majority of mean shift trackers in the literature, including its original formulation (Comaniciu, Ramesh & Meer, 2000), use color histograms. Color histograms were previously qualitatively discussed in Chapter 2 of this review, but now the specifics of applying these histograms to a kernel-based tracker will be detailed. The idea of a basic color histogram is to consider each color in turn and count the number of pixels across the target support that are of this color. Mathematically, each bin within a color histogram can be constructed using:

$$q_u = C \sum_{i=1}^{n} \delta \left[b\left(\mathbf{x}_i^*\right) - u \right], \tag{5.16}$$

where $\mathbf{x}_i^* = (x^*, y^*)$ is a single target pixel at some temporal instant, i ranges so that $\mathbf{x}_i^*$ covers the template support, C is a constant that can be included to ensure that the histogram bins sum to one, u is a particular histogram bin, and δ is the Dirac delta function. The function b can be thought of as a very simple function that maps a pixel color value (the input) to its corresponding histogram bin (the output). Since each bin corresponds to a particular color, Eq. 5.16 will only increment the count for a bin when a pixel of the correct color is encountered.

In the mean shift tracking framework, the basic histogram of Eq. 5.16 is slightly modified to include a kernel weighting function. The rationale behind the kernel function is to weight pixels depending on their spatial location within the tracking window. Heuristically, pixels at the center of the window are very likely to truly belong to the target and are correspondingly weighted highly. On the other hand, since the tracking window does not perfectly adhere to the outline of the target, it is believed that pixels near the border of the tracking window have a higher chance of being part of the background. Due to this decreased certainty, pixels near the border of the tracking window receive lower weights. Equation 5.16 can be adjusted to include the kernel weighting function as follows

$$\hat{q}_u = C \sum_{i=1}^{n} k\left(||\mathbf{x}_i^*||^2\right) \delta\left[b\left(\mathbf{x}_i^*\right) - u\right], \tag{5.17}$$

where k is the profile of the tracking kernel. Note that under our notational convention, when referring to kernels, uppercase indicates the kernel itself while lowercase refers to the kernel profile. Following the notation of (Comaniciu, Ramesh & Meer, 2000), Eq. 5.17 is the template histogram, originally defined in the first frame of the video (although it may be updated as tracking progresses). This process of weighting by a kernel to compute an approximate probability distribution is commonly referred to as the "kernel density estimation".

When tracking a target between frames, it may be necessary to evaluate several target candidates before a final, optimal target position is found for the current frame. The histograms for the target candidates are evaluated using

$$\hat{p}_u = C \sum_{i=1}^{n} k\left(\left|\left|\frac{\mathbf{y} - \mathbf{x}_i^*}{h}\right|\right|^2\right) \delta\left[b\left(\mathbf{x}_i^*\right) - u\right], \tag{5.18}$$

where $\mathbf{y}$ is the center of the target candidate's tracking window, h is the bandwidth of the tracking kernel and i ranges so that $\mathbf{x}_i^*$ covers the candidate support. The kernel bandwidth allows for scale changes of the target throughout the video sequence.

The final component of the histograms that must be explained is the kernel. The Epanechnikov kernel, has been shown to be effective (Birchfield & Rangarajan, 2005; Comaniciu, Ramesh & Meer, 2000; Hager, Dewan & Stewart, 2004) and appears to be the most commonly used kernel in the application of the mean shift algorithm to computer vision. However, it should be noted that other types of kernels, such as Gaussian-based kernels, (Collins, 2003) have been considered as well. One restriction imposed upon the kernel is that it must be convex and monotonically decreasing (Comaniciu, Ramesh & Meer, 2000). Since the Epanechnikov kernel is the most popular in the literature, it will be the focus of this paper. This kernel, which is graphically shown in Figure 5.5, is mathematically defined as

$$K_E\left(x\right) = \begin{cases} \frac{1}{2}{c_d}^{-1}\left(d+2\right)\left(1 - ||\mathbf{x}||^2\right), & \text{if } \ ||\mathbf{x}|| < 1 \\ 0, & \text{otherwise} \end{cases}$$

where c_d is the volume of a unit d-dimensional sphere.

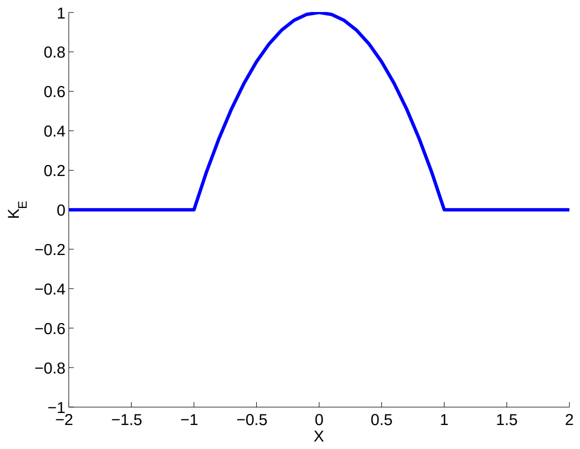


Figure 5.5: The Epanechnikov kernel commonly used in mean shift tracking.

Matching Method

Now that we have defined histogram models for the template and candidates, a method is needed for comparing these models. Although a variety of methods could be used for comparing histograms (e.g., Earth mover's distance (Rubner, Tomasi & Guibas, 2000), Euclidean distance), following (Comaniciu, Ramesh & Meer, 2000), the Bhattacharyya coefficient is used. The discrete, sample estimate of the Bhattacharyya coefficient is computed using

$$\hat{\rho}(\mathbf{y}) \equiv \hat{\rho}(\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}) = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y})\, \hat{q}_u} \tag{5.19}$$

where u varies over all histogram bins. Due to the definition of the Bhattacharyya coefficient, in order to minimize the distance between two histograms, Eq. 5.19 must be maximized with respect to target position, $\mathbf{y}$. With a similarity measure in place, the final component of the mean shift tracker is that of determining an efficient mechanism for searching for the optimal position in the current frame.

Target Position Estimation

Similar to simple gradient-ascent methods, the mean shift algorithm seeks the modes of a distribution by using an estimate of the function's gradient. As shown in (Fukunaga & Hostetler, 1975), an estimate of the density gradient can be defined in terms of a so-called "mean shift vector". The modes of a distribution can be located by iteratively computing this mean shift vector and translating the center of the kernel to the specified location. This iterative procedure can be employed to maximize an objective function, as is done in the mean shift tracker of (Comaniciu, Ramesh &

Meer, 2000).

The Bhattacharyya coefficient can be maximized via mean shift iterations (Fukunaga & Hostetler, 1975; Comaniciu, Ramesh & Meer, 2000) by first taking the linear approximation of Eq. 5.19 and substituting the definition of the target candidate histogram, Eq. 5.18, into the resulting linear equation. Note that the candidate histogram equation must be substituted, as it is the only equation that is a function of the target candidate position $\mathbf{y}$, over which optimization is being performed. The resulting mathematical expression is

$$\rho\left(\hat{\mathbf{p}}\left(\mathbf{y}\right),\hat{\mathbf{q}}\right) \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{\hat{p}_u\left(\hat{\mathbf{y}}_0\right)\hat{q}_u} + \frac{C_h}{2}\sum_{i=1}^{n_h} w_i k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i^*}{h}\right\|^2\right), \tag{5.20}$$

where

$$w_i = \sum_{u=1}^{m}\delta\left[b\left(\mathbf{x}_i\right)-u\right]\sqrt{\frac{\hat{q}_u}{\hat{p}_u\left(\hat{\mathbf{y}}_0\right)}}. \tag{5.21}$$

The second term in Eq. 5.20 is in a form similar to that of a standard kernel density estimate, the only significant difference being the inclusion of the weighting terms, w_i. As shown in (Fukunaga & Hostetler, 1975; Comaniciu, Ramesh & Meer, 2000), the local optimum of a kernel density estimate can found using the mean shift vector. This implies that the linearized Bhattacharyya coefficient equation can be maximized by using mean shift iterations. The specific mean shift vector that can be used to perform the desired maximization is

$$\hat{\mathbf{y}}_1 = \left[\frac{\sum_{i=1}^{n_h}\mathbf{x}_i^* w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0-\mathbf{x}_i^*}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0-\mathbf{x}_i^*}{h}\right\|^2\right)}\right] \tag{5.22}$$

where $g\left(x\right) = k'\left(x\right)$ is the derivative with respect to x of tracking kernel profile, k, and $\hat{\mathbf{y}}_0$ is the current position of the target.

Thus, the position of the target in the current frame is estimated as follows. Starting from the target's position in the previous frame, iterations are performed whereby the mean shift vector is computed and the target candidate is moved to the position indicated by the mean shift vector. These steps are repeated until convergence has been reached or a fixed number of iterations have been executed.

Scale and Template Updates

In the original mean shift formulation, scale and template updates were handled in a very simplistic fashion. The target template was updated to be a weighted combination between the best candidate in the current frame and the previous template. The weights were selected based on the Bhattacharyya coefficient (confident matches received less aggressive updates). Scale updates were performed by completing mean shift optimization using a range of different target scales. The scale that provided the best match was used as the final solution and for updating the template. Clearly, the scale and template updates proposed in (Comaniciu, Ramesh & Meer, 2000) were not the focus of the work.

Performance Results

The qualitative performance of the original mean shift tracker, demonstrated in (Comaniciu, Ramesh & Meer, 2000), is quite good. Three challenging sequences are shown that focus on the problems of tracking American football players and persons on a subway platform. The football sequence presents a number of tracking challenges including partial occlusions, multiple distractors with similar color distributions, and camera motion causing significant levels of motion blur. The subway sequences are also challenging, involving partial occlusions, significant clutter, scale changes, and numerous foreground objects with similar appearances. For all demoed sequences, the system performs very well. However, it should be noted that in one of the subway sequences, a person with a white jacket is tracked. This woman is essentially the only person in the entire scene with white clothing. Furthermore, although tracking never fails on the football sequence, the tracking window is not always well-centered on the target. Finally, since color features are used, it is likely that the system will not be overly robust to changes in illumination.

5.3.2 Extensions to the Mean Shift Tracker

In the years following its initial inception (Comaniciu, Ramesh & Meer, 2000), a number of trackers were proposed that extended the basic mean shift tracker. The majority of these extensions were more incremental than radically changing the mean shift framework. Some works, such as Comaniciu et al.'s extended paper (Comaniciu, Ramesh & Meer, 2003), introduced predictive Kalman filters to smooth the target tracks. Depending on the application domain, the inclusion of a Kalman filter could be beneficial or detrimental. For example, if the motion is expected to be smooth and not experience any sudden large changes (e.g., large passenger airplane traveling across the sky) than a Kalman filter should be beneficial in smoothing the target track. On the other hand, if the target is expected to make sudden changes in direction or otherwise move erratically (e.g., ping pong ball during a table tennis match), Kalman filtering could be detrimental, causing overshoot and possible loss of the target's position (Nummiaro, Koller-Meier & Gool, 2002).

In addition to the inclusion of predictive components, throughout the extensions, a number of themes can be observed. Some works (Birchfield & Rangarajan, 2005; Hager, Dewan & Stewart, 2004; Leung & Gong, 2006; Parameswaran, Ramesh & Zoghlami, 2006; Fan, Yang & Wu, 2007) have attempted to include additional spatial information. Meanwhile, other researchers have considered increasing the speed of the algorithm (Hager, Dewan & Stewart, 2004; Leung & Gong, 2006). As the scale update solution proposed in (Comaniciu, Ramesh & Meer, 2000) was far from elegant, techniques have been proposed for more effectively handling changes in target scale (Collins, 2003; Hager, Dewan & Stewart, 2004). Finally, the matter of feature spaces and automatically selecting the most discriminative features for tracking has been investigated (Collins & Liu, 2003; Leung & Gong, 2006; Cannons & Wildes, 2007; Avidan, 2007; Yin, Porikli & Collins, 2008; Parag, Porikli & Elgammal, 2008). These various classes of extensions will now be described in brief.

Including Spatial Information

The fact that spatial information is essentially discarded when constructing the histograms that are used in mean shift tracking could be considered an obvious point of weakness. On the other hand, this lack of spatial information can be advantageous, as it allows targets to undergo planar rotation and non-rigid transformations without changing the resulting histogram. Template trackers are seemingly at the other extreme as they represent the target in a pixel-wise fashion, and thus retain all spatial information. Researchers have noted this lack of spatial information in the histogram model and have considered methods to incorporate various amounts of spatial information into the mean shift framework (Birchfield & Rangarajan, 2005; Hager, Dewan & Stewart, 2004; Leung & Gong, 2006; Parameswaran, Ramesh & Zoghlami, 2006; Fan, Yang & Wu, 2007).

One of the first works that considered this problem was that of (Birchfield & Rangarajan, 2005). In this case, the authors create a so-called "spatiogram" instead of a standard histogram. Essentially, a spatiogram is a histogram where each bin is weighted by the spatial mean and covariance of its contributing pixels. Hence, a spatiogram bin is now described by a set of values $\{q_u, \mu_u, \Sigma_u\}$ which correspond to the count, mean, and covariance of the pixels contributing to that bin, respectively. Due to the inclusion of this additional information, a modified matching metric must be derived. In (Birchfield & Rangarajan, 2005), the metric used is

$$\rho_{mod}\left(q_u, p_u\right) = \sum_{u=1}^{m} \phi_u \rho\left(q_u, p_u\right) \tag{5.23}$$

where

$$\phi_u = \eta \exp\left(-\frac{1}{2}\left(\mu_u^p - \mu_u^q\right)\hat{\Sigma}_u^{-1}\left(\mu_u^p - \mu_u^q\right)\right). \tag{5.24}$$

Note that $\rho\left(q_u, p_u\right)$ is just a standard method for comparing histogram bins, such as the Bhattacharyya coefficient as was used in (Comaniciu, Ramesh & Meer, 2000). Furthermore, μ_u^p and μ_u^q are the mean of the candidate and target pixels that contribute to bin u, η is a Gaussian normalization constant, and $\hat{\Sigma}_u^{-1}$ is the combined covariance for the two histograms.

The authors substitute the spatiograms along with this new matching criterion into the mean shift framework. The derivation proceeds in a similar manner to that presented in (Comaniciu, Ramesh & Meer, 2000). The end result is a mean shift vector where each pixel votes for the magnitude and direction of the shift. This is in contrast to the standard mean shift method where each pixel is implicitly voting for a shift in the direction of the pixel itself. The authors demonstrate that in head tracking situations, the use of spatiograms provides improved performance over the standard mean shift algorithm.

In (Leung & Gong, 2006) the authors attempted to include spatial information by performing normalized cuts (Shi & Malik, 2000) on the initial frame. The authors' argument is that the kernel weighting assumption (pixels near the center are more reliable) is not always true nor accurate. Normalized cuts provides a more precise outline of the target in the first frame. Throughout the rest of the sequence, the authors only use pixels within this target region (represented as a 2D histogram) to construct candidate models. Although a sequence is shown where this approach outperforms that of standard mean shift, their method and implementation appear to have severe limitations as the object mask does not seem to be updated.

Finally, in (Hager, Dewan & Stewart, 2004) an interesting approach is presented where multiple kernels are used for tracking a target. The authors of (Hager, Dewan & Stewart, 2004) argue that kernel histogram trackers can be customized to be closer to either extreme of the region-based trackers (blob trackers or pixel-wise template trackers) by using multiple kernels. Specifically, a connection is initially drawn between template trackers and mean shift trackers. To that end, an SSD error function, known as the Matusita metric (Matusita, 1955), is presented. The Matusita metric can be shown to share a close relationship with the Bhattacharyya coefficient through the following derivation

$$\begin{aligned} O(\mathbf{y}) &= \left|\left|\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{y})}\right|\right|^2 \\ &= \left[\left(\sqrt{q_1} - \sqrt{p_1(\mathbf{y})}\right)^2 + \left(\sqrt{q_2} - \sqrt{p_2(\mathbf{y})}\right)^2 + \dots \right. \\ &\qquad \left. \left(\sqrt{q_m} - \sqrt{p_m(\mathbf{y})}\right)^2\right] \\ &= \left(q_1 - 2\sqrt{q_1 p_1(\mathbf{y})} + p_1(\mathbf{y})\right) + \\ &\qquad \left(q_2 - 2\sqrt{q_2 p_2(\mathbf{y})} + p_2(\mathbf{y})\right) + \\ &\qquad \dots \left(q_m - m\sqrt{q_m p_m(\mathbf{y})} + p_m(\mathbf{y})\right) \\ &= \sum_i^m q_i + \sum_i^m p_i(\mathbf{y}) - 2\sum_i^m \sqrt{q_m p_m(\mathbf{y})} \\ &= 2 - 2\rho(\mathbf{q}, \mathbf{p}(\mathbf{y})), \end{aligned} \tag{5.25}$$

where $\mathbf{p}$ is the target and, $\mathbf{q}$ is the template, and m are the number of histogram bins. The last step in the derivation is performed with the knowledge that the kernel histogram bins sum to unity (since they represent discrete probability distributions) and using Eq. 5.19. Thus, the minima of Eq. 5.25 coincides with the maxima of the Bhattacharyya coefficient. Hence, if a method were available for finding the local optima of the SSD-based Matusita metric, then given the same starting point, the mean shift algorithm and this unknown optimization algorithm should converge to the same solution. With that goal in mind, the authors derive a Newton-style iterative procedure for optimizing Eq. 5.25. Thus, the best local match between a template kernel modulated histogram and a number of candidates can be found using a template-matching inspired SSD metric.

The authors argue this is a viable alternative to performing optimization with mean shift iterations for numerous reasons. One advantage of this alternative optimization is that it is easier to incorporate parametric motion models that deform the tracking window. Typically, in region trackers, these parametric models are reserved for use in warping-based trackers. However, under this new optimization framework, deformations to the tracking window such as shearing and scaling can be incorporated. Another advantage of the proposed SSD-based optimization procedure is that it allows for the tracking with multiple kernels. Multiple kernels can help to resolve ambiguities and further constrain the target's spatial structure. Figure 5.6 provides an example of when single kernel-based trackers result in ambiguous solutions. It should be noted, however, that the inclusion of multiple kernels requires more attention from the designer to ensure that appropriate kernels are used and that the

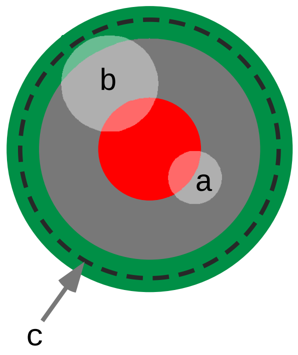


Figure 5.6: The spatial ambiguity problem when using a single tracking kernel. Kernels a and b are ambiguous and can move anywhere around their corresponding circles. Kernel c has a unique, optimal position.

tracker is sensitive to the desired directions of motion.

Recently, the work of Hager et al. was extended in (Fan, Yang & Wu, 2007). In this work, a theoretical analysis is performed that shows the conditions for motion observability with a single kernel. Furthermore, a tracking architecture is proposed where multiple kernels operate together in a collaborative fashion. Under their proposed framework, it would be possible to simply use pair-wise collaborations. Instead, the authors elect to incorporate information regarding the global relationships amongst the kernels using a principal component analysis-based method. In another recent system, (Parameswaran, Ramesh & Zoghlami, 2006), human tracking was achieved by combining the responses of multiple radially symmetric kernels to create an overall kernel function. A novel aspect of this system is that each individual kernel's bandwidth can be learned and varied individually. Such flexibility is useful when tracking articulated targets (e.g., humans) where larger bandwidths are needed for areas such as the arms and legs and smaller bandwidths can be used for more stable regions such as the torso.

As evidenced from some of the most recent works, it appears that a popular method of incorporating additional spatial information into the kernel histogram tracking paradigm is by making use of multiple kernels. As research in the area of multiple kernel tracking is relatively new, further studies should be conducted to more concretely ascertain its true utility.

Improving Tracker Efficiency

In addition to considering the inclusion of spatial information, the papers (Hager, Dewan & Stewart, 2004) and (Leung & Gong, 2006) also contributed to the goal of making the mean shift algorithm more efficient. In (Leung & Gong, 2006) uniform random sampling was used to select a subset of tracking window pixels when constructing the target and candidate histograms. Surprisingly, for simple color distributions, just 5 samples are required to achieve accurate tracking. Typical targets, however, require approximately 15 samples. Nonetheless, targets can easily have a support region that extends into thousands of pixels, making the solution proposed in (Leung & Gong, 2006) attractive.

An additional advantage of the alternative optimization procedure proposed in (Hager, Dewan & Stewart, 2004) is that it converges more quickly than mean shift iterations. The Newton-style optimization approach attempts to move to the local optima in one step. The authors qualitatively and quantitatively observe the reduced number of iterations when using the Newton-style optimization. Although it is expected that the overall speed improvement is substantial, no comparison is mentioned regarding the number of computations that are required per iteration for the two approaches.

Updating Target Scale

It might not be immediately obvious why scale update is of significant concern when developing a tracker. The designer might falsely think that a tracking window that is much larger than the target is acceptable because it encompass the entire target support, and corresponding, all information about the target. On the other hand, one might think that a tracking window that is smaller than the target is also acceptable because it can still lock onto some portion of the target. In fact, neither of these solutions is ideal for different reasons.

Tracking windows that are larger than the target itself are not desirable as the model then encompasses many background pixels in addition to the true target pixels. If the number of background pixels is substantial, the developed model might capture as much (or more) information about the background than the target itself! At the other end of the spectrum are tracking windows that are smaller than the target. This situation is also not ideal as there may be a number of candidate solutions that are essentially equally good. As a result, due to noise, small illumination changes, etc. the tracking window may move between these essentially identical optima within the solution space, providing a very erratic target track. The target window will typically be neither consistently placed on the target nor well-centered.

The original mean shift algorithm implemented an extremely crude mechanism for updating the size of the target window. Specifically, mean shift optimization was performed using three different sizes of target windows and the scale that provided the best optimum was kept. Other researchers have taken note of the brute-force nature of this scheme and have proposed more elegant methods for adaptively changing the target scale. We have already mentioned the first more eloquent technique that can be used for changing scales. In particular, in (Hager, Dewan & Stewart, 2004) target deformations can be easily incorporated into the optimization process. Scale changes are included in the set of allowable parametric transformations.

Another work attempted to solve the scale selection problem by using a scale space-based approach for adapting target size (Collins, 2003). Previous work by Lindeberg (Lindeberg, 1998) on feature point detection inspired this mean shift tracking work by Collins. Conceptually, the tracking procedure operates on a scale space that is generated by convolving a modified version of the image with a multi-scale difference of Gaussian filter bank in the spatial domain and subsequently filtering with an Epanechnikov kernel in the scale dimension. The modes of this scale space correspond to blobs at multiple scales within the image. The goal of the tracker in (Collins, 2003) is to track these blobs throughout scale space (i.e. both in the spatial domain and in the scale domain). A two-phase mean shift procedure is performed where the scale is initially held constant and mean shift iterations are performed to converge to the nearest local spatial mode. Subsequently, the spatial position is held constant and mean shift iterations are performed across the scale axis. These two stages are iterated until both scale and spatial position parameters have stabilized. This scale selection technique is experimentally shown to be vastly superior to that seen in (Comaniciu, Ramesh & Meer, 2000).

Feature Selection and Feature Advancement

One common theme that arises across all tracking systems (but is arguably most prevalent for region trackers) is that of selecting features for tracking. Within the kernel histogram tracker domain, the types of features investigated have been extremely limited. Most works have focused on the use of color histograms. Oftentimes different color spaces and different bin quantizations are used, but these are minor changes. In this section, we will discuss methods of automatically adapting and selecting features in each frame to attain the best tracking results (Spengler & Schiele, 2003; Collins & Liu, 2003; Leung & Gong, 2006; Avidan, 2007; Yin, Porikli & Collins, 2008). Furthermore, we will describe a mean shift tracker that incorporates an extremely novel, rich feature set (Cannons & Wildes, 2007).

Most trackers are designed to locate the position of the target in video frames using a rigid set of features. It is naïve to assume that the particular features chosen by the designer will be effective for all tracking scenarios. In some situations (e.g., cluttered background) it may be more appropriate to use color-based features. In other occasions (e.g., similar colored foreground and background, illumination changes), performance may be increased by using gradient information. An ideal tracker would have a method of automatically detecting when the use of each type of feature is appropriate and dynamically switching between the various available features. The first steps toward this goal were presented in (Collins & Liu, 2003).

In this work by Collins and Liu, the “optimal” feature is defined as that which maximally discriminates between the foreground and background portions of image.

Using color as an example, if the color black is common both on the target and in the background, it is not an effective feature (if we are presented with a random black pixel, we could not say with certainty if it belonged to the foreground or background). On the other hand, if the target contains numerous red pixels and the background never contains red pixels, red would be considered to be a good feature. In a similar manner, if the background contained a large amount of green and the foreground does not contain any green pixels, green would also be considered to be a good feature.

Although their work could be applied to any pool of features (e.g., color, texture, shape, motion), in (Collins & Liu, 2003), the authors elect to use various weighted combinations of the R, G, and B components of a pixel to represent different features. Specifically, a feature is defined as:

$$F = \{w_1R + w_2G + w_3B | w_* \in [-2, -1, 0, 1, 2]\} \tag{5.26}$$

where w_* are the integer weights. After removing all repeated features (when the weight vector is a multiple of a previous weight vector) and the weight vector consisting of all zeros, 49 possible features remain.

The system in (Collins & Liu, 2003) as well as many following works (e.g., (Leung & Gong, 2006; Avidan, 2007; Yin, Porikli & Collins, 2008)) take a different approach to tracking to the standard mean shift trackers. These systems essentially generate pixel-wise classification decisions capturing the likelihood that a given pixel corresponds to the target. Combining the pixel-wise classifications into a likelihood image (or confidence image), one can essentially perform mean shift optimization on: (1) a single likelihood image; (2) a set of likelihood images (e.g., (Collins & Liu, 2003)); or (3) a combined likelihood map (e.g., (Avidan, 2007; Yin, Porikli & Collins, 2008)). With this conceptual framework in mind, we will now describe the specific method employed in (Collins & Liu, 2003), whereby mean shift optimization is performed over a set of likelihood maps.

Having defined a method of computing the optimal feature in the current frame (Eq. 5.26), the on-line feature selection tracker of (Collins & Liu, 2003) performs the following steps:

1. Select a feature
2. Using this feature, construct histograms for the foreground and background
3. Compute a log likelihood for each histogram bin
4. Compute the separability between the foreground and background with respect to the current feature
5. Repeat steps 1-4 for all features
6. Select the top N features

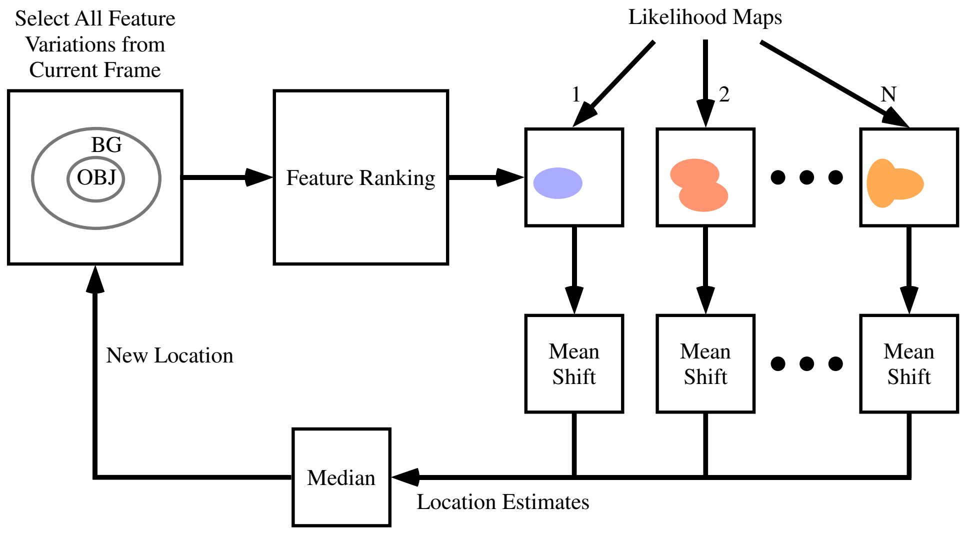


Figure 5.7: Block diagram of the on-line feature selection tracker proposed in (Collins & Liu, 2003).

7. Load the next video frame

8. Create N likelihood images. A likelihood image for a particular feature is constructed by replacing each image pixel value with the corresponding log likelihood histogram value.

9. Perform N instances of mean shift tracking on the N likelihood images.

10. Combine the mean shift tracking results

11. Repeat the above steps for all video frames

Figure 5.7 shows a pictorial block diagram of the system processing steps. It should be noted that in Step 3, the log likelihood values are computed according to the equation:

$$L(i) = \log \frac{\max\{p(i), \epsilon\}}{\max\{q(i), \epsilon\}} \tag{5.27}$$

where $\mathbf{p}$ is the normalized object histogram, $\mathbf{q}$ is the normalized background histogram, and ϵ is a small offset included to avoid numerical errors. Furthermore, the separability between the foreground and background with respect to the current feature is measured via the variance ratio of $L(i)$.

More recently, the authors of (Leung & Gong, 2006) attempted to extend the feature selection scheme proposed by Collins and Liu. In (Leung & Gong, 2006) the use of real-valued weights is proposed so that a steepest decent technique can be used to find the optimal weight vector. Although superior performance is reported when compared to the standard mean shift algorithm, no comparison is made between the systems in (Leung & Gong, 2006) and (Collins & Liu, 2003). It would be interesting to know the gains of incorporating real valued weight and thus, a much larger pool of features. The authors of (Leung & Gong, 2006) do note that the steepest decent algorithm requires 0.4 seconds in Matlab, which is presumably more computationally intense than the approach in (Collins & Liu, 2003). Certainly, a down-fall of including an online feature selection component (especially a complex one) is that it will increase computation time.

Other systems have been proposed that follow a very similar approach to that of Collins et al. (Spengler & Schiele, 2003; Avidan, 2007; Yin, Porikli & Collins, 2008; Parag, Porikli & Elgammal, 2008). The system in (Yin, Porikli & Collins, 2008) is essentially an extension of the work (Collins & Liu, 2003). Once again, likelihood maps are generated for a variety of different modalities. Example modalities include histograms of intensities, histograms of oriented gradients, and template correlations. The primary difference between this work and that of the system in (Collins & Liu, 2003) is that the individual likelihood maps are fused into a combined likelihood map prior to performing the mean shift algorithm. This combined likelihood map is created by taking the weighted sum of the individual likelihood maps, where the weights are set based on a modality's ability to distinguish between foreground and background regions in the previous frame. Experimental results indicate that the proposed algorithm outperforms the algorithm in (Collins & Liu, 2003).

In a similar spirit to the abovementioned works, Avidan derived a system that generates a combined confidence map using a number of feature modalities. However, rather than explicitly generating a map for each modality, the combined map is computed using feature vectors and an ensemble classification scheme. The idea in (Avidan, 2007) is to perform pixel-wise classification using a strong classifier that is composed of a number of weak classifiers. The feature vector used to describe each pixel can be selected by the designer; however, Avidan suggests the use of an eight-bin HOG (computed in a 5×5 neighborhood) supplemented with the pixel's RGB values. Upon computing foreground confidence scores using the ensemble classifier, the mean shift algorithm is used to locate the local optimum. To ensure that the ensemble classifier adapts to the changing target properties, the weak classifiers are evaluated at the conclusion of processing for each frame. The k best weak classifiers are retained for future frames. Furthermore, $q - k$ new weak classifiers are incorporated (q is the total number of weak classifiers) that are trained using information in the current frame. A number of excellent results are illustrating in the work for face, body, and vehicle tracking scenarios. Results for simple occlusion situations (with the inclusion of a predictive filter) are also demonstrated. Interestingly enough, the later work in (Yin, Porikli & Collins, 2008) reports universally superior performance to Avidan's system. However, no definitive conclusions can be drawn based on this result as the implementation of Avidan's ensemble tracker in (Yin, Porikli & Collins, 2008) is not identical to that in Avidan's original work.

Another recent extension to Avidan's ensemble tracker was proposed in (Parag, Porikli & Elgammal, 2008). In this case, rather than simply eliminating the worst performing weak classifiers, the weak classifiers are continuously updated using the most recent data from the current frame and an online boosting algorithm. The proposed system avoids the problem of selecting the number of weak classifiers that should be replaced in each frame, which is essentially just an arbitrary parameter in Avidan's ensemble tracker. A qualitative analysis shows the proposed system outperforms the ensemble tracker.

The idea of selecting and combining a multitude of different features when performing mean shift tracking is certainly one research avenue that should be explored in more detail. The works of (Collins & Liu, 2003; Leung & Gong, 2006; Avidan, 2007; Yin, Porikli & Collins, 2008; Parag, Porikli & Elgammal, 2008) are certainly an excellent starting point and illustrate that there are gains to be had by using multiple modalities. Presently, there are some shortcomings of the abovementioned systems. These systems all essentially follow the mentality of performing binary classification decisions. With this approach, a pixel is evaluated as to whether it belongs to the foreground or the background. An immediate question arises: "how can multi-target tracking be performed in this framework?" It appears this question has yet to be fully studied, but it certainly warrants additional research. An additional problem with these mean shift trackers is that there appears to be little rationale for including each modality. The authors basically select a number of modalities that are very simple and can be computed efficiently. Given the redundant and complementary nature of the feature set, the hope is that at least one modality will be sufficiently discriminative for successful tracking at each frame. As the authors have shown, this approach can be successful, but there is something less than satisfactory with such a "try everything and hope something works" mentality.

As opposed to actually modifying the feature set that is used during tracking, a different approach considered in (Chen & Yang, 2007) is to weight target areas, based on the confidence that they are being tracked properly. Conceptually, this idea is somewhat similar to the normalized cuts-based approach by Leung et al. discussed in the previous subsection; however, in (Chen & Yang, 2007) weightings are determined based on feature discriminability. Specifically, in (Chen & Yang, 2007), the proposed approach downweights target pixels that share a similar color to pixels found in the surrounding background. Furthermore, computations are performed to determine the probability that each target pixel is being occluded. Target pixels that are believed to be occluded are also downweighted. Mean shift tracking is subsequently performed on a confidence weighted image.

Another alternative to combining a number of disparate features (Spengler &

Schiele, 2003; Collins & Liu, 2003; Leung & Gong, 2006; Avidan, 2007; Yin, Porikli & Collins, 2008) or performing additional confidence weighting (Chen & Yang, 2007) would be to derive a single feature set, derived under a common framework, that is more descriptive than individual features previously considered in tracking. This latter approach is explored in (Cannons & Wildes, 2007). It is somewhat surprisingly that, to our knowledge, HOG did not appear in the mean shift tracking literature prior to the works of (Avidan, 2007; Yin, Porikli & Collins, 2008). Such a representation captures the local spatial orientation structure of the target. In (Cannons & Wildes, 2007), the idea is to define a HOG-style feature set that not only captures the spatial orientations of a target (like HOGs), but also captures the dynamic aspects of the targets as well. Specifically, in this work, a very unique feature set, one derived from spatiotemporal oriented energies (described in detail in Chapter 2), is used within the mean shift framework. The argument for these features is that they are much more powerful and rich than color histograms as evidenced by a number of realistic situations.

The authors argue that spatiotemporal oriented energies can provide an extremely rich description of a target. In particular, energies are computed across 10 orientations (the minimum number of orientations required to provide a spanning basis set for the highest order filters they employ) in three dimensional (x, y, t) image space and three scales. These energies capture both static and dynamic properties of a target depending on their orientation. In particular, energies that are derived from filters that remain within the image plane capture purely spatial structure; whereas, those that are computed from filters that extend into the temporal dimension capture dynamic information. Multiscale processing is also important, as coarse scales capture gross spatial patterns and overall target motion while finer scales capture detailed spatial patterns and motions of individual parts (e.g., limbs). By encompassing both spatial and temporal target characteristics in an integrated fashion, tracking is supported in the presence of significant clutter. Furthermore, spatiotemporal oriented energies are robust to illumination variations due to the broadness of the filters and the normalization stage used during their computation (Eq. 2.4).

To integrate oriented energies into the mean shift framework, the authors derive a so-called "oriented energy histogram". An oriented energy histogram contains a bin for the energy of the target at each scale/orientation combination. Thus, the entire histogram shows the energy of the target across all orientations and scales. Since the authors compute oriented energies at three scales and 10 orientations, their histograms are 30-dimensional. To create such a histogram, the energy of each "channel" (i.e. each scale/orientation) is summed across the target support. The summed energy for each channel becomes a particular histogram bin. For illustrative purposes, a portion of the same figure as seen in Chapter 2, containing sample spatiotemporal oriented energies is reproduced in Fig. 5.8. Furthermore, this figure pictorially illustrates the process of constructing an oriented energy histogram from the energy images.

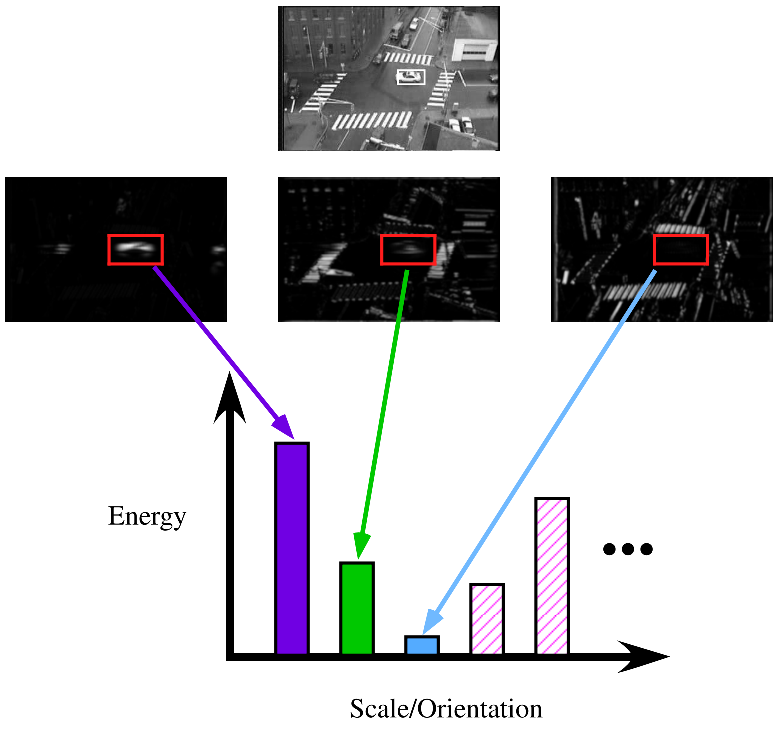


Figure 5.8: Frame from a MERL traffic video sequence with select corresponding energy channels (top two rows). The bottom row shows the construction of an oriented energy histogram by summing across the target support for each energy channel.

The introduction of the new type of histogram in (Cannons & Wildes, 2007) necessitated a slight change in derivation for the mean shift vector. Specifically, the weights used when computing the mean shift vector are updated to include the energy of each pixel

$$w_i = \sum_{u=1}^{m} \hat{E}\left(\mathbf{x}_i^*; \phi_u\right) \sqrt{\frac{\hat{q}_u}{\hat{p}_u\left(\hat{\mathbf{y}}_0\right)}}, \tag{5.28}$$

where $\hat{E}\left(\mathbf{x}_i^*; \phi_u\right)$ is the energy of a particular target pixel $\mathbf{x}_i^*$ for energy computed at a particular scale/orientation combination, ϕ_u.

In this work (Cannons & Wildes, 2007), the authors demonstrated the benefits of their proposed feature set over color histograms in several experimental situations: multiple similar colored targets; drastic illumination changes; small targets; and background clutter. The problem of finding more descriptive feature sets is common to all areas of tracking and the work of Cannons and Wildes is a reasonable first step as it provides quite a rich description including both dynamic and static target properties. Perhaps the promising results obtained in this work indicate that additional research efforts should be devoted to experimenting with more novel feature sets for tracking.

In summary, kernel histogram methods such as the mean shift tracker are currently very popular due to their speed, simplicity, as well as their robustness to clutter and occlusions. It appears that contributions have been made toward the scale and template update problem as well as that of feature selection. Nonetheless, it appears these areas could certainly still use further research.

5.4 Other Region Trackers

In an ideal world, for reviews such as this one, we would be able to perfectly place all region trackers into a small set of sub-categories. This, in general, cannot be done. However, perhaps it is beneficial for the field overall if perfect categorization cannot be achieved because it shows that innovation and a willingness to explore radically different approaches exists. This section will briefly mention a number of works that do not quite fit in any of the categories we defined previously. Many of these works make use of Kalman filters (Mittal & Davis, 2003; Kang, Cohen & Medioni, 2003; Zhao & Nevatia, 2004) or particle filters (Wu & Huang, 2001; Isard & MacCormick, 2001; Nummiaro, Koller-Meier & Gool, 2002; Okuma, Taleghani, Freitas, Little & Lowe, 2004) to assist in tracking. Others make use of 3D information or multiple cameras.

To begin, the system by Fieguth and Terzopoulos (Fieguth & Terzopoulos, 1997) was one of the earlier papers to make use of color information and was published in the same year as Wren's PFinder system (Wren, Azarbayejani, Darrell & Pentland, 1997). The proposed system employed very basic tracking techniques, but was clearly designed for speed. This tracker represented the target with N rigidly connected regions. For instance, if a person's head and shoulders were being tracked, six regions might be defined that cover the person's forehead, cheeks/eyes, nose, mouth, and shirt. Each region is described simply by its mean color value and matching is performed by taking the ratios of the mean RGB values for the template and candidates. Searching for the best match is performed through a nonlinear velocity prediction mechanism and a local search around the predicted position. Another interesting aspect of this work is that occlusion events are explicitly modeled and dealt with. In particular, multiple hypothesis are created that correspond to the different combinations of the defined regions being contiguously occluded. The hypotheses that are sufficiently probable are propagated to the next frame where there should be more information that can be used to resolve the ambiguity. Although this work is simplistic by today's standards (the relationships between regions is rigid and must be defined by the designer, no scale updating, no template updating) at the time it was impressive, especially considering that it ran at frame-rate.

Part of the reason why some trackers do not properly fit into any of the above predefined categories is that they use three dimensional information or information from multiple cameras. These experimental conditions do not immediately disqualify a tracker from falling into one of the previous categories. However, as will be described, typically this extra information requires different techniques and additional processing. One such example is the human tracker presented in (Zhao & Nevatia, 2004) that combines blob analysis, warping, and Kalman filtering. The blob analysis component of the system is used during human detection. Specifically, blobs are analyzed to determine the location of a person's head and subsequently a 3D ellipsoid model is initialized using the head position as well as knowledge of the camera model and the ground plane.

The authors argue that pure blob-trackers are limited in tracking accuracy and as a result, make use of other tools for the actual tracking. In particular, Kalman filtering predicts the position of the target in subsequent frames and a local search obtains the optimal measurement in the vicinity. In addition to optimizing translational parameters during the local search, warps on the image ellipse involving rotation and scaling are also permitted. Thus, this tracker combines blob-based methods for detection and target warping with Kalman filter prediction during tracking. One significant drawback of this approach is that knowledge of the camera model and the ground plane are required, which limits quick deployment in an arbitrary environment and limits its usefulness with moving cameras.

The M_2 tracker is another example of a unique system that makes use of 3D models and multiple cameras for the purpose of human tracking (Mittal & Davis, 2003). The hardware required for this tracking system is quite complex — anywhere from 4 to 16 synchronized cameras. Using knowledge of the target position in previous frames, the algorithm begins by segmenting objects in each camera image. Image-wise segmentation is based on color information (a cylinder model is used that represents a person's average color at a particular height) and the proximity of the projected pixel in 3D space to the estimated current location of the person. After successful segmentation, the cameras are used in a pairwise fashion whereby corresponding epipolar lines that intersect a particular person in each image are projected into 3D space. The projected epipolar lines are used to identify the points that comprise the person's body in 3D. After completing this process for all cameras and all individuals, a map can be created that shows the 3D positions of all the individuals projected onto the ground plane. The actual tracking mechanism is deployed on the computed ground plane map. Specifically, Kalman filters with constant velocity motion models are used to track ground plane blobs.

The system is quite impressive in its ability to effectively aggregate information across all the cameras to achieve accurate tracking even in a confined space with substantial occlusion. Moreover, the system initializes and learns its models at run-time, making manual initialization unnecessary. Of course, the system is not perfect. An elaborate 4 - 16 camera system is not overly practical outside of the laboratory

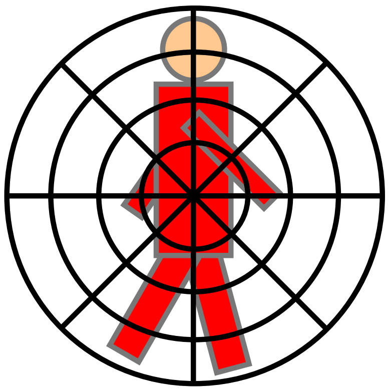


Figure 5.9: Regional target representation used in (Kang, Cohen & Medioni, 2003).

or specially designed indoor environments. Furthermore, the system relies on illumination variant color features and is quite computationally intense, running at 5 seconds/frame.

In (Kang, Cohen & Medioni, 2003), a multi-camera system is proposed that is much more general-purpose than that seen in (Mittal & Davis, 2003). Kang et al. assume a camera setup where a PTZ and stationary camera are present and have slightly overlapping fields of view. Affine warping is used to register consecutive frames from the moving camera so that background subtraction can be performed. A homography warp is used to bring the moving camera and stationary camera into correspondence. Tracking is achieved by maximizing a joint probability model that consists of a motion model and an appearance model. The motion model is represented using a unique Kalman filtering process that incorporates information from both PTZ and stationary cameras in the state vector. Specifically, the state vector used during Kalman filtering is comprised of the 2D position and velocity of the target as observed in each cameras' image domain. The appearance model is also somewhat novel in that it represents the color of the target using a polar coordinate-inspired "bulls-eye" representation. The bulls-eye representation, shown in Fig. 5.9, divides the target into bins that are at a particular quantized angle and distance from the target center. The average color of each of the quantized bins is computed and is represented with a Gaussian. This is another alternative color target representation that can be used in place of the color histogram or a single mean value. The advantage of this representation is that is retains some level of spatial information regarding the location of individual colors. Tracking is performed by maximizing a joint probability that is comprised of the individual motion and appearance probabilities.

Other trackers in the literature have used or extended the particle filter in their development. The BraMBLe tracker (Isard & MacCormick, 2001) shares some similarities with the M_2 tracker (Mittal & Davis, 2003), but rather than using an array of cameras, a single calibrated camera is deployed. In Isard and MacCormick's tracker, a grid with width and height spacing of 5 pixels is laid over the image. At each grid crossing, a bank of filters (radially symmetric Gaussians and second derivative of Gaussian-based Mexican hat functions) are applied to the three image channels. These filter responses are then used to compute observation probabilities — the probability of the filter responses, $\mathbf{z}_g$, at a particular grid location, given that the pixel corresponds to a particular layer, l_g (i.e. $p(\mathbf{z}_g|l_g)$). A separate layer exists for the targets within the scene and for the background. The foreground model, $p(\mathbf{z}_g|l_g = \text{foreground})$ and the background model $p(\mathbf{z}_g|l_g = \text{background})$ are both modeled as mixture of Gaussians and are learned from training data. This observation model is then used as the measurement distribution in a particle filter.

The novelty of the BraMBle system is that the observation distribution can obtain an unknown and varying number of targets (as opposed to more typical situations where the measurement distribution represents multiple hypotheses regarding a single target's location). To limit the effects of occlusion, individuals are represented as "cyliners" (more precisely, a set of four disks with a common vertical axis, but different diameters) in 3D space. The state during particle filtering encompasses both the 3D floor position and shape of the "cylinder". Fairly standard particle filtering machinery is employed using this unique observation distribution and the 3D human target representation. The system is fairly impressive, overall, as it tracks a varying number of individuals in a principled fashion, even with some occlusion. However, as tracking is performed using 3D state information, camera calibration information is required. Additionally, the tracker often fails when two persons cross paths causing almost complete occlusion. This limitation is most likely due to the fact that the authors elected to base the observation probability distribution on the two classes, foreground and background layers, rather than separating each foreground target layer.

Two other systems that made use of particle filters to develop trackers were presented by (Nummiaro, Koller-Meier & Gool, 2002) and (Okuma, Taleghani, Freitas, Little & Lowe, 2004). In (Nummiaro, Koller-Meier & Gool, 2002) a basic particle filter was designed that used color histograms for obtaining the measurement distribution. The majority of the system in (Nummiaro, Koller-Meier & Gool, 2002) is very similar to the mean shift algorithm (Comaniciu, Ramesh & Meer, 2000) — color histograms are computed using a kernel-weighting function and the Bhattacharyya coefficient is used for comparing histograms. However, instead of performing inter-frame tracking via mean shift optimization, the matching via color histograms enables the creation of a measurement distribution. This measurement distribution is then incorporated into a constant velocity particle filter. An interesting aspect of this work is the initialization scheme. Assuming that a rough template histogram is available, then particles can be placed at strategic locations within the field of view (e.g., at doors and at the sides of the image) where it is believed the target will enter. When a certain percentage of these particles produce a matching score that is substantially higher than general background matches, it is assumed that a target has entered. In

Chapter 2, we termed such initialization systems as "color-based target spotting" algorithms. Overall, the particle filter-based tracker compares favorably to a standard mean shift tracker and a mean shift tracker coupled with a Kalman filter.

The work of (Okuma, Taleghani, Freitas, Little & Lowe, 2004) also introduces an interesting AdaBoost-based detection mechanism into the particle filtering framework for the purpose of tracking hockey players. In a similar manner to (Nummiaro, Koller-Meier & Gool, 2002), Okuma et al. compute the observation probability distribution using color histograms. However, in an attempt to retain more spatial information, the target region is subdivided into an upper and lower section that are assigned separate histograms. Additionally, as the system focuses on the tracking of multiple hockey players simultaneously, an extension of the standard particle filter, is used. This filtering mechanism is known as the mixture particle filter (Vermaak, Doucet & Perez, 2003). Mixture particle filters allow for the simultaneous tracking of multiple targets and are actually somewhat similar in spirit to the observation probability proposed in (Isard & MacCormick, 2001). However, in (Okuma, Taleghani, Freitas, Little & Lowe, 2004), the observation probability is explicitly modeled as a mixture of components where each component corresponds to a different target.

In addition to the use of mixture particle filter, another novelty of the system in (Okuma, Taleghani, Freitas, Little & Lowe, 2004) is the incorporation of an Adaboost detection mechanism into the tracking process. A cascaded Adaboost algorithm (Viola & Jones, 2001), is used to detect the hockey players in each frame. This detection information is incorporated into the predictive stage of particle filtering. In standard particle filter trackers, a motion model is employed to realize the deterministic drift imparted on the target between two consecutive frames. However, in Okuma et al.'s work, the prediction distribution is defined as a weighted combination of a motion model and the Adaboost detection results. The authors argue that this combination yields superior results to using either component in isolation, which is not surprising, given the erratic movements of the hockey players that the system is tested on and the potential for partial or full occlusion.

In (Wu & Huang, 2001) the particle filter framework was expanded to accommodate a tracker that employed features of multiple modalities. The general problem of using a larger group of features when using a sampling based tracking method is that a larger number of samples are required to properly represent the distributions. In fact, as the dimensionality of the state increases, the number of required samples increases exponentially (Wu & Huang, 2001). The authors use factorization methods to divide the larger problem into a number of smaller problems that have state spaces with lower dimensionality, a process that they term "co-inference". The end result is a two-stage particle filtering system. The two modalities used here are shape and color. Thus, the system generates shape samples and subsequently updates these samples using shape measurement information. The sampling generation and observation procedure is repeated for color information. The interesting aspect is that color weights (weights, as defined in Eq. 2.14) are used for the shape samples and shape weights are used for color samples. The rationale is that the most effective samples are those that exhibit strength across both modalities. On the other hand, the inclusion of both modalities also provides additional robustness when one modality should fail. Indeed, the results that are presented in (Wu & Huang, 2001) clearly illustrate examples where if one modality was used in isolation, the tracker would undoubtedly fail.

Another, very unique, region tracker was presented by Tao et al. (Tao, Sawhney & Kumar, 2002). This tracker was developed primarily for following vehicles from a camera mounted on a moving, aerial platform. The authors elect to essentially segment the video into image regions, or layers, that appear to be moving coherently together, thereby tracking the target and everything else in the scene. The shape, motion, and appearance of each layer are all parametrically modeled and are optimized using the EM algorithm (Dempster, Laird & Rubin, 1977). The EM algorithm begins by assigning ownership probabilities for each of the pixels. Having divided the image into regions whose points presumably move in a similar manner, parametric motion estimates for each motion layer are obtained during the M-step of the algorithm. The iterative process of the EM algorithm is conducted until convergence is attained. The results reported are impressive, especially considering the low-contrast nature of the aerial video sequences. Unsurprisingly, the proposed tracker is shown to provide results that are superior to a simple template matching tracker and a temporal differencing-based blob tracker. However, it is not clear that a slightly more advanced tracker (e.g., a tracker that "subtracts off" camera motion via an affine warp and then performs frame-to-frame tracking on the registered images) could not achieve levels of success similar to the proposed system.

Along the lines of motion segmentation, the systems of (Murray, Bradshaw, McLauchlan, Reid & Sharkey, 1995; Daniilidis, Krauss, Hansen & Sommer, 1998) employ similar approaches whereby the motion field for an image is first computed. Subsequently, motion segmentation is performed and the dominant motion is assumed to arise from the motion of the moving camera. The remaining regions with significant levels of motion are concluded to correspond to the target. These target pixels are then grouped using morphological operators. In both works, the center positions of the blob targets are used to control the movement of an active camera.

Before concluding this chapter, we should briefly make note of a number of additional unique region trackers. In all of the tracking literature, there are numerous very original trackers. Rather than trying to list all of them, we will attempt to briefly highlight some of the most recent and impressive works. In (Spengler & Schiele, 2003), a system extremely similar to (Yin, Porikli & Collins, 2008) was proposed. A combined likelihood map is generated by taking the weighted summation of individual likelihood maps. The main difference between the work of Yin et al. and the system in (Spengler & Schiele, 2003) is that Spengler et al. set the target position as the maximally probable location in the combined likelihood map (rather than performing mean shift optimizations). Another recent and unique work is presented in (Leichter, Lindenbaum & Rivlin, 2007). This work extends kernel-based tracking by incorporating both color and edge features as well as allowing the kernel to undergo affine transformations. Another very interesting tracker documented in (Badrinarayanan, Perez, Cler & Oisel, 2007) continues the study of how to combine multiple modalities while tracking. In this work, a color-based particle filter tracker and a template-based tracker are run in tandem. Even though both trackers are continuously running, the combined system merely relies on the results of one of the two trackers. In the event that the results of the tracker that is currently being relied upon are deemed unreliable, control switches to the other tracker. This switching back and forth between trackers continues indefinitely.

Moving away from trackers that attempt to combine multiple modalities, in (Nguyen, 2006), a texture-based tracker is proposed that obtains very impressive qualitative tracking results. The texture features are derived using a bank of Gabor filters at multiple scales and orientations. While determining the optimal target translation, the system attempts to maximize the image textures that agree with previously observed target regions while simultaneously not matching background textures. Another interesting region tracker (Kim, 2008), combines blob tracking with rigid template tracking. Background subtraction is performed and corners in the foreground image regions are identified. Inter-frame tracking of corners is performed in a search region using normalized cross-correlation. Clustering is subsequently performed to group corners into higher level objects. Another corner-based tracker was presented in (Yin & Collins, 2007b). Once again, inter-frame matching of corner regions is performed using normalized cross-correlation; however, in this work, higher level objects are found using graph-based techniques. The work in (Han, Comaniciu, Zhu & Davis, 2008) makes contributions toward region-based probabilistic trackers. Specifically, a method is proposed for estimating multi-modal tracking densities in an online and efficient manner using techniques based on kernel density approximation.

Historically, researchers have been limited to either assuming unimodal distributions or have been restricted to using relatively expensive sampling methods with multi-modal distributions. Of course, exceptions to this rule have been previously demonstrated and discussed in this review (e.g., (Cham & Rehg, 1999)). Finally, as mentioned previously, some trackers employ continuous detection schemes that make use of the efficient integral histogram (Adam, Rivlin & Shimshoni, 2006; Nejhum, Ho & Yang, 2008). As the histogram representation is well-known to collapse spatial information, both of the abovementioned works attempt to retain additional spatial information by subdividing the tracking window into a number of smaller regions.

5.5 Recapitulation

In summary, most region-based trackers can be cast as belonging to blob trackers, point-wise template trackers, or kernel histogram trackers, but there are definitely some unique exceptions. Some of these remaining trackers extend the original particle filter framework while others use multiple cameras and 3D target representations. The combining of several sources of target information, a common theme throughout this paper, was also once again considered by trackers in this section. The next chapter will continue the theme of analyzing trackers that are especially unique. In Chapter 6, however, trackers that cannot be classified strictly as belonging to Chapters 3 - 5 will be highlighted.

Chapter 6

Combined Trackers

Although the majority of the trackers in the literature can be classified as belonging to one of the categories in Chapters 3 - 5, there will always be exceptions. In this chapter, trackers that do not fit well into any of the previous categories will be described. Particular attention will be paid to trackers that combine techniques seen in the previous chapters. We will divide the discussion of this special category of trackers into three sections. First, trackers that combine components from the three main categories of trackers (as discussed in Chapters 3 - 5) will be explored (Smith & Brady, 1995; Chung, MacLean & Dickinson, 2006; Shao, Porikli & Chellappa, 2007; Yilmaz, 2007; Rathi, Vaswani, Tannenbaum & Yezzi, 2007; Yang, Wu & Hua, 2008). Subsequently, trackers that probabilistically combine multiple modalities and multiple target representations (e.g., regions and contours) will be described (Rasmussen & Hager, 2001; Serby, Meier & Gool, 2000; Leichter, Lindenbaum & Rivlin, 2004; Han, Joo & Davis, 2007). Finally, the chapter will close with a description of silhouette-based trackers (Haritaoglu, Harwood & Davis, 2000, 1999; Wang, Bebis & Miller, 2006).

6.1 Category Combining Trackers

The previous three chapters have described three general categories in which the majority of trackers can be grouped. Although trackers typically employ methods from one of the three classes in isolation, it is possible to combine the techniques from various classes to create a combined tracker. It appears that combining pixel-wise warping with other types of trackers is a popular approach (Smith & Brady, 1995; Chung, MacLean & Dickinson, 2006; Shao, Porikli & Chellappa, 2007). Other works have attempted to combined contour tracking with kernel histogram methods (Yilmaz, 2007; Yang, Wu & Hua, 2008). These category-crossing systems will be described in this section.

Most of the region warping systems in the literature follow the general procedure of finding the optimal set of warping parameters that bring the template from the previous frame into closest alignment with a region in the current frame. The best matching location is considered to be the new target position. Other trackers make variations on this approach and use the warping mechanism in other manners. For example, in (Smith & Brady, 1995) warping is used to estimate the overall motion of a cluster of discrete points that have been tracked. Other systems use region warping as an initial motion estimate that is then refined using other techniques (Chung, MacLean & Dickinson, 2006; Shao, Porikli & Chellappa, 2007). These works will now discussed in more detail.

The ASSET-2 tracking system (Smith & Brady, 1995) is not particularly impressive in the sense that it is not based on an overly principled approach and the results are not groundbreaking. Nonetheless, in the context of this report, this tracker does have some unique properties. Specifically, the tracker includes ideas from all three of the previous chapters. Due to this unique combination of techniques, this work is worth at least a cursory discussion.

During execution, the ASSET-2 tracker performs five main processes:

1. Feature detection
2. Feature tracking
3. Flow clustering
4. Cluster Tracking and Filtering
5. Boundary Enhancement

Feature detection and tracking are performed in a fairly simplistic fashion. The Harris corner detector is used to locate feature points due to its low computational cost and its ability to consistently produce the same features. These interest point features are then individually tracked from frame-to-frame. In the first frame, initial matches are established using a nearest neighbor matching scheme that also incorporates appearance-based heuristics. Once initial velocity estimates have been established in the first two frames, a simplified Kalman filter is introduced. This filtering framework reduces the search space when performing data association in subsequent frames.

Once the correspondences between the interest points has been determined and the instantaneous velocity has been computed, the field of these point tracks are clustered into higher level objects. A unique clustering scheme is employed that attempts to grow clusters from initial seed regions. The current cluster considers all points to which it is spatially connected. When a cluster considers a single interest point, the difference between its instantaneous velocity and the current overall velocity of the cluster is computed. If the difference is sufficiently small, the interest point is added to the cluster. Once a cluster reaches steady state (i.e. no more connected interest points join the cluster), a new cluster is initialized and the process repeats. Note that as each cluster develops, it maintains an overall running velocity estimate for the high level object it represents. When just a few interest points belong to a cluster, this velocity is represented with a constant flow model. However, once a sufficient number of interest point measurements are obtained, an affine warp between the two frames is estimated for the cluster.

After cluster ownership and velocity estimates have been established, data association and predictive filtering are performed for the high level objects. Data association is performed based on the clusters' motion models and shape. Other heuristics are employed to make matching more reliable. Once data association has been established, the simplified Kalman filtering mechanism can be updated with the new measurement.

Clusters provide position and shape information that is used during the Kalman filter updating process. Positional information can be easily combined by taking a weighted difference between the predicted and actual position of the cluster. Comparing the shapes of the measured and predicted clusters is a little more complex. To that end, a radial target representation is used, which the authors argue to be somewhat analogous to snakes. As Fig. 6.1 shows, this radial target representation divides the space of 2D orientations into discrete bins. For each orientation bin, the distance between the centroid and the furthest cluster point is computed. Thus, this representation obtains a rough sketch of the target contour outline. Under this representation, the predicted and current shapes can be combined more easily by comparing the distances for each orientation bin. The final component of the system is an optional boundary enhancement stage that once again works with the radial target representation. The idea behind this stage is to ensure that the cluster boundary is aligned with true intensity edges within the image. Thus, along each 2D orientation, the defining boundary of the cluster is pulled inward or outward toward the closest image edge.

The system is demonstrated for the application of vehicle tracking. Instances where the camera is moving are considered and the overall results appear sufficient but not exceptional. Regardless, the unique approach and the culmination of ideas across a variety of tracking styles make this system noteworthy.

A more recent work that incorporates affine warping with contour-based techniques was presented by Chung, MacLean, and Dickinson (Chung, MacLean & Dickinson, 2006). As shown in Fig. 6.2, the general approach behind this work is to first estimate the target motion by computing an affine warp for the target. This affine warp provides an initial starting point upon which a snake-based contour can more precisely delineate the target boundary. Subsequently, the affine motion of the target can be re-estimated for the current frame by only considering the pixels that lie within the final contour. We will now discuss the main components of this system in

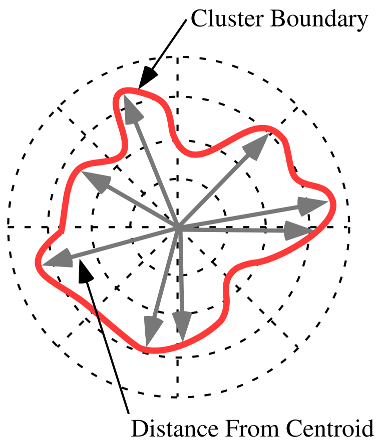


Figure 6.1: Radial target representation used in (Smith & Brady, 1995) that measures the maximum distance from the cluster centroid to the furthest cluster edge for discrete orientation bins.

more detail.

The central idea of the first stage of the system is to estimate a layered motion model of the scene. When computing this layered motion model, each pixel provides a constraint, $c(\mathbf{x})$. The probability that this constraint agrees with the current layered motion model can be computed according to

$$p(c(\mathbf{x})) = \sum_{j=1}^{n} \pi_j p_j (c(\mathbf{x}) | \theta_j, \mathbf{x}) \tag{6.1}$$

where π_j are mixing parameters, $\mathbf{x} = (x, y)$ are image domain coordinates, and θ are the parameters of the model. Essentially, Eq. 6.1 indicates that the probability of the constraint is the weighted sum of the probabilities that this constraint was generated by each motion layer.

Using Eq. 6.1, the most likely ownership of each pixel can be computed. In other words, the probability that a constraint comes from a particular motion layer, j, is

$$p(c(\mathbf{x})| j) = \frac{\pi_j p_j (c(\mathbf{x}) | \theta_j, \mathbf{x})}{\sum_{k=1}^{n} \pi_k p_k (c(\mathbf{x}) | \theta_k, \mathbf{x})} \tag{6.2}$$

Finally, the overall probability of a layered motion model is the product of the probabilities obtained through each constraint (i.e. the product of Eq. 6.1 for all pixels).

With these formulations in place, the EM algorithm (Dempster, Laird & Rubin, 1977) is used is order to compute an optimal estimate of the motion layers. The EM formulation used is similar to that seen in (Tao, Sawhney & Kumar, 2002). After warping the regions according to the parametric models obtained with the EM algo-

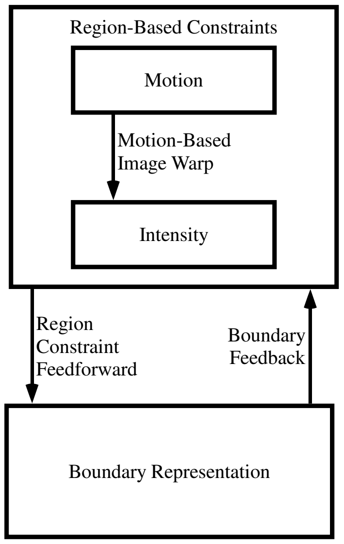


Figure 6.2: Block diagram of the system proposed by Chung et al. Adapted from (Chung, MacLean & Dickinson, 2006).

rithm, intensity comparisons are performed between each pixel in the current frame and its corresponding pixel in the previous frame. The difference in intensity between the corresponding pixels is represented via a Gaussian distribution. Furthermore, a normalized ownership, $Q_j(\mathbf{x})$, is computed that indicates the probability that pixel $\mathbf{x}$ truly agrees with motion layer j, given the brightness constancy assumption between corresponding pixels. These ownerships will be used when minimizing the snake energy.

The final component of the tracker in (Chung, MacLean & Dickinson, 2006) is to use a snake-based algorithm to fine-tune the tracking results obtained with the region-based methods. In this work, standard snakes are employed with an energy that consists of internal and external energy terms. The internal snake energy is defined in the same manner as in most snake works (Kass, Witkin & Terzopoulos, 1988; Terzopoulos & Szeliski, 1992; Leymarie & Levine, 1993) (i.e. Eq. 4.1). However, in this work, a unique external energy is proposed. Specifically, the external energy consists of terms related to distance, motion and intensity

$$E_{ext} = \gamma E_{dist}(x_i) + \delta E_{mot}(x_i) + \kappa E_{intensity}(x_i)\,. \tag{6.3}$$

As the name suggests, the distance energy is based on the distance from x_i to the nearest image edge. The motion energy considers the pixel motion ownership, Q_j. If this ownership is above a threshold for the current region, the energy is further reduced to pull the contour toward that point. Thus, pixels that are aligned with image edges but do not have strong motion ownership will be attractive to the snake.

However, if a pixel has strong motion ownership and is aligned with an image edge, it is an extremely good candidate contour point. Finally, the intensity energy is based on balloon forces. Intuitively, if a contour point's motion ownership, Q_j, is largest for the current region, j, then the contour will be encouraged to expand outward along the normal. If the pixel's intensity constraint, Q_j agrees more strongly with a different motion layer, the current snake will be encouraged to shrink inwardly along the normal direction.

In summary, Chung et al. provide a tracker that effectively combines region-based methods with contour-based methods. The results of their system are fairly impressive. Some of the test sequences illustrate the tracking of toy construction vehicles. These vehicles have very complex shapes with deep indents and sharp corners. Nonetheless, the combined tracker is able to delineate the boundaries quite accurately. Certainly, the boundaries are delineated more accurately with the proposed algorithm than they would be with most region-based trackers. One or two worrisome aspects of the results include the fact that no occlusion experiments are shown. One sequence considered will clearly have a significant occlusion event but the authors elected to not illustrate results on these frames. Furthermore, almost all sequences considered targets that only experienced essentially horizontal translational movement.

Another paper that shares many similarities to that of (Chung, MacLean & Dickinson, 2006) was presented by Shao et al. (Shao, Porikli & Chellappa, 2007). Once again in this work, image warping is first performed to obtain an initial contour estimate in the current frame that is then refined using fairly standard contour-based techniques. More explicitly, the tracker performs three main processes: motion estimation, contour deformation, and shape regulation. We will now discuss in more detail the three steps of this tracking system.

The authors perform a standard affine transformation to the B-spline control points that define the contour. A unique aspect of this work is that the affine parameters, θ_t are estimated using a particle filtering framework. Thus, the state of the particle filter is simply the six affine parameters and the filtering framework follows very closely to the general particle filter explained in Chapter 2. Specifically, prediction is performed in a straightforward manner using the previous affine parameters. Furthermore, the update/measurement process is almost identical to that seen in the CONDENSATION algorithm (Isard & Blake, 1996).

Given that a single warp has to be found that optimally meets the constraints of all contour control points, it is possible that some or many of the control points are not exactly aligned with true intensity edges within the image. As a result, a subsequent refinement stage massages the curve such that the control points can locally move to be aligned with intensity edges. The contour refinement stage in (Shao, Porikli & Chellappa, 2007) is similar in spirit to many of the contour trackers discussed in Chapter 4. Specifically, lines normal to the contour are constructed at each discrete point and a search is conducted along this 1D line to find the optimal position. There is one unique aspect of the refinement algorithm proposed in (Shao, Porikli & Chellappa, 2007) — the normal lines' length and centering are adaptive. The normal line lengths are determined from training sequences. In particular, during training, portions of the target contour that vary frequently (e.g., swinging arms/legs on a human) are identified and are determined to require a larger search radius. In contrast, sections of the contour that move relatively steadily (e.g., torso and head of a human) can use shorter normal lines. As mentioned, the center position of the normals can be adapted as well. The intuitive idea is to adjust the lines based on a distance transform to minimize the overlap and tangling that occurs between various normal lines.

By searching along the set of normal lines to the contour, the most probable positions of the control points are established using a number of cues including: edge magnitude, edge orientation, shape templates, and foreground estimation. These different modalities are fused in a probabilistic sense to maximize the probability of observing the combined features, giving the estimated contour control points. As is common throughout the contour tracking literature, the modalities are assumed to be independent.

After the contour-based optimization step, the tracker in (Shao, Porikli & Chellappa, 2007) performs one final stage of processing. This, shape regulation stage, ensures that the contour matches allowable target shapes that were observed during training. Specifically, in a similar fashion to Eigen representations, a basis set is constructed using training imagery. This basis set defines a model of the allowable shapes that the target can take throughout the sequence. During this final stage, the estimated contour is projected onto the shape subspace provided by the basis vectors. The projected version of the contour is the final estimate for the current frame.

The experimental results reported in this work are quite respectable. The tracker is applied mostly to the problem of tracking humans. In many cases, the human targets are very small but the tracker nonetheless provides accurate contours. One noticeable area of weakness is that the tracker can not reliably segment deep contour concavities (e.g., the area between a human's outstretched legs). In fairness, most contour trackers struggle with this problem, with one notable exception appearing to be the system in (Yilmaz, Lin & Shah, 2004). Even still, Shao et al. demonstrate effective tracking results with moving cameras, substantial shadow effects, partial occlusion, and a quickly zooming camera.

A recently proposed work that combines warping machinery with contour tracking was proposed in (Rathi, Vaswani, Tannenbaum & Yezzi, 2007). In this work, a particle filter was incorporated into a level set tracker to smooth the target track and allow for the recovery of larger target motions. The particle filter is used to estimated the affine motion parameters that describe the inter-frame global motion of the target. Subsequently, contour evolution is used to recover the local deformations of the target. This idea of recovering the global target motion with an affine warp and the local deformations via curve evolution is reminiscent of the other approaches just discussed.

Finally, another work that shares similarities to the works by Chung et al. and Shao et al. is presented in (Marchand, Bouthemy & Chaumette, 2001). In this work, an affine warping mechanism is used to obtain a coarse estimate of a target's motion that is subsequently refined using 3D model-based methods. The two stages of tracking (warping and 3D model-based) use very similar techniques to those already mentioned in this review and thus, will not be elaborated upon. The system does offer an additional novelty — that of incorporating active camera controls. In particular, an image is obtained beforehand that shows the target in a reference pose. The goal of the active camera system is to attain the reference viewpoint of the target, when starting from any arbitrary view. To that end, the distances between corresponding features in the reference frame and the current frame are computed. The distances are then provided to a visual servoing system that brings the current viewpoint into alignment with the reference view in a manner that decreases the error in an exponential fashion. Sample results showing the tracking of fairly simple polyhedral objects are shown in (Marchand, Bouthemy & Chaumette, 2001). Despite the fact that the targets are simple, rigid shapes, they are often tracked with accuracy in front of very cluttered backgrounds and throughout minor shadow effects.

In addition to combining a warping mechanism with other tracking machinery, systems have attempted to combine contour-based ideas with kernel histogram methods (Yilmaz, 2007; Yang, Wu & Hua, 2008). In (Yilmaz, 2007), a more precise kernel is obtained by precisely delineating the target boundary in the first frame. The kernel is then defined as the signed distance from the contour, where regions inside the target receive positive values and those outside receive weights of zero. Note that this is a similar function to what is used when typically defining a level-set surface. This "level set kernel" is then used during tracking. One significant limitation of the approach is that the kernel does not have the ability to adapt its shape over time. Note that this work is conceptually very similar to (Leung & Gong, 2006) (discussed in Chapter 5), where normalized cuts were used to determine the shape of a target in the first frame before performing kernel histogram tracking.

Continuing with combined systems that make use of kernel histogram methods, in (Yang, Wu & Hua, 2008) a very different approach is undertaken that improves tracking results by dynamically making use of other scene objects. To leverage a scene object during tracking, the object must consistently appear with the target, have motion that is correlated with the target's own motion, and be easy to track. The surrounding objects are identified based on their color and motion characteristics and are identified using online data mining techniques. These surrounding objects are tracked using the standard mean shift algorithm while a contour-based tracker follows the true target of interest. Very impressive results are shown when the tracked region is part of a larger object (e.g., tracking the face of a person as the main target and other body parts as the surround objects).

Finally, in (Babu, Perez & Bouthemy, 2007), a rigid template tracker was combined with a mean shift tracker. Once again, the modules are run in a serial manner such that the output of the rigid template tracker provides an initial motion estimate for the mean shift tracker. Furthermore, likelihood maps are used to define pixel-wise weights that indicate the probability that a pixel belongs to the foreground target. Pixels that are likely to have originated from background regions are downweighted during SSD and mean shift tracking. Finally, mean shift tracking is performed using several tracking windows placed randomly over the target region. The likelihood maps as well as the pixel-wise weights allow the system to determine which of the mean shift tracking windows should be trusted when updating the target position. This design allows the system to be much more robust to partial occlusions because the system only requires a subset of the mean shift tracking windows to be visible to obtain accurate tracking results. In the original mean shift tracker (Comaniciu, Ramesh & Meer, 2000), partial occlusions would be much more problematic and might cause the single tracking window to drift off of the center of the true target. This behavior often occurs because the tracking window tries to "avoid" surfaces that partially occlude the target when they provide a very bad match to the template.

In summary, one method to make use of multiple tracking paradigms is by using the various components from the three main chapters in a sequential or combined fashion. It appears the most popular approach is to combine a warping-based approach with a secondary mechanism, although other combinations were also briefly discussed.

6.2 Probabilistic Combination Trackers

In the previous section, we saw how different tracking paradigms can essentially be combined by performing them in a sequential fashion to continuously refine the tracking results. Another method of combining various modalities and tracking paradigms is by relying on probability theory. Systems that take this probabilistic combination approach will be elaborated upon in this section.

In (Serby, Meier & Gool, 2000) a tracker was developed that combines a variety of features in a probabilistic framework using a particle filter. The features vary from curves to textured regions. The tracker uses a very straightforward particle filter implementation where the state is composed of the 2D target position and its six affine motion parameters. Since we have already discussed the standard particle filter formulation in Chapter 2, here we will focus on the system's observation models, $p\left(\mathbf{z}_t|\mathbf{x}_t\right)$ for the various modalities.

As is typically the case, in (Serby, Meier & Gool, 2000) it is assumed that the various feature modalities are independent. The authors also make the assumption that the low-level template features, $\left(x_t^1, \ldots x_t^M\right)$, are rigidly linked. Thus, the target state can be completely described by applying the warp transformation parameters to the low-level features that describe the target. This transformation allows the comparison between the target state and the image measurements. Combining the aforementioned assumptions, the observation probability is

$$p\left(\mathbf{z}_t|\mathbf{x}_t\right) = \prod_{i=1}^{I} p_I\left(\mathbf{z}_t|\mathbf{x}_t^i\right) \prod_{j=1}^{J} p_E\left(\mathbf{z}_t|\mathbf{x}_t^j\right) \prod_{k=1}^{K} p_T\left(\mathbf{z}_t|\mathbf{x}_t^k\right) \prod_{l=1}^{L} p_H\left(\mathbf{z}_t|\mathbf{x}_t^l\right), \tag{6.4}$$

where p_I, p_E, p_T, p_H correspond to the interest point, straight/curved edge, textured region, and homogeneous region observation probabilities, respectively.

The observation densities for each of the modalities are all defined similarly using negative exponentials. The interest point probability, $p_I\left(\mathbf{z}_t|\mathbf{x}_t^i\right)$, is based on the negative exponential of the distance between the template interest point, $\mathbf{x}_t^i$ and the nearest interest point found using the Harris corner detector. Similarly, for computing, $p_E\left(\mathbf{z}_t|\mathbf{x}_t^j\right)$, normal lines along the target boundary are drawn and the minimum distance is computed between the target's boundary and the nearest edge point found using a 1D edge detector. The texture probability is evaluated using the Bhattacharyya coefficient between the color histograms generated by the template and the measurements. Finally, $p_H\left(\mathbf{z}_t|\mathbf{x}_t^l\right)$ is determined using the average color difference between the template and measurement regions.

The results in this paper show examples where a tracker that just uses a single modality fails but the proposed tracker succeeds. Specifically, an example is shown where a book is tracked in a cluttered environment. If a standard contour tracker that only employed intensity edge information were used, the system would fail due to the background clutter. A second example is shown where a car is successfully tracked even though other cars with similar color distributions are driving nearby. In this second case, the proposed tracker once again succeeds, but a color histogram-based tracker fails. Although the limited results seem reasonable, it is hard to speculate on the overall utility and power of the tracker given that only two video sequences were demonstrated.

Another paper that considers the probabilistic combination of multiple feature modalities when tracking is (Rasmussen & Hager, 2001). However, in this paper, the tracking via multiple modalities is performed using statistical data association tools that we discussed in Chapter 2. In this work, the authors present trackers that use three different feature modalities: homogeneous regions, textured regions, and snakes. As we can see, these modalities correspond to most of the feature-types that were employed in (Serby, Meier & Gool, 2000). We will now discuss the various features and statistical data association techniques used in this system in more detail.

As is the case with many systems that formulate the tracking problem in a probabilistic fashion, the goal is to compute the maximum *a posterior* estimate of the tracker state. Using Bayes' rule, we can compute this estimate using the likelihood

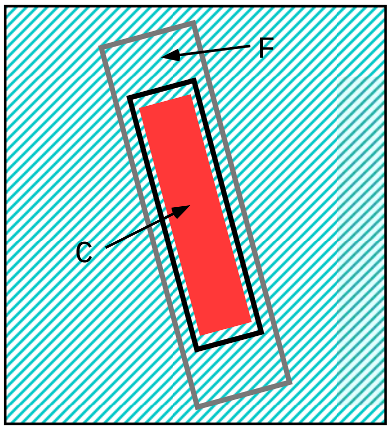


Figure 6.3: Homogeneous target representation of a target and its surrounding background used in (Rasmussen & Hager, 2001).

function. Thus, the first aspect of the tracker in (Rasmussen & Hager, 2001) that must be considered is the definition of the likelihood functions for each tracker modality. With respect to the homogeneous regions, the goal is to determine the optimal location, size, and orientation of a rectangle C that corresponds to the target. As shown in Fig. 6.3, an outer rectangle, F, that shares the same center point and orientation as C but is much larger, is also used when computing the maximum likelihood. The goal is to find the best placement of the rectangles such that C maximally matches the predicted color and F is maximally different from the predicted color.

The second modality that can be used during tracking is texture. With textured regions, the authors make use of SSD-based matching scores to define the likelihood function. The final tracking modality considered in (Rasmussen & Hager, 2001) uses intensity-based snakes. The snake-based likelihood function in this work is very reminiscent of that derived in the CONDENSATION algorithm. In particular, the likelihood is based on the sum of the differences between the predicted contour points and the closest measured edges along normal lines.

Having defined the likelihoods for the various modalities, the main outstanding question that must be defined is: "in what tracking machinery are these likelihood functions used?" For example, in this review, we have shown that such likelihood functions are used extensively in Kalman filtering. Furthermore, this maximum *a posterior* approach is a main component of particle filtering. In (Rasmussen & Hager, 2001), several statistical tracking tools are explored, including the probabilistic data association filter, the joint probabilistic data association filter, and the so-called joint-likelihood filter. We will briefly elaborate upon the differences between these three techniques and how they are deployed in the proposed trackers.

When only attempting to track a single object with a single modality, Rasmussen et al. suggest that a PDAF-based approach can be employed. The PDAF-based scheme can be effective in tracking single objects in simple environments. However, if two targets with similar appearances are in a scene moving close to one another, there is a good chance that the PDAF filter will get confused. In particular, the proposed PDAF-based scheme does not contain any mechanism for attempting to determine which measurements stem from the true target and which are a result of the distractor. The problem is even more significant because oftentimes structures in the background will appear similar to targets as well. Thus, predominant structures in a cluttered background can distract the PDAF-based filter if a suitable feature set is not used. Fortunately, there are tracking tools that perform statistical data association when multiple targets are present, such as the JPDAF that was already discussed in Chapter 2. The JPDAF provides a statistical mechanism for determining measurement to target correspondences. Specifically, this filter ensures that a target only correspond to a single measurement and vice versa.

Although the JPDAF allows for the tracking of multiple targets, one shortcoming of the approach is that all trackers must be of the same modality. Furthermore, a JPDAF-based approach experiences difficulty when the targets of interest overlap, precisely the time when the best performance is needed. The measurement process breaks down because the targets are assumed to be independent. However, when the targets move closely to one another, this assumption is violated (e.g., due to occlusions). To ameliorate these shortcomings, the authors propose a joint likelihood filter (JLF). The main difference between the JLF and the JPDAF is that the JLF considers the depth ordering during the measurement process. With the JLF, the relative depth orderings between the various targets are enumerated, and the most probable depth orderings are maintained. The targets are no longer assumed to be independent and as a result, the depth ordering effects the computation of the joint likelihood function. Despite its added complexity, the JLF more accurately models real-world interactions between targets. The system performance is shown to benefit from this modification.

In (Rasmussen & Hager, 2001) results are shown for trackers that are designed using each of the different modalities. Furthermore, examples are shown with each of the different tracking filters. Qualitatively, the JLF appears to be fairly effective at tracking similar colored objects even during partial occlusion. Specifically, an example is shown whereby two chess pieces of the same color are tracked through a partial occlusion. One of the trackers uses the snake modality while the other uses the textured region modality. Other examples illustrate effective tracking of specific human body parts. For instance, one example shows two individuals walking in opposite directions as two homogeneous region trackers attempt to follow their heads. During the occlusion event, the PDAF tracker is lured away from the occluded individual onto the person in the foreground. In contrast, the JPDAF framework can handle this form of occlusion and successfully continues tracking the two heads after occlusion. Overall, the most striking aspect of this work is the ability to maintain multiple trackers that continue following the appropriate targets even during occlusion. Furthermore, the fact that the system can track very specific body parts (e.g., a person's arm) is quite impressive, as well.

Another work by Han et al. (Han, Joo & Davis, 2007) argues that blindly fusing the results from multiple sensors, as is often done, is suboptimal. In general, sensor fusion in a particle filter framework will treat each modality evenly and will generate an overall likelihood function by simply taking the product of the individual likelihoods. As an alternative, the authors propose generating separate target tracks using each of the different modalities and subsequently combining the modalities based on the confidence of the obtained tracks. In a similar vein, in (Leichter, Lindenbaum & Rivlin, 2004) a theoretical analysis is provided for combining the results of multiple probabilistic-based trackers that are running in parallel. Derivations are shown for parallel trackers that use different state spaces. Furthermore, the authors consider both trackers that propagate analytic distributions as well as those that consider particle-based distributions.

To summarize, in this section (and throughout this review), we have shown that a second method of combining features and tracking modalities in a principled manner is through the use of a probabilistic framework.

6.3 Silhouette Trackers

Oftentimes, silhouette trackers are simply considered to be in the same category as contour trackers. The reason these types of trackers are often grouped together is that silhouettes and contours essentially represent the same information — both representations define the border between the foreground object and the background. However, for the works considered in this paper, the methodological differences between contour trackers and silhouette trackers is substantial. As discussed in Chapter 4, contour trackers will typically make use of an energy function that is subsequently optimized using differential equations and oftentimes, level sets. Silhouette trackers, on the other hand, typically analyze the shape of the silhouette and will use tracking mechanisms that are more often seen in blob tracking frameworks. Thus, the silhouette trackers considered here share similarities both with contour trackers and with region-based trackers. Accordingly, these silhouettes trackers find themselves in this final category of other/combined trackers.

Perhaps the most famous silhouette-based system is W^4 (Haritaoglu, Harwood & Davis, 2000) and its subsequent extensions, such as Hydra (Haritaoglu, Harwood & Davis, 1999). W^4 is an extensive surveillance system that includes modules for background modeling, foreground person detection, single person tracking, and group tracking. Furthermore, additional modules examine whether or not a person is carrying an object and the tracking of particular body parts. Examination of the entire surveillance suite is outside the scope of this review. Instead, we will focus on the various tracking techniques that are employed in W^4 and Hydra.

The tracking modules (as well as most of the other system models) in (Haritaoglu, Harwood & Davis, 2000, 1999) are primarily based on heuristics rather than rigorous mathematics. As a result, the tracking mechanism are rather simplistic. Notably, an individual is represented by a bounding box (encompassing his or her foreground silhouette), as well as the median position of the foreground region. When tracking a single person, second order motion models are used to estimate his or her position in the current frame. The bounding box corresponding to the predicted position is compared against the detected foreground bounding boxes in the current frame. One-to-one matching is established between predicted boxes and measured boxes based on nearest neighbor matching.

One might imagine that the median centroid of the foreground region could be used to update the dynamic model of the corresponding individual. However, the authors of (Haritaoglu, Harwood & Davis, 2000, 1999) argue that such an approach will not provide reliable results because the consistency of the detected silhouettes is poor. Since the exact shape of the silhouette will change significantly from frame-to-frame, simply following the centroid position will provide very noisy and inaccurate velocity estimates. Instead, it is proposed that a more precise algorithm is used to bring the predicted and measured targets into closer alignment. Specifically, a binary edge correlation search is performed, whereby the predicted silhouette is moved within a small search region to find the best match. This secondary alignment procedure enables more accurate target velocities to be computed.

The tracking of groups of people is performed in a very similar manner to that of tracking individuals and was original proposed in the Hydra extension to W^4 (Haritaoglu, Harwood & Davis, 1999). The main difference is that the tracking is performed using only the heads of the individuals within the group. Thus, it is necessary to first determine the positions of the individuals' heads in the foreground silhouette. To perform this estimation, a simple, two-step heuristic approach is employed. The head detection module begins by analyzing the group foreground silhouette using a convex hull algorithm. This algorithm returns candidate head positions on the outline of the silhouette. Based on the distance to the predicted head locations, the pixels within the silhouette are determined to belong to a particular head (i.e. person's body) using a minimum Euclidean distance criterion. Subsequently, for a particular person within the group, his or her corresponding pixels are projected onto an axis perpendicular to the major axis defined by his or her torso. In the resulting "projected histogram" one would expect the true head location to correspond to the highest histogram peak. Candidate head positions obtained through the convex hull algorithm are only kept if they have sufficient support from the projected histogram.

Once an individual's head has been detected within the group silhouette, the tracking process is very similar to that of tracking a single person. In particular, data association is established between the predicted position of the head and the closet measured head in the silhouette. However, instead of using binary correlation (as was done in the single person case), precise positional matching is established using intensity-based correlation in a local search region.

The overall performance of the W^4 surveillance system is quite impressive, especially at the time of its initial inception. To this day, it still remains a heavily cited tracking and surveillance work. The system is reported to perform quite reliably over long ranges of time and through different environmental conditions. As mentioned, this system is mainly ad hoc, which in itself is not a problem, so long as the system performs up to the desired standard. For the particular range of applications the authors consider, the W^4 tracker performs quite well. However, everything is not perfect with the W^4 tracker. If it were, many aspects of the computer vision surveillance problem would be "solved." One of the problems with the W^4 system is that it appears very brittle and overly reliant on ad hoc assumptions. The system is only illustrated to work on a variety of video sequences with a camera angle that is fairly close to the ground. Furthermore, typically the humans have a number of pixels on target, resulting in projection histograms of a particular shape. If the camera viewpoint is changed such that it is much further overhead or that the targets of interest are further away (resulting in fewer pixels on target), it appears the W^4 system would struggle.

Another downside of the assumptions made by W^4 is that it is almost completely reliant on illumination variant information. Foreground detection and tracking both rely heavily on intensity information. If foreground detection is performed poorly, the whole surveillance system will break down. Thus, changes in illumination are known and reported to cause significant problems with the W^4 system. When global illumination changes arise, the system resets itself and reconstructs a new background model. This solution is obviously not ideal, but at least it provides an automated method of recovering from global illumination changes. An aspect of illumination not considered in this work is that of local illumination changes caused by, for example, point lights or shadows. During intensity-based correlation, we expect local intensity changes would significantly degrade the results of the W^4 system.

A more recent example of a silhouette-based tracker was presented in (Wang, Bebis & Miller, 2006). In this system, background subtraction is first performed and inter-frame matching between the blobs is performed using the shape of the silhouette. In particular, vertical and horizontal projection histograms are created from each silhouette. These projection histograms are compared against the histograms of tracked objects using the Manhattan distance. A unique aspect of this work is that once inter-frame tracking has been performed, feedback is provided to the background subtraction algorithm for the purpose of dynamically modifying system thresholds. The thresholds are modified in an attempt to keep the foreground silhouettes of the targets as consistent as possible across the video sequence.

6.4 Recapitulation

In this chapter, we have described tracking systems that do not correspond to the categories proposed in Chapters 3 - 5. In fact, many of the trackers proposed in this chapter combine various other methods either in a probabilistic framework or in a sequential manner. A popular approach when considering the sequential combination of tracking paradigms is to obtain a rough estimate of the target motion using parametric warping methods, and subsequently, to refine the estimate using a contour tracker. Unique silhouette trackers were also discussed in this chapter. Although a silhouette is inherently the same as a contour in that it defines the boundary between target and non-target regions, the methods used by silhouette trackers are very different from standard contour-based trackers. In the next chapter, we will discuss the current state of visual tracking.

Chapter 7

Current Status

This chapter will focus on the current status of the field of visual tracking. We will begin the chapter by commenting on the qualitative and quantitative methods that are most commonly used to evaluate tracking systems. With respect to qualitative evaluation, a set of often-considered sub-problems will be defined. Subsequently, the various classes and subclasses of trackers that have been summarized in this report will have their strengths and weaknesses compared, based on the abovementioned evaluation criterion.

7.1 Evaluation Criterion

As is the case in almost any research field, methods are required for evaluating the performance of the proposed idea or product. Preferably, these evaluation criterion will consist of a standard set of quantitative metrics that allow for fast and accurate comparisons. Unfortunately, visual tracking does not currently have a standard method of evaluating proposed trackers (Heckenberg, 2006). As a result, when submitting a new tracking work to a conference or journal, it is the authors' responsibility to prove, with whatever metric they feel is necessary, that their system makes a significant contribution. Despite the lack of a standard quantitative methods, there are a number of subproblems in visual tracking upon which trackers can be qualitatively judged. Furthermore, there are several publicly available data sets that can be used for evaluation purposes. These subproblems and data sets will be discussed in this section.

7.1.1 Subproblems in Visual Tracking

To completely "solve" the problem of visual tracking, there are a number of challenges that the ideal tracker must be able to accommodate. These subproblems include:

1. Automatic Initialization
2. Robustness to Partial and Full Occlusion
3. Robustness to Illumination Changes
4. Simultaneous Tracking of Multiple Targets
5. Robustness to Foreground and Background Clutter
6. Tracking of Non-Rigid Targets
7. Robustness to Scale Changes.

These subproblems are fairly self-explanatory and most have been mentioned at one or more points throughout this paper.

Due to the lack of of standard, quantitative methods of comparing algorithms, researchers often resort to using the abovementioned categories to evaluate their proposed systems. In fact, typically a new work will be published that solely focuses on contributing to one or more of the above sub-problems. For example, in (Yilmaz, Lin & Shah, 2004) a contour-based tracker is proposed that is claimed to be able to cope with partial occlusions. Accordingly, the authors select and demonstrate results for video sequences where occlusions occur. Since no claims are made regarding the other subproblems, results concerning these other subproblem are not shown in (Yilmaz, Lin & Shah, 2004). As a result, we have little knowledge regarding how robust the tracker in (Yilmaz, Lin & Shah, 2004) is to foreground and background clutter, for example.

7.1.2 Data Sets

Ideally, the field of visual tracking would contain a database of standard videos that would be universally used by researchers. Furthermore, for each database, it would be ideal if there were particular quantitative metrics that could be utilized to define the relative success or failure of the proposed algorithm. Unfortunately, the field of visual tracking is not currently at that point. Nonetheless, there are several publicly available databases that can be used for system evaluation. These databases will be the discussion of this subsection.

Some of the most prominent sources of data for surveillance and tracking problems are: (1) the Performance Evaluation of Tracking and Surveillance (PETS) databases (PETS, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007); (2) the Gait-Based HumanID Challenge by the Defense Advanced Research Agency (DARPA) (Sarkar, Phillips, Liu, Vega, Grother & Bowyer, 2005); and (3) the Video Verification of Identity dataset (VIVID) (Collins, Zhou & Teh, 2005). It appears that the PETS databases are by far the most commonly used. Nonetheless, we will take a brief moment now to discuss each of these sources of data.

Of any of the data that is publicly available, it appears that the PETS databases are by far the closest to becoming the standard. Annually, a new PETS database is released to the general public. Alongside the data itself, a particular surveillance challenge is posed. For example, for PETS2007 data, one of the goals is to detect the loitering of individuals in the video sequences. Ground truth results are also provided for the data. In more recent PETS databases (PETS2006, PETS2007) this ground truth is primarily used to identify the position of a piece of luggage in forgotten or stolen luggage scenarios. However, previous databases (e.g., PETS2004) have provided ground truth data identifying the spatiotemporal coordinates of interacting humans. Researchers in the field of tracking and surveillance are then encouraged to test their algorithms (or design new algorithms) on the new sources of data as they become available every year. A PETS workshop is held annually in conjunction with a major academic conference and the best algorithms with respect to that year's PETS database will be published in the conference proceedings.

The PETS workshops have now been running for seven years. Some of these datasets focus on surveillance in outdoor environments (e.g., PETS2001) while others consider surveillance problems in airport or train stations (e.g., PETS2006, PETS2007). The dataset for PETS2004 consists of indoor surveillance footage in the foyer of INRIA Labs in France and is commonly referred to as the Context Aware Vision using Image-Based Active Recognition (CAVIAR) dataset (Fisher, 2002).

Although the PETS databases are definitely a significant step in the right direction, they are far from perfect. Outside of the PETS workshops themselves, researchers do make use of the data when evaluating their proposed approaches. However, when these tests are performed, more often than not, the authors will use the data in a manner that was not originally intended. In other words, the ground truth data that is provided with the data is not typically used. Perhaps the reason for this abuse of the data is that the PETS challenges proposed every year are often very narrow in focus. For example, PETS2006 and PETS2007 are primarily concerned with detecting and recognizing security issues at transportation hubs. The main challenges with these datasets is to detect activities such as the abandoning/theft of luggage and loitering. Visual tracking, unto itself, is not typically the primary goal of the PETS challenges. Instead, the PETS organizers tend to focus on higher level activities. This fact is not overly surprising, as PETS is a first and foremost, a surveillance project, not a tracking project. Since tracking is not the focus of the problem (but more of a step along the way to successful higher-level event understanding), the sequences in these datasets are not designed to rigorously and systematically test the aforementioned subproblems in visual tracking. More specifically, PETS data is not designed to measure algorithms' ability to cope with changes in illumination, target scale changes, etc.

Another source of data that is publicly available for the tracking community is the HumanID gait recognition database (Sarkar, Phillips, Liu, Vega, Grother & Bowyer, 2005). The data was collected at the University of South Florida where 33 subjects individually walked an elliptical-shaped path. Different experimental conditions were considered whereby the types of shoes, walking surfaces, and starting points were varied. Additionally, sequences were made for individuals when they were carrying an object and when they were unencumbered as well as from two different viewpoints. As one might guess, the primary focus of the HumanID challenge is not single person tracking. Instead, the goal of this work is to perform gait identification — to consistently determine the identity of a walker across all experimental conditions. In a similar manner to the PETS data, this dataset provides video sequences where the tracking of humans can be performed. However, the human tracking is rather straightforward, as there is only a single person moving within the scene as observed by a stationary camera. Thus, once again we find that although tracking is possible using this data, doing so does not rigorously test the subproblems that require resolution in the field of visual tracking.

In contrast to the sources of data just discussed, VIVID is one database that was designed for the purpose of target tracking (Collins, Zhou & Teh, 2005). In particular, this data set focuses on the tracking of ground vehicles from airborne sensor platforms, and is maintained by the Robotics Institute at Carnegie Mellon University. Since this data was designed for tracking, a number of the subproblems in the field are considered. For example, some of the sequences involve: multiple moving objects that are similar in appearance, moving cameras, and occlusions. Ground truth data is available for determining the percentage of frames that the target is successfully tracked as well as the accuracy of the target shape. Although PETS data is certainly the most renowned of all the datasets considered in this review, VIVID is arguably the best suited for the problem of visual tracking. However, the VIVID database does have its drawbacks. To begin, only a handful (ten) of sequences are available. Furthermore, all sequences consider the problem of tracking vehicles in airborne imagery, a field that is much less popular than tracking humans for surveillance purposes. Additionally, although the database provides sequences that consider some of the subproblems of visual tracking, the videos do not systematically and rigorously test a tracker with respect to these challenges. Finally, it appears that the data is not commonly used by researchers.

7.1.3 Quantitative Analysis

Quantitative analysis looses some of its impact when the data that is being used is non-standard. However, it is always preferable if some form of quantitative results are included in the event that future researchers would like to compare their algorithms on the same videos using the same metrics. Furthermore, even if the numerical values can not be compared against other systems directly, some form of numeric score provides a reasonable summary of a system's performance. Not unlike the situation with data sets, in visual tracking there is really no standard quantitative measure of system performance. In this section, we will elaborate upon some of the metrics that have been used in the past.

In the point tracking literature, Shafique et al. propose a metric that is quite rigorous when dealing with synthetic data (Shafique & Shah, 2003). In particular, a "modified track-based error" is computed as: $E_T^C = 1 - \frac{T_C^C}{T_t^C}$ where T_C^C is the number of *completely* correct point tracks computed by the tracker and T_t^C are the total number of tracks that extend from the first to the last frame of the sequence. In order for a point track to be considered correct, the labeling of the point must be consistent and correct for every frame of the video. This error sets a fairly high standard for point trackers because even one mistake will invalidate a track. Presumably, it would also be possible to lower the standard so that small errors in a track would still be considered to be "correct". For example, another error metric derived from the track-based error might count the number of tracks in which $X\%$, or more, of the points along a track are correct (where $0 \leq X \leq 100$). To our knowledge, no such percentage-wise analysis has been performed in the tracking literature.

Other discrete feature trackers also tend to rely on visual inspection when it comes to evaluating their algorithms. Of course, there are some notable exceptions. As is common with other types of trackers as well, in (Zhang & Faugeras, 1992), the utility of the proposed line tracker is evaluated by comparing the recorded motion parameters against the true motion parameters. This evaluation method can only be used when the ground truth motion of the target(s) is known. With respect to 3D model-based trackers, it appears that most works (Crowley, Stelmaszyk & Discours, 1988; Lowe, 1992; Koller, Daniilidis & Nagel, 1993) determine quantitative system performance based on the number of model lines that are associated correctly with the appropriate image measurements. One weakness of this method is that it provides no indication of the distance between the correctly associated model and measured lines.

By far the class of trackers that rely the most heavily on visual inspection for evaluation purposes are contour trackers. Out of all of the contour trackers surveyed in this review, only a single one undergoes a rigorous, quantitative evaluation. The particular work of note is the cell tracker by Li et al. (Li, Miller, Weiss, Campbell & Kanade, 2006). In many ways, this tracker is very similar to point trackers because numerous very small targets are tracked throughout video sequences. Unsurprisingly, the performance metric employed in this work is similar to those seen in the point tracking literature. Specifically, manually labeled data is used to compute the percentage of cell tracks that are correct for the entire video sequence. In a similar fashion to point trackers, if a mistake is made in just one frame, a cell track is deemed incorrect. For the valid cell tracks, the authors also compute the distance between the manually and automatically computed cell centroids.

Region tracking papers can also be guilty of forgoing a quantitative analysis, but it appears this occurs less often than with contour trackers. A variety of quantitative measures have been proposed in the region tracking literature. For example, some researchers quantitatively test the goodness of their dynamic and system models by adding various levels of noise (Rosales & Sclaroff, 1999) while others will compute the error between the parametric motion estimates and the known ground-truth (Shi & Tomasi, 1994; Wren, Azarbayejani, Darrell & Pentland, 1997). Systems that perform higher level vision tasks, such as action recognition, often compute the number of correctly detected events (Intille, Davis & Bobick, 1997). However, even though this is a quantitative measure of accuracy, it is diluted in terms of providing precise feedback on tracking accuracy. Point-wise template trackers often compute the residual SSD error between the target template and the optimal region found in the current frame (Shi & Tomasi, 1994; Hager & Belhumeur, 1998). Other trackers (Enzweiler, Wildes & Herpers, 2005; Birchfield & Rangarajan, 2005) compute the error between the estimated and manually labeled target positions. Finally, some trackers compute a confidence value that measures the goodness of fit between the template and the current target region (Enzweiler, Wildes & Herpers, 2005; Comaniciu, Ramesh & Meer, 2000; Cannons & Wildes, 2007).

Another region-based metric recently proposed by Collins et al. (Collins, Zhou & Teh, 2005) is that of track percentage. This metric is essentially the number of frames that the target is tracked successfully before it falls off the target, divided by the total number of frames. In order to compute this metric, the meaning of "tracked successfully" must also be defined. In (Collins, Zhou & Teh, 2005), a target is considered to be successfully tracked in a given frame if the measured bounding box overlaps the ground truth bounding box by one or more pixels. This metric for success appears to be very low, given the minimal requirement for bounding box overlap. Another shortcoming of this metric is that two algorithms may obtain an identical score even if one algorithm remains much more centered on the true target than the other. To combat these limitations, in (Collins, Zhou & Teh, 2005), a secondary criterion is proposed that also measures the percentage overlap between the bounding boxes for the frames in which tracking was "successfully" performed.

Since ground truth data is rather sparse in the field of visual tracking, Wu et al. (Wu, Sankaranarayanan & Chellappa, 2007) proposed a method of computing the performance of particle filter-based trackers that does not require any outside information (aside from initialization information, if necessary). Conceptually, the idea is to perform tracking in the forward direction (as per usual) as well as in the backward direction (from the current frame back to the first frame). Upon completion of the backward tracking stage, the posterior distribution at $t = 0$ found using backward tracking is compared against the forward tracker's prior distribution at $t = 0$. It is postulated that if the appearance of the target remains relatively constant and tracking is performed successfully, these distributions should be similar. Since the two distributions are represented as a collection of particle samples, they are compared using the Mahalanobis distance.

7.2 Tracker Comparisons

In this section, we will provide a comparison of the relative merits of each class of tracker. During our discussion, we will highlight which of the visual tracking subproblems have been considered with reference to specific papers in the literature. We will begin with a discussion of discrete feature trackers.

7.2.1 Discrete Feature Trackers

The current status of statistical and deterministic point trackers is actually quite good. Results in (Shafique & Shah, 2003) have shown that state of the art point trackers can maintain an 80% track-based error, even when there are numerous points in the field of view (up to 100). Furthermore, this level of accuracy can be obtained when there are significant levels of noise and points are permitted to enter and leave the scene. With respect to the other tracking subproblems, point-based trackers have been shown to be capable of recovering from occlusions (e.g., (Veenman, Reinders & Backer, 2001; Shafique & Shah, 2003)). To our knowledge, no point trackers have considered the problem of illumination changes. This is partially because most results demonstrated for these trackers involve synthetic or somewhat artificial sequences where target positions can be precisely obtained. Naturally, non-rigid targets and scale changes are not applicable with point trackers. On the other hand, these trackers can definitely track multiple targets (that is essentially the entire problem of point tracking), even when the number of targets in the visual field changes (Salari & Sethi, 1990; Shafique & Shah, 2003). Finally, point trackers can clearly cope with foreground clutter, as the point tracking problem directly involves multiple targets that look identical.

Although the current status of point trackers looks very good, the single and significant problem is that they can only be used to track point targets! Of course, collections of point features can be tracked using point tracking-based techniques. Subsequently, points that are moving with similar velocities can be grouped together to track high level objects. A system similar to this description is proposed in (Smith & Brady, 1995). The problem with such a system is that there are two significant sources of error: (1) Errors associated with effectively and *consistently* extracting the point features; and (2) Errors associated with grouping the points into higher level objects. These errors are apparently significant, because not many works consider this approach, other than (Smith & Brady, 1995).

Other discrete feature trackers, such as those that follow groups of edges and lines, in many ways, have not advanced as far as point trackers. Almost all line tracker works cited in this review have only evaluated their systems in laboratory settings. In (Crowley, Stelmaszyk & Discours, 1988), for example, only the lines of simple three dimensional objects are tracked in front of a constant intensity background. The results displayed in (Deriche & Faugeras, 1991) and (Zhang & Faugeras, 1992) are slightly more complex, taken from indoor laboratory/hallway-type environments.

The work by Zhang and Faugeras (Zhang & Faugeras, 1992) is, to our knowledge, the most complete and impressive example of a line tracking system. Their line tracker is shown to be able to track rigid objects in a cluttered office environment. To track higher level objects, the individual lines are grouped according to their common motion properties. Initialization is automatic in this work as the authors elect to track all lines within the scene. Furthermore, individual lines can be occluded for a short period of time due to EKF framework that is used. Of course, if the dynamics of a line change significantly during occlusion, it will not be properly tracked throughout the occlusion event. The tracker of (Zhang & Faugeras, 1992), as well as other discrete feature trackers, should be fairly robust to illumination changes. These trackers possess this quality because the extraction of discrete features is more robust to illumination then most other features. The system of (Zhang & Faugeras, 1992) can track multiple lines and multiple objects simultaneously, but these objects must be rigid, due to the assumption that lines corresponding to the same object move with similar velocities. Due to the fact that no high level target model is maintained, the proposed line tracker can naturally accommodate target scale changes without any additional mechanisms. In fact, in (Zhang & Faugeras, 1992), one of the sequences considers a forward moving camera toward stationary collection of targets.

Thus, line trackers appear to have numerous positives in terms of being able to deal with many of the subproblems of tracking. The problem with these trackers is that high levels of performance have only been demonstrated in fairly simple laboratory environments. Presumably, experimental evaluation has been limited to these environments because these systems do not respond well when used with realistic scenery. More specifically, even though the line tracker of (Zhang & Faugeras, 1992) was shown that it can operate when the laboratory background is cluttered, outdoor videos will most likely have even more complex and time varying backgrounds. Line trackers rely heavily on accurate edge detection and line grouping techniques. These processes become increasingly difficult and error prone when the amount of clutter increases. The process of grouping lines into higher level objects based on their velocities (Zhang & Faugeras, 1992) will also undoubtedly experience greater errors in more challenging sequences and will fail altogether if multiple targets are moving with similar velocities in confined spaces. Finally, a significant limitation of this approach is that the targets to be tracked must be well-represented by a collection of lines. Although many inanimate objects meet this criterion (e.g., boxes, televisions), many other animate and inanimate objects do not (e.g., humans, animals, balls). Since the work of Zhang and Faugeras, very little research appears to have been completed in the area of line tracking; a fact that probably indicates that Zhang and Faugeras advanced the field as far as obviously possible.

Trackers that make use of 3D models have also seen less research in recent years. Nonetheless, these trackers have demonstrated some success when it comes to tackling some of the subproblems in visual tracking. In particular, in (Lowe, 1991) the tracker was shown to be capable of dealing with partial occlusions when tracking a non-rigid object. Also, the target box was tracked despite the fact that the author's hands often occluded significant portions of it. Furthermore, in both (Lowe, 1991) and (Koller, Daniilidis & Nagel, 1993), the respective trackers achieved success despite somewhat challenging illumination conditions. In (Lowe, 1991) target tracking is demonstrated in only a laboratory-type environment, but the author claims that the system attained good results over a wide-range of lighting conditions. On the other hand, in (Koller, Daniilidis & Nagel, 1993), cars are followed in surveillance-style outdoor footage where lighting conditions are variable and shadows are present.

Not unlike line trackers, 3D model-based trackers can naturally accommodate target changes of scale. Specifically, when a target changes scale in the image domain, it is typically due to relative motion away from or toward the camera. Thus, changes in scale with 3D model-based trackers simply correspond to different transformations from the model to image coordinate frame. Qualitative evidence of tracking throughout scale changes has been presented in (Koller, Daniilidis & Nagel, 1993; Vacchetti, Lepetit & Fua, 2004). In (Koller, Daniilidis & Nagel, 1993), qualitatively successful tracks are shown for tracking cars throughout larger changes in depth. Furthermore, in (Vacchetti, Lepetit & Fua, 2004), stationary targets in a hallway are tracked as the camera moves forward.

To our knowledge, it appears that of all the trackers that incorporate 3D models, only the one in (Koller, Daniilidis & Nagel, 1993) proposed a method of dealing with the subproblem of automatic initialization. This system made use of a motion segmentation stage to get a coarse estimate of the initial position and orientation of a car. The proposed automated initialization scheme is reported to work, but only in limited situations. There are undoubtedly a significant number of local minima when attempting to optimize the model parameters that bring the projection into alignment with image data. Accordingly, 3D model-based systems have historically been quite sensitive in relying on initial estimates that are quite close to the true optimum. In a similar vein, it would be expected that 3D model-based trackers would not fare well in the case of total occlusions (since good initial estimates are required for every frame). However, with the recent 3D model-based systems that track by continuous detection (Leng & Wang, 2004; Lepetit, Lagger & Fua, 2005), the problems of occlusion and initialization should be less of an issue. An area of 3D model-based tracking that appears to not have been studied is that of tracking multiple targets simultaneously.

The ability of 3D model-based trackers to handle significant levels of background clutter is also suspect and has not been adequately studied in the literature. Most works in this field only consider laboratory environments (Lowe, 1991; Gennery, 1992; Verghese, Gale & Dyer, 1990). In Lowe's defense, there is some clutter in the background provided by his clothing and background scenery. However, in most outdoor scenery, we would expect to see higher levels of clutter. Although the car tracker of (Koller, Daniilidis & Nagel, 1993) operates outdoors, there is little clutter present on the asphalt that can disturb the tracker. More recent work by Vachettit et al. (Vacchetti, Lepetit & Fua, 2004) attempts to illustrate robustness to background clutter by tracking an object in front of a checkerboard background. This result is interesting as a demonstration, but it is somewhat contrived. It appears that, in significant levels of clutter, feature-based 3D model trackers may not fair well. These systems depend on the successful and error-free extraction of target edges and lines, processes that develop greater errors as the amount of clutter increases. In very complex environments where the clutter has similar statistics and is spatially close to the target, 3D model-based approaches will most likely fail. In such examples, it is challenging for these trackers to be able to separate signal from noise and the optimization space contains numerous local minima. More complex techniques such as the use of robust estimators (as was done in (Vacchetti, Lepetit & Fua, 2004)) or random sample consensus (RANSAC) (Fischler & Bolles, 1981) may allow 3D model-based systems to perform reasonably well in somewhat cluttered environments. Presently, however, it appears that no convincing demonstration of feature-based 3D model tracking has been documented in the literature.

7.2.2 Contour Trackers

It is no secret that contour trackers have come a long way since their original inception in 1988 (Kass, Witkin & Terzopoulos, 1988). In the beginning, it appears that the authors did not really view snakes as much more than an interactive tool that could be used effectively in tandem with a human operator for determining precise boundaries of targets. Currently, these trackers are far from interactive and provide results that can be qualitatively quite impressive.

The initial snake contours had a number of limitations that hindered their utility. For example, to our knowledge, they all required automatic initialization and could not easily or naturally track multiple targets. Although the original snake formulations (Kass, Witkin & Terzopoulos, 1988; Terzopoulos & Szeliski, 1992; Leymarie & Levine, 1993) could not handle occlusions, Peterfreund's velocity snake tracker (Peterfreund, 1999) did make some strides for detecting occlusion events. In fact, one example is illustrated whereby a target is almost entirely occluded. The system recognizes the occlusion and is capable of tracking the target after the event has occurred. The example is somewhat artificial in that the target essentially remains stationary throughout the entire video. However, the detection of the portions of the contour that no longer represent the true target during occlusion events appears to be fairly reliable, qualitatively. Thus, some simplified occlusion handling is present within the snake literature. Even still, we would expect that more complex occlusion events would prove to be problematic for Peterfreund's velocity snake tracker.

The introduction of level set methods certainly revitalized the field of contour tracking. Unlike their snake counterparts, it has been demonstrated that level-set methods are much more amenable to automated initialization (Paragios & Deriche, 2000) given that the right assumptions hold. Specifically, in (Paragios & Deriche, 2000), the authors assume that the target(s) of interest are the only moving objects in the scene and that there is no significant clutter immediately surrounding the object. In such situations, contour evolution for the initial and subsequent frames can first be performed on a foreground map, which will essentially attract the contour of any arbitrary size toward a rough outline of the target. Of course, the introduction of a moving camera or distractor moving objects would cause this automatic initialization scheme to struggle. Yet, essentially all currently available automatic initialization methods only operate correctly when certain assumptions are met.

Level set contours have also allowed for more straightforward tracking of multiple targets simultaneously. As discussed in Chapter 4, this ability arises due to the fact that level set contour trackers naturally allow topological changes. However, one should not be led to believe that this ability is a panacea for multiple target tracking. The contour tracking literature seldom discusses the fact that the contour itself has no notion of the identity of individual targets or the number of targets that are being tracked. Typical contour-based trackers just output the final contour region, which is the local optimum of a particular objective function. In simple, single target tracking situations, this "providing of a bounding contour" is essentially visual target tracking. However, to perform visual tracking of multiple targets simultaneously, one must keep track of the identity of each target and record its position within the scene for every frame. Level set contour trackers, even with their ability to handle topological changes, usually do not meet this standard. Specifically, providing a bounding contour between foreground regions and the background is not the same as recording the identity and position of each target throughout all video frames. Essentially, the end result is that to perform multiple target tracking, level set contour methods still require some mechanism to provide data association between frames.

Of all the level set contour trackers considered in this review, only one made use of data association techniques — the work in (Li, Miller, Weiss, Campbell & Kanade, 2006). In this paper, a bank of Kalman filters is used to track a number of individual cells. Although simple Kalman filters are used in this work, more advanced statistical data association tools could also be used (e.g., JPDAF). The problem is that statistical data association methods were originally developed to deal with point targets, not contours. Thus, the problem is to determine a method of using these statistical tools within the contour-tracking framework. The method employed in (Li, Miller, Weiss, Campbell & Kanade, 2006) is to extract the centroid position of targets to represent them as points. Although this approach was successful in the application domain of tracking small, roughly circular cells, in general, the problem of obtaining accurate centroid positions for large targets is an error-prone problem. As a result, the predictive filtering and data association mechanisms would be challenged by this extremely noisy centroid data. Furthermore, centroid positional measurements will typically become worse when an object becomes partially occluded; exactly the time when data association tools need to be operating at their best.

In several of the previous contour works, the trackers were not particularly robust to clutter. This limitation was partially due to the fact that early contour trackers only made use of intensity edge measurements. With the advancement of new techniques and the gradual inclusion of regional-based descriptors, it appears these systems have become somewhat more resilient to clutter (Mansouri, 2002; Yilmaz, Lin & Shah, 2004). In particular, (Mansouri, 2002) shows one impressive sequence (the manege sequence) showing a rotating carousal. Although the target selected appears to have a distinctive color and intensity properties, it is successfully tracked, despite significant levels of clutter in the scene. Perhaps the most famous example of tracking in clutter is the CONDENSATION algorithm (Isard & Blake, 1996). In this case, it is the inclusion of multiple hypotheses through the use of particle filtering that allows for success. However, to achieve the demonstrated level of tracking, Isard and Blake required copious amounts of system training.

It appears that the problem of illumination invariance has not been overly well-studied in the context of contour trackers. Intensity gradients presumably should be fairly invariant to illumination changes, whereas some of the more regional-based descriptors being introduced recently will be less so. One aspect of illumination that has been studied, is that of shadows. The systems in (Bertalmio, Sapiro & Randall, 2000) and (Paragios & Deriche, 2000) were both distracted into including shadow regions as part of the tracked object, due to the systems' reliance on intensity edges.

In addition to the occlusion handling methods proposed for snake-based trackers in (Peterfreund, 1999), in (Yilmaz, Lin & Shah, 2004) an occlusion handling scheme has been mentioned for level set contour trackers. Qualitatively, the system's ability to deal with target partial and near-full occlusions appear to be acceptable. However, the approach employed is rather heuristic and we believe that more rigorous data association and occlusion handling techniques are necessary in general situations. Thus, it appears the area of occlusion handling for contour trackers could use additional study.

Contour trackers, in general, have a distinct advantage over other types of trackers as they can typically track non-rigid targets that change in scale in a very natural fashion. So long as the non-rigid deformations and scale changes of the target are sufficiently slow (i.e. on the order of the target motion), most contour trackers can still retain accurate target delineation. Of course, contour trackers are not perfect when it comes to scale changes and non-rigid targets. For instance, snake contours will not be able to properly track non-rigid target deformations that cause deep and sharp peaks and valleys to form on the target boundary.

Although the qualitative results of contour-based trackers have progressed significantly since the work of (Kass, Witkin & Terzopoulos, 1988), quantitative evaluation in the field is extremely sparse. As a result, very little appears to be known in regards to the quantitative performance analysis of contour trackers. Limited quantitative analyses are provided in (Isard & Blake, 1996) and (Paragios & Deriche, 2000). In (Isard & Blake, 1996) the spatiotemporal coordinates of a target are plotted on XYT axis for the proposed CONDENSATION tracker and a simple Kalman filter tracker. From the provided figure, it is clear that the track provided by the CONSENDATION algorithm is superior. The authors also note that the proposed algorithm successfully tracks the target position throughout the sequence. However, no other quantitative details are provided to substantiate this claim or precisely define the accuracy. In (Paragios & Deriche, 2000), quantitative tracking results are not shown, but the run-time of the proposed Hermes surface updating algorithm is numerically compared against the standard narrow band and fast marching methods. As mentioned previously, the one notable exception to the lack of rigorous quantitative analysis in the contour tracking literature is the work by Li et al. (Li, Miller, Weiss, Campbell & Kanade, 2006).

To conclude this section, we must make note of one of the primary benefits of contour trackers over other classes of trackers. Specifically, contour trackers provide an exact outline of the target(s) being tracked without making use of any type of 3D model. Discrete feature, in general, do not yield object outlines that are as accurate as those obtained with contour trackers. Additionally, most regional trackers do not output accurate target delineations either. Thus, if obtaining very accurate outlines of non-rigid, scale-varying, targets is the top priority, a contour tracker is probably the best choice.

7.2.3 Region Trackers

It appears that among all the papers surveyed in this review, at least one region-based tracker has experienced some level of success in all of the abovementioned subproblems of visual tracking. Yet, at the same time, it appears that no tracker has been created that simultaneously and robustly provides a solution to each of the subproblems. We will now discuss specific region trackers that contribute to the subproblems within visual tracking, starting with automatic initialization.

Although manual initialization is popular within the realm of region trackers, automatic methods have been successfully presented. For example, (Enzweiler, Wildes & Herpers, 2005) demonstrates automatic initialization by detecting regions that exhibit coherent motion energies, (Okuma, Taleghani, Freitas, Little & Lowe, 2004) uses an Adaboost-based target spotting approach, and (Wren, Azarbayejani, Darrell & Pentland, 1997; Stauffer & Grimson, 2000) make use of background modeling approaches. As is the case with almost all automatic initialization schemes, certain assumptions are made regarding the targets of interest in order for these schemes to operate effectively. For example, in (Enzweiler, Wildes & Herpers, 2005) it is assumed that the targets are moving in a coherent fashion — a fairly reasonable assumption when considering the application of target tracking.

With respect to partial and full occlusion, several region-based works have examined this problem. Some of the most robust region-based trackers to partial and full occlusion are those that make use of the mean shift tracking paradigm (Comaniciu, Ramesh & Meer, 2000; Cannons & Wildes, 2007). Mean shift trackers collapse all spatial information across the target support and represent the target of interest as a histogram. Thus, if a portion of the target is occluded, the overall effect on the target histogram is marginal, often resulting in continual successful tracking. Note that this result only holds true so long as the unoccluded portion of the target generates a histogram that is similar to the template. On the other hand, if multiple objects are present within the scene that have a similar feature-based appearance (e.g., color), then a mean shift tracker could easily be fooled into tracking the the wrong target when an occlusion event occurs. Adding a predictive filtering mechanism, as was done in (Nummiaro, Koller-Meier & Gool, 2002), would help to reduce the chance of errors during partial or full or occlusions. Another approach, taken in (Cannons & Wildes, 2007) is to further enrich the target representation to reduce the chance of confusing multiple foreground targets. Results are demonstrated in this work where a target is tracked successfully, even when it is fully occluded by a target that has almost an identical color distribution.

Point-wise template trackers are typically less robust to partial and full occlusion than kernel histogram methods. However, even within the warping framework, occlusion handling solutions have been proposed. In (Hager & Belhumeur, 1998) an explicit occlusion-handling mechanism is introduced that identifies pixels that have large residuals during the warping process. These pixels are assumed to be occluding the true target and are removed from the motion estimation process. In (Jepson, Fleet & El-Maraghi, 2003), robustness to significant levels of occlusion is demonstrated by using a unique target representation and updating scheme that considers long-term aspects of the target as well as those aspects that are changing rapidly.

Illumination is a significant problem that has been considered in the realm of region-based trackers as well. Unlike 3D model trackers, point trackers, line trackers, and contour trackers that often rely on discrete features and image edges, most region tracker use intensity or color features that are notorious for being sensitive to changes in illumination. Some kernel histogram trackers have proposed making use of color spaces that are more robust to illumination, such as normalized RGB or HSV (Hager, Dewan & Stewart, 2004; Birchfield & Rangarajan, 2005). Such a target representation is argued to improve robustness to illumination changes. However, to our knowledge, no rigorous quantitative study has been completed to show the exact benefits of using various color spaces with respect to tracking accuracy. The spatiotemporal oriented energy feature set proposed in (Cannons & Wildes, 2007) not only provides a rich description of a target, but it also provides an representation that is quite robust to illumination changes. Results are illustrated where an individual walks in and out of very dark and very bright image regions.

With respect to warping trackers, the problem of illumination has been touched upon as well (Hager & Belhumeur, 1998; Avidan, 2001; Jepson, Fleet & El-Maraghi, 2003). In (Hager & Belhumeur, 1998), the basis set representation of the target is captured under a number of illumination conditions. There are, however, some limitations to this approach. To begin, one has to capture training samples of the target under a number of varying illumination conditions — a time consuming process. Additionally, in the test sequences, illumination conditions that cannot be represented as a linear combination of the basis vectors cannot be properly modeled. The SVM warping tracker proposed by Avidan (Avidan, 2001) also claims to be more robust to illumination changes than a standard SSD template tracker. The robustness to illumination is gained due to the machine learning classification nature of the matching metric. Nonetheless, the authors admit that the approach is not overly robust to illumination changes, just more so than SSD trackers. In (Jepson, Fleet & El-Maraghi, 2003), sequences were also demonstrated where the target walked through shadow regions. The spatially orientation selective filters are shown to be somewhat robust to illumination changes as tracking is successfully performed throughout these sequences.

Region trackers have also been demonstrated to be capable of tracking multiple targets simultaneously. In the case of monocular camera tracking, solutions are provided in (Okuma, Taleghani, Freitas, Little & Lowe, 2004; Stauffer & Grimson, 2000; Enzweiler, Wildes & Herpers, 2005). In (Enzweiler, Wildes & Herpers, 2005), a simple PETS video sequence is shown where two individuals are tracked. However, multi-target tracking is not the focus of this work and it appears that two warping-based trackers were simply instantiated within the scene to track the two persons. Since no predictive filtering or data association techniques are employed, the method will most likely fail during occlusions or other more complex multi-target situations. In (Stauffer & Grimson, 2000), a multiple hypothesis tracker is used to follow multiple agents within the scene. From the results presented, it is hard to determine how accurately the multiple targets are tracked, but the authors note that data association is not extremely robust during occlusion or crossing events. Finally, in (Okuma, Taleghani, Freitas, Little & Lowe, 2004), a tracker that was designed for the purpose of following multiple targets using a particle filter framework is presented and the results are quite impressive. Qualitatively, it appears the tracker has the ability to follow multiple targets robustly, even though the targets share similar appearance, occlude one another, and move in close proximity. Unfortunately, no quantitative results are reported for this work.

Other types of region trackers that make use of specialized hardware and/or prior knowledge about the camera have also illustrated the capability of tracking multiple targets simultaneously. For example, the M_2 tracker (Mittal & Davis, 2003) has been shown to be capable of tracking up to 6 people simultaneously in a confined environment. To achieve this level of performance, an array of as many as 16 cameras is required, because any single camera experiences dramatic occlusions. The tracker in (Zhao & Nevatia, 2004) appears to be very impressive — tracking multiple humans when they walk in groups and occlude one another in surveillance-type sequences. Quantitative results for this tracker show very low error rates as well. The only drawback of the system in (Zhao & Nevatia, 2004) is that prior knowledge of the camera model and the ground plane are needed. Beymer et al. also illustrate the tracking of two or three people in a narrow hallway using Kalman filters and stereo cameras (Beymer & Konolige, 1999). The stereo camera provides the significant benefit of being able to resolve many partial occlusions through the use of depth information.

The problem of clutter has also been considered in the region tracking framework. Blob trackers can handle clutter very easily so long as the clutter is in the background and stationary (e.g., (Wren, Azarbayejani, Darrell & Pentland, 1997; Stauffer & Grimson, 2000)). Successful results for blob trackers have also been shown for moving cameras (e.g. (Yin & Collins, 2007a)). Point-wise template trackers have been shown to be capable of following targets of interest throughout complex environments, as long as occlusion is not significant (e.g., (Enzweiler, Wildes & Herpers, 2005; Jepson, Fleet & El-Maraghi, 2003; Black & Jepson, 1998)). Finally, the histogram representation used in mean shift trackers is resilient to clutter due to the histogram representation and kernel weighting. Several mean shift systems have illustrated that tracking in cluttered environments is possible (Comaniciu, Ramesh & Meer, 2000, 2003; Collins & Liu, 2003; Birchfield & Rangarajan, 2005; Leung & Gong, 2006; Cannons & Wildes, 2007)).

With respect to the tracking of non-rigid targets, many region trackers have illustrated this capability, with the emphasis being on the tracking of humans (e.g., (Black & Jepson, 1998; Cham & Rehg, 1999; Okuma, Taleghani, Freitas, Little & Lowe, 2004; Beymer & Konolige, 1999; Nguyen, Worring & Boomgaard, 2001; Isard & MacCormick, 2001; Comaniciu, Ramesh & Meer, 2003; Collins, 2003; Collins & Liu, 2003; Mittal & Davis, 2003; Zhao & Nevatia, 2004; Enzweiler, Wildes & Herpers, 2005; Cannons & Wildes, 2007)). Specifically, histogram-based trackers usually have success when tracking non-rigid objects as many of these deformations only cause minor changes in the overall histograms. Finally, scale changes is a problem that is naturally handled by most parametric warping-based trackers (e.g., affine warping trackers). Results have been shown for many warping trackers where scaling effects have been substantial (e.g., (Jepson, Fleet & El-Maraghi, 2003; Enzweiler, Wildes & Herpers, 2005)). In the original mean shift tracker, scale changes were handled unintelligently by using a brute-force method of repeating optimization with multiple tracking windows of different sizes. As reported in (Collins, 2003), this method prevents the tracking crop box from growing too large, but does limit the tracker from representing the target with a window that is too small. Accordingly, a superior approach is proposed in (Collins, 2003) where mean shift optimization is performed in scale space, with results demonstrating the utility of the approach.

7.3 Summary Discussion

Table 7.1 provides a summary of the above comparisons and references particular works for each category of tracker that have made significant progress on the various subproblems in tracking. In our opinion, such a table is essentially one of the only ways that the huge variety of trackers can be quickly compared, at present. It should be noted that if a tracker is listed in a cell, it does not mean that the tracker has completely "solved" the corresponding subproblem. Rather, the entries indicate that a particular tracker has demonstrated significant progress or a partial solution to a problem.

Using Table 7.1, we will now consider the current status of each of the classes of trackers, in a qualitative sense. Specifically, we would like to determine how robust of a tracker could be created if one were permitted to literally pick and choose ideas from the various trackers within a single category. To begin, let us consider 3D model-based trackers. If we were to create a tracker that borrowed ideas from previous works, we could theoretically create a tracker that is robust to most of the subproblems. In particular: (Koller, Daniilidis & Nagel, 1993) provides an adequate method for performing automatic initialization; continuous detection schemes provide an effective means of dealing with occlusion (Leng & Wang, 2004; Lepetit, Lagger & Fua, 2005); the vehicle tracker in (Koller, Daniilidis & Nagel, 1993) was shown to be somewhat robust to outdoor illumination changes; in (Lepetit, Lagger & Fua, 2005), a tracker was demonstrated to be capable of following a target in a semi-cluttered scene; the system in (Lowe, 1991) dealt with simple, non-rigid motion; and scale changes are handled naturally. Notably absent is the ability of tracking multiple targets simultaneously. To our knowledge, 3D model-based trackers have not been demonstrated to be capable of fulfilling such a task. Using data association techniques, it should be possible to follow multiple target with a 3D model-based tracker, but computation time might be problematic. Additionally, 3D model-based trackers are typically demonstrated on a restricted class of objects. These objects are amenable to modeling via highly structured geometric relationships between discrete features (e.g., many man-made objects). These trackers have not been thoroughly tested on more "natural" objects (e.g., biotic entities).

Another weak point of our proposed "super" tracker might be the ability to cope with significant non-rigid deformations. Lowe's system (Lowe, 1991) was demonstrated to be effective for simple non-rigid target deformations. However, as the number of allowable deformations increases, so to does the number of parameters that must be estimated. The inclusion of additional parameters will undoubtedly make the estimates noisier and require additional processing time, most likely making complex non-rigid transformations impractical. To summarize, presently a "super" 3D model tracker could be created from previous works that:

- Has automatic initialization capabilities
- Can deal with occlusions, limited clutter, simple non-rigid deformations, and scale changes

In a similar manner, a "super" contour-based tracker could be created as well. In this case, to construct our tracker, we might select: the automatic initialization procedure shown in (Paragios & Deriche, 2000); the partial occlusion handling mechanisms of (Yilmaz, Lin & Shah, 2004); the multiple target tracking abilities presented in (Li, Miller, Weiss, Campbell & Kanade, 2006); and tools used to cope with clutter as seen in (Isard & Blake, 1996) or (Yilmaz, Lin & Shah, 2004). In our opinion, the weak points of such a "super" tracker would be the initialization scheme, the occlusion handling ability, and the inherent limitation of relying on a contour that delineates the target boundary. Specifically, the initialization method proposed in (Paragios & Deriche, 2000) required fairly uniform backgrounds and although qualitatively very good, the occlusion handling mechanism of (Yilmaz, Lin & Shah, 2004) requires additional rigorous testing. In (Li, Miller, Weiss, Campbell & Kanade, 2006) a bank of Kalman filters was used for data association; however, it appears that more complex techniques could be used, as appropriate. Additionally, although illumination variance has not been well-studied with respect to contour tracking, given that many contour trackers rely on gradient information (fairly robust to illumination changes), it is likely that illumination changes could be overcome. Finally, by construction, contour trackers rely on the assumption that the target is well-defined by a bounding contour. This assumption is certainly not always true. For example, in camouflage situations a delineating contour may not be visible (even to humans), making other target properties (e.g., motion) a more salient cue for tracking.

Thus, currently, a "super" contour tracker could feasibly be created from prior systems that:

- Provides automatic initialization in simple scenes
- Probably accommodates partial occlusions

- Probably is robust to limited illumination changes
- Allows for multiple targets to be tracked in simple situations
- Can track throughout clutter, non-rigid deformations, and scale changes

Finally, we consider the thought experiment of constructing a “super” region tracker from previously proposed systems. In this case, we might elect to choose: the automatic initialization systems provided by (Okuma, Taleghani, Freitas, Little & Lowe, 2004) or (Enzweiler, Wildes & Herpers, 2005); the occlusion handling abilities of (Jepson, Fleet & El-Maraghi, 2003) or (Comaniciu, Ramesh & Meer, 2000); the robustness to illumination changes provided in (Cannons & Wildes, 2007); the ability to track multiple targets provided by (Okuma, Taleghani, Freitas, Little & Lowe, 2004); the robustness to clutter as seen in (Comaniciu, Ramesh & Meer, 2000) or (Cannons & Wildes, 2007); the ability to track non-rigid target provided by (Comaniciu, Ramesh & Meer, 2000); and the capability of tracking targets as they change in scale as seen in (Jepson, Fleet & El-Maraghi, 2003) or (Collins, 2003). The above algorithms were selected as they currently provide the most convincing qualitative evidence of being able to solve the defined subproblems of visual tracking.

Of the three categories of trackers, the region-based “super” system appears to have the fewest shortcomings. Moreover, it appears that there are a greater number of reasonable possibilities that could be employed for solving each of the subproblems. One weakness of the combined region tracker would arguably be its occlusion handling capabilities. Systems (e.g., (Comaniciu, Ramesh & Meer, 2000; Jepson, Fleet & El-Maraghi, 2003)) have demonstrated good results when tracking through partial occlusions, but successful tracking through full occlusions has not been convincingly demonstrated. Furthermore, region trackers are not effective at delineating the exact target boundary. Nonetheless, a “super” region tracker could be constructed that theoretically has the ability to overcome most of the defined subproblems of tracking.

Unlike the other categories of trackers, there is less of a unifying framework for the systems described in Chapter 6. Nonetheless, a similar analysis of constructing a “super” tracker can still be repeated for these unique trackers. In theory, a combined “super” tracker should provide performance at least as good as any of the previously considered “super” trackers. Specifically, individual trackers found in Chapters 3 - 5 can be considered to be a subset of the combined trackers (where a single tracker is combined with a non-existent second tracker). Presently, however, we will strictly consider combined trackers that have already been presented in the literature and are described in Chapter 6 of this review. In many ways, this restriction places the combined trackers at a disadvantage as relatively few were discussed when compared to the other chapters.

When constructing a “super” combined tracker, we might select: the occlusion handling abilities of (Yang, Wu & Hua, 2008); the tracking of multiple targets provided by (Rasmussen & Hager, 2001); the robustness to clutter, as demonstrated in (Rasmussen & Hager, 2001); the ability to track non-rigid target provided by (Shao, Porikli & Chellappa, 2007); and the capability of tracking targets as they change in scale as seen in (Yang, Wu & Hua, 2008). These algorithms demonstrate excellent performance with respect to these subproblems of visual tracking. It should be noted that no combined systems considered in this review have demonstrated high-quality results with respect to the problems of automatic initialization and robustness to illumination changes. However, as noted previously, solutions to these problems could be borrowed from other tracking paradigms (e.g., region trackers). Not unlike region trackers, perhaps a weak point of the "super" combined tracker would be tracking through occlusions. However, the system in (Yang, Wu & Hua, 2008) has demonstrated excellent results with respect to this problem by using surrounding scene information during tracking. However, further rigorous testing is required before the full utility of the approach in (Yang, Wu & Hua, 2008) can be stated with certainty. Combined trackers have another advantage over region-based trackers in that they can accurately delineate target outlines by combining region-based and contour-based methods (e.g., (Shao, Porikli & Chellappa, 2007)). These apparent advantages do, however, come at a cost — complexity and computation time. The combination of multiple trackers makes the overall system more complicated and requires more computational resources. There is also the matter of determining how to optimally combine multiple tracking paradigms in a simple, principled manner. This problem has yet to be fully explored by researchers.

Given the currently accepted practices with regards to evaluating tracking algorithms, it is quite challenging to provide any meaningful quantitative summary discussion regarding the various tracking classes. Perhaps one of the only quantitative values that we could use (i.e. that is fairly consistently reported) is the run-time of the various algorithms. However, even this metric would be challenging to compare fairly, given that the algorithms were deployed on different hardware and in a variety of programming languages. With respect to this review, runtime has not been a particular point of emphasis. Accordingly, we will not perform such a quantitative analysis on runtime.

To summarize, arguably the region-based trackers have progressed the furthest in terms of solving the qualitative subproblems that we have defined for the field of visual tracking. However, even after performing this summary analysis, it is still unclear as to how meaningful it is without a means of performing more rigorous quantitative comparisons. As mentioned, some datasets are available for testing trackers (e.g., PETS), but the primary focus is often on applications other than visual tracking. If and when a more complete visual tracking dataset is created, the field will be able to ascertain the current status much more accurately. At that time, it might also become apparent that the field has not progressed quite as far is it appears. Until such a dataset is made available, our best efforts of comparison are limited to qualitative summary tables, such as that seen in 7.1.

Table 7.1: Summary of Success on Visual Tracking Sub-Problems

Problem	Discrete Feature Trackers	Contour Trackers	Region Trackers
Initialization	Most Point Trackers Line: (Zhang & Faugeras, 1992) 3D-Model: (Koller, Daniilidis & Nagel, 1993)	(Paragios & Deriche, 2000)	Blob: (Wren, Azarbayejani, Darrell & Pentland, 1997), (Stauffer & Grimson, 2000) Point-Wise: (Enzweiler, Wildes & Herpers, 2005) Others: (Okuma, Taleghani, Freitas, Little & Lowe, 2004)
Occlusion	Line: (Zhang & Faugeras, 1992) 3D-Model: (Lowe, 1991), (Lepetit, Lagger & Fua, 2005)	(Peterfreund, 1999), (Yilmaz, Lin & Shah, 2004)	Point-Wise: (Hager & Belhumeur, 1998), (Jepson, Fleet & El-Maraghi, 2003), (Beymer & Konolige, 1999) Kernel: (Comaniciu, Ramesh & Meer, 2000), (Cannons & Wildes, 2007)
Illumination	Most Point Trackers Line: (Deriche & Faugeras, 1991), (Zhang & Faugeras, 1992) 3D-Model: (Koller, Daniilidis & Nagel, 1993)	(Chen, Rui & Huang, 2001)	Point-Wise: (Hager & Belhumeur, 1998), (Jepson, Fleet & El-Maraghi, 2003), (Ross, Lim & Lin, 2008) Kernel: (Hager, Dewan & Stewart, 2004), (Cannons & Wildes, 2007)
Multiple Targets	Most Point Trackers Line: (Zhang & Faugeras, 1992)	(Li, Miller, Weiss, Campbell & Kanade, 2006)	Blob: (Stauffer & Grimson, 2000) Others: (Beymer & Konolige, 1999; Mittal & Davis, 2003), (Mittal & Davis, 2003), (Okuma, Taleghani, Freitas, Little & Lowe, 2004), (Zhao & Nevatia, 2004)
Clutter	Line: (Zhang & Faugeras, 1992) 3D-Model: (Lowe, 1991), (Vacchetti, Lepetit & Fua, 2004), (Lepetit, Lagger & Fua, 2005)	(Isard & Blake, 1996), (Mansouri, 2002), (Yilmaz, Lin & Shah, 2004), (Peterfreund, 1999)	Blob: (Wren, Azarbayejani, Darrell & Pentland, 1997), (Stauffer & Grimson, 2000), (Yin & Collins, 2007a) Point-Wise: (Black & Jepson, 1998), (Enzweiler, Wildes & Herpers, 2005), (Jepson, Fleet & El-Maraghi, 2003) Kernel: (Comaniciu, Ramesh & Meer, 2000), (Collins & Liu, 2003), (Birchfield & Rangarajan, 2005), (Leung & Gong, 2006), (Cannons & Wildes, 2007)
Non-Rigid	3D-Model: (Lowe, 1991)	Most	Point-Wise: (Black & Jepson, 1998), (Beymer & Konolige, 1999), (Cham & Rehg, 1999), (Nguyen, Worring & Boomgaard, 2001), (Enzweiler, Wildes & Herpers, 2005) Kernel: (Comaniciu, Ramesh & Meer, 2003), (Collins, 2003), (Collins & Liu, 2003), (Cannons & Wildes, 2007) Others: (Isard & MacCormick, 2001), (Mittal & Davis, 2003), (Zhao & Nevatia, 2004), (Okuma, Taleghani, Freitas, Little & Lowe, 2004)
Scale	Line: (Zhang & Faugeras, 1992) 3D Model: (Koller, Daniilidis & Nagel, 1993), (Vacchetti, Lepetit & Fua, 2004)	Most	Point-Wise: (Jepson, Fleet & El-Maraghi, 2003), (Enzweiler, Wildes & Herpers, 2005), Kernel: (Collins, 2003)

Chapter 8

Open Problems

To conclude this review on visual tracking, we will discuss some of the most significant open problems in the field of monocular visual tracking. As evidenced by the table shown at the conclusion of the previous chapter, improvements upon subproblems such as occlusion, multi-target tracking, initialization, etc. are still needed. Trackers that enhance performance, and especially, consistency and robustness are welcomed and needed. Rather than dwell on these issues that have already been identified and discussed previously, in this chapter we will focus on more significant overridding problems within the field of visual tracking. These are problems that we feel must be resolved to truly advance visual tracking to the next level and allow the more widespread adoption of such systems outside of academia. Furthermore, strides in these areas also will positively impact the various subproblems delineated in Table 7.1.

8.1 Quantitative Evaluation

As alluded to previously in this review, one aspect of visual tracking that is embarrassingly lacking is a means of quantitatively computing the performance of proposed tracking systems. An effective means of evaluating tracking algorithms would be the creation of a large, standard database of video sequences and ground truth labeling information. An ideal database would have a number of desirable properties. To begin, it would consists of numerous video sequences. These sequences would be divided into training and testing sets in the event that learning based trackers are to be tested. Another important aspect of the database is that it must be updated with new sequences on a regular basis. Should the database remain unchanged for a great number of years, there is a good chance that researchers will begin to "overfit" their algorithms to provide optimal performance on the database test set, rather than creating well-rounded trackers. Perhaps the most important quality of a good database is that it should systematically vary the tracking conditions (i.e. individually test on changes in illumination, changes in scale, occlusions, etc.). By creating a dataset where each each parameter is varied in turn, researchers can determine the relative strengths and weaknesses of proposed algorithms with respect to each subproblem of visual tracking.

Other fields, such as face recognition and stereo vision have large, standard databases that are almost always employed when researchers are presenting their work. For example, the face recognition technology (FERET) database (Phillips, Moon, Rizvi & Rauss, 2000) was developed by the United State's Department of Defense. This database consists of faces from approximately 1,200 different individuals with a total of over 14,000 images. Just as importantly, the database explicitly separates images for testing performance under varying lighting conditions, when the subjects wear glasses, and when significant time has passed between the acquisition dates of the images. Explicit techniques have also been detailed for evaluating the performance of new systems on this data. Prior to the introduction of the FERET database, the field of face recognition was not unlike that of visual tracking. In particular, several different researchers, including Alex Pentland (Pentland, Moghaddam & Starner, 1994) and Joseph Wilder (Wilder, Mammone, Meer, Flanagan, Xu, Tsai, Weiner & Zhang, 1994) had created their own databases for evaluation. These databases did not contain nearly as many images as the FERET database. Furthermore, when several different sources of data are available, researchers become divided on which one to use, hampering fair evaluation. The FERET database brought a unifying and standard method of algorithm evaluation. Interestingly, face recognition systems are becoming more popular in the commercial domain.

The success and importance of standard databases is not just limited to the field of face recognition. Stereo vision researchers make use of the standard Middlebury dataset (Scharstein & Szeliski, 2002) when evaluating their algorithms. Although the databases are not typically overly large, they do provide ground-truth disparity information. Furthermore, new data is made available every few years in an attempt to prevent researchers for fine-tuning their algorithms for specific imagery. In addition to making the data available, algorithm rankings can also be found on-line. Furthermore, results of new algorithms (in the form of disparity maps) can be submitted on-line to the researchers that maintain the Middlebury database. The submitted disparity maps are evaluated in an objective manner and subsequently, the authors are informed of the performance of their algorithms.

Another database in the field of computer vision is Berkeley's segmentation data (Martin, Fowlkes, Tal & Malik, 2001). Segmentation is known to be a very subjective problem, making the evaluation of automated algorithms challenging. To provide a means of evaluating segmentation systems in an objective, quantitative manner, the Berkeley database was constructed. This database consists of approximately 1,000 unique images that have been segmented by hand by 30 individuals. Benchmark results for Berkeley's database are also provided online.

Clearly large, standard databases with well-defined evaluation methodologies and strategies for comparing against other systems in the field have had significant impact on other research areas. We firmly believe that the introduction of such an evaluation paradigm to the field of visual tracking would have equally as large of an effect. A researcher or group of researchers could literally make a career of constructing and maintaining a dataset, defining performance metrics, and objectively ranking results that are provided by other researchers in the field. In fact, some standard evaluation systems require authors to submit their algorithms to a third party who will execute and evaluate the algorithms on their behalf. Such a system would most likely be the ideal end goal to be attained for visual tracking. Realistically, the field will not arrive at a complete evaluation system immediately. However, the creation of a large, standard database with well-defined evaluation metrics, and that systematically presents the various subproblems of visual tracking would be an excellent first step. Subsequently, the database could be updated to ensure system's are not being fine-tuned. Finally, in the long term, it would be excellent if on-line system comparisons and algorithm evaluation submissions were available.

8.2 Richer and Adapting Feature Sets

The particular features used by a visual tracking system is a topic that has been mentioned numerous times throughout this review. The features used have a huge impact on the tracking accuracy that can be obtained by a system. One can think of the "garbage in, garbage out" philosophy. In other words, if poor features are selected for tracking that do not sufficiently distinguish the target, it does not matter what tracking machinery is used. In particular, with a poor choice of features, all tracking machinery will perform roughly equally poorly. On the other hand, an extremely descriptive feature set would go a long way toward solving the visual tracking subproblems listed in 7.1. Imagine if the "perfect" feature set had been found for the problem of visual tracking that could distinguish a specified target from any other object in a scene under any lighting condition. If such a feature set existed, at a minimum, the problems of occlusion, illumination changes, clutter, and multi-target tracking would be solved.

Great strides have recently been made in terms of developing more rich feature sets (Cannons & Wildes, 2007). Oriented energy features certainly have provided a mechanism for capturing both spatial and dynamic properties of a target, but their wide-spread applicability to trackers outside of the mean shift paradigm has yet to be demonstrated. Furthermore, although oriented energies provide a very rich target description, it is questionable as to whether they are the "optimal" features. Further research may uncover a new type of feature that is even richer and more descriptive. Researchers are encouraged to consider this problem.

On the other hand, perhaps there is no single "optimal" feature. Maybe no single feature exists that will allow computer vision tracking systems to operate effectively across all situations. For instance, despite the extreme richness of spatiotemporal oriented energies, there still remain instances when color features will provide superior performance. Considering this point of view, perhaps a second problem that should be considered in tandem is: "how should multiple feature modalities be combined in a principled fashion?" As mentioned previously, the initial exploratory work on this topic has begun (Spengler & Schiele, 2003; Collins & Liu, 2003; Leung & Gong, 2006; Avidan, 2007; Yin, Porikli & Collins, 2008). Alternative methods of combining color and other types of features will undoubtedly be an extremely active area within the field of visual tracking in coming years.

8.3 Template Updates

The issue of updating the target template is another problem that has received only limited attention within the field of visual tracking. The problems of online feature selection and template updating are very closely related, but for the purpose of this review, we are differentiating between the two problems. Specifically, online feature selection deals with the task of determining what target features should be used when comparing the current image data to the template. Template updating, on the other hand, considers the problem of keeping the target template up-to-date with the current target appearance, regardless of what feature representation is used.

One of the factors that makes this problem challenging is the seemingly conflicting requirements. If the template is adjusted quickly so that it can keep up with fast-changing targets, there is also a high likelihood that it will be distracted by clutter or occlusion events. On the other hand, templates that are updated too slowly are not well-adept for tracking targets that experience rapid changes. Initial research by Jepson et al. has been conducted on this issue (Jepson, Fleet & El-Maraghi, 2003). However, a problem as fundamental and important as template updates surely requires additional research. Perhaps the reason that template updates have not been the focus of greater research is that most video sequences considered in the literature are quite short (less than 100 frames). Accordingly, the targets of interest will often not experience significant changes during the duration of the sequence, making robust template updating mechanisms unnecessary. However, for visual trackers to become truly useful for automated systems in commercial environments, superior methods of updating templates will certainly be required.

Not unlike the identification of an extremely rich feature set, solving the template update problem would yield significant contributions toward solving the various sub-problems listed in Table 7.1. In particular, non-rigid deformations and changes in illumination would no longer pose a significant problem, as whenever these changes occurred, the template would be correctly updated to capture the new appearance of the target. It should be highlighted that the problems of occlusion, multi-target tracking, and clutter might still exist. More specifically, if the feature set used is not sufficiently descriptive to discriminate between the target and other scene regions (foreground and background), even "perfect" template updating would not be able to provide robust results for these subproblems.

8.4 Combined Tracking Methods

As evidenced by the trackers described in Chapter 6, the idea of combining multiple trackers from different paradigms is of great interest to researchers and appears to be a promising approach. In a similar manner to the features used, it is unlikely that any particular tracking mechanism can perform well across all tracking situations. Acknowledging this assumption, the problem then becomes that of optimally combining information from multiple sources in a principled and rigorous framework. Initial studies in this area have considered the sequential refinement of tracking results (e.g., (Chung, MacLean & Dickinson, 2006; Shao, Porikli & Chellappa, 2007) and the probabilistic combination of tracking information (e.g., (Rasmussen & Hager, 2001; Leichter, Lindenbaum & Rivlin, 2004; Han, Joo & Davis, 2007); however, further research is required to determine the benefits and shortcomings of these disparate approaches. In addition to studying the method of combining the information provided by multiple tracking paradigms, more research is required in understanding the paradigms themselves. Specifically, a deeper understanding of the complimentary nature of various approaches to visual tracking is required. With this understanding, researchers can determine what paradigms should be incorporated into their systems in order to design trackers that are robust to the foreseeable challenges that will be encountered.

8.5 Final Remarks

Visual tracking, in the grand scheme of academic research, is an extremely new research area. However, in the past 20-30 years since visual tracking has become a significant focus of attention, it is undeniable that great strides have been made. These advances have been observed in every category of tracker — from discrete feature trackers, to contour trackers, and region trackers. Given how fundamental visual tracking is to numerous other computer vision applications, the amount of attention and advancement visual tracking has seen in the past few decades is not surprising.

Many researchers have moved on to studying higher level applications (e.g., security and surveillance) that merely make use of visual tracking as a tool. To these researchers, current visual trackers presumably provide satisfactory results for, at least, a limited range of applications and scenarios that are being studied. Despite the advancements in the field and the deployment in higher level applications, we truly believe there is significant work that must be completed before visual tracking is completely "solved".

In our opinion, the field of visual tracking is currently somewhat stagnant. By stagnant, we do not mean that no new ideas are being explored and introduced, because this is certainly not true. Rather, the field is stagnant in that many new systems and ideas are being introduced, but there is no rigorous means of evaluating them. In other words, tracking systems are being produced but there is no way of definitively knowing which tracking system is "best". To advance the field, a thorough database with standard testing procedures is desperately needed.

If we had to select the most significant works in visual tracking that had the largest impact on the field, we would choose: (Kalman, 1960), (Sethi & Jain, 1987), (Lowe, 1992), (Kass, Witkin & Terzopoulos, 1988), (Caselles, Kimmel & Sapiro, 1995), (Isard & Blake, 1996), (Wren, Azarbayejani, Darrell & Pentland, 1997), (Lucas & Kanade, 1981), (Bergen, Anandan, Hanna & Hingorani, 1992), and (Comaniciu, Ramesh & Meer, 2000). The abovementioned seminal works were highlighted for the following reasons. The research in (Kalman, 1960) and (Isard & Blake, 1996) provided the predictive machinery that is used almost exclusively within the field of visual tracking. Although it is certainly true that the particle filter was seen in other fields prior to the work in (Isard & Blake, 1996), this research introduced, and clearly illustrated, the benefits of particle filtering for visual tracking. With respect to discrete feature trackers, in (Sethi & Jain, 1987), deterministic point tracking that used more than just nearest neighbor matching methods was introduced. For many years, the subsequent deterministic point trackers were primarily refinements on this original approach, rather than radically new techniques. The 3D model-based tracker in (Lowe, 1992) was arguably the first to apply 3D model detection techniques to tracking, develop algorithms for both the feature correspondence and alignment subproblems, and clearly demonstrate the power of model-based tracking techniques. In terms of contour-based trackers, the work of (Kass, Witkin & Terzopoulos, 1988) was essentially the seminal work in the entire field and (Caselles, Kimmel & Sapiro, 1995) introduced level sets to the contour evolution paradigm — a representation that was arguably needed to move the field forward. Finally, in the realm of region-trackers, (Wren, Azarbayejani, Darrell & Pentland, 1997) demonstrated how simple color-based blob tracking approaches can work surprisingly well in limited situations with stationary cameras. Furthermore, (Lucas & Kanade, 1981) provided the seminal work in terms of translational image alignment while (Bergen, Anandan, Hanna & Hingorani, 1992) revitalized the area of image warping by introducing a unifying framework and allowing for a wider range of parametric motion models to be employed. Warping-based trackers published today often use a framework that has only been slightly modified from that proposed in (Bergen, Anandan, Hanna & Hingorani, 1992). Finally, the work in (Comaniciu, Ramesh & Meer, 2000) appeared to renew interest in the field of visual tracking and created significant interest in color histogram target representations, which have been used extensively in the last eight years.

It feels like the next significant technological advancement in the field will probably be centered around the features used during tracking, because of the pervasive impact advances here can have, as well as recent interest in improved features, discussed above. However, it is certainly possible that we are completely incorrect and that the next major advancement might arise from the development of new tracking machinery or something else entirely. That is the exciting thing about research — one never knows for certain what and when the next major advancement will be.

Bibliography

Adam, A., Rivlin, E. & Shimshoni, I. (2006). Robust fragments-based tracking using the integral histogram. In *CVPR* (pp. 798–805).

Adelson, E. & Bergen, J. (1985). Spatiotemporal energy models for the perception of motion. *JOSA A*, *2(2)*, 284–299.

Aggarwal, J. K. & Cai, Q. (1999). Human motion analysis: a review. *CVIU*, *73(3)*, 428–440.

Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *IJCV*, *2(3)*, 283–310.

Anderson, C. H., Burt, P. J. & van der Wal, G. S. (1985). Change detection and tracking using pyramid transform techniques. *SPIE Intelligent Robots and Computer Vision*, *579*, 72–78.

Avidan, S. (2001). Support vector tracking. In *CVPR* (pp. 184–191).

Avidan, S. (2007). Ensemble tracking. *PAMI*, *29(2)*, 261–271.

Ayer, S. & Sawhney, H. S. (1995). Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *ICCV* (pp. 777–784).

Babu, R. V., Perez, P. & Bouthemy, P. (2007). Robust tracking with motion estimation and local kernel-based color modeling. *IVC*, *25*, 1205–1216.

Badrinarayanan, V., Perez, P., Cler, F. & Oisel, L. (2007). Probabilistic color and adaptive multi-feature tracking with dynamically switched priority between cues. In *ICCV* (pp. 1–8).

Baker, S., Gross, R. & Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework: Part 4. Technical Report CMU-RI-TR-04-14, Robotics Institute, Carnegie Mellon University.

Bar-Shalom, Y. & Fortmann, T. E. (1988). *Tracking and data association.* San Diego, California: Academic Press, Inc.

Beaulieu, J. M. & Goldberg, M. (1989). Hierarchy in picture segmentation: a stepwise optimization approach. *PAMI*, *11(2)*, 150–163.

Bergen, J. R., Anandan, P., Hanna, K. J. & Hingorani, R. (1992). Hierarchical model-based motion estimation. In *ECCV* (pp. 237–252).

Bergen, J. R., Burt, P. J., Hingorani, R. & Peleg, S. (1990). Transparent-motion analysis. In *ECCV* (pp. 566–569).

Bergholm, F. (1987). Edge focusing. *PAMI*, *9(6)*, 726–741.

Bertalmio, M., Sapiro, G. & Randall, G. (2000). Morphing active contours. *PAMI*, *22(7)*, 733–737.

Besl, P. J. & McKay, N. D. (1992). A method for registration of $3-d$ shapes. *PAMI*, *14(2)*, 239–256.

Beymer, D. & Konolige, K. (1999). Real-time tracking of multiple people using continuous detection. In *ICCV*.

Bigün, J., Granlund, G. H. & Wiklund, J. (1991). Multidimensional orientation estimation with applications to texture analysis and optical flow. *PAMI*, *13(8)*, 775–790.

Birchfield, S. & Rangarajan, S. (2005). Spatiograms versus histograms for region-based tracking. *CVPR*, *2*, 1158–1163.

Birchfield, S. T. & Pundlik, S. J. (2008). Joint tracking of features and edges. In *CVPR*.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford, New York: Oxford University Press.

Black, M. J. & Anandan, P. (1996). The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *CVIU*, *63(1)*, 75–104.

Black, M. J. & Jepson, A. D. (1998). EigenTracking: robust matching and tracking of articulated objects using a view-based representation. *IJCV*, *26(1)*, 63–84.

Blake, A., Isard, M. & Reynard, D. (1994). Learning to track curves in motion. In *Proceedings of the Conference on Decision and Control* (pp. 3788–3793).

Brand, M. & Bhotika, R. (2001). Flexible flow for 3D nonrigid tracking and shape recovery. In *CVPR*.

Brand, M. & Kettnaker, V. (2000). Discovery and segmentation of activities in video. *IEEE PAMI*, *22(8)*, 844–851.

Brown, M. Z., Burschka, D. & Hager, G. D. (2003). Advances in computational stereo. *PAMI*, *25(8)*, 993–1008.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167.

Burt, P. (1981). Fast filter transforms for image processing. *Computer Graphics and Image Processing*, *16(1)*, 20–51.

Burt, P. (1991). Image motion analysis made simple and fast, one component at a time. In *British Machine Vision Conference* (pp. 1–8).

Burt, P. & Adelson, E. (1983). The Laplacian pyramid as a compact image code. *IEEE TC*, *31(4)*, 532–540.

Burt, P., Yen, C. & Xu, X. (1982). Local correlation measures for motion analysis: a comparative analysis. In *PRIP* (pp. 269–274).

Burt, P. J., Bergen, J. R., Hingorani, R., Lee, W. A. & Leung, A. (1989). Object tracking with a moving camera. In *Workshop on Visual Motion* (pp. 2–12).

Burt, P. J., Yen, C. & Xy, X. (1983). Multi-resolution flow — through motion analysis. In *CVPR* (pp. 246–252).

Cannons, K. & Wildes, R. (2007). Spatiotemporal oriented energy features for visual tracking. In *ACCV* (pp. 532–543).

Canny, J. (1986). A computational approach to edge detection. *PAMI*, *8(6)*, 679–698.

Caselles, V., Kimmel, R. & Sapiro, G. (1995). Geodesic active contours. In *Fifth International Conference on Digital Object Identifier* (pp. 694–699).

Caselles, V., Kimmel, R. & Sapiro, G. (1997). Geodesic active contour. *IJCV*, *22(1)*, 61–79.

Cham, T. J. & Rehg, J. M. (1999). A multiple hypothesis approach to figure tracking. In *CVPR* (pp. 239–245).

Chang, C., Chia, T. & Yang, C. (2005). Modified temporal difference method for change detection. *Optical Engineering*, *44(2)*.

Chen, D. & Yang, J. (2007). Robust online tracking via online dynamic spatial bias appearance models. *PAMI*, *29(12)*, 2157–2169.

Chen, Y., Rui, Y. & Huang, T. S. (2001). JPDAF based HMM for real-time contour tracking. In *ICCV* (pp. 543–550).

Chung, D., MacLean, W. J. & Dickinson, S. (2006). Integrating region and boundary information for spatially coherent object tracking. *Image and Vision Computing*, *24*, 680–692.

Collins, R. (2003). Mean-shift blob tracking through scale space. *CVPR*, *2*, 234–240.

Collins, R., Zhou, X. & Teh, S. K. (2005). An open source tracking testbed and evaluation web site. In *PETS 2005* (pp. 361–366).

Collins, R. T. & Liu, Y. (2003). On-line selection of discriminative tracking features. In *ICCV* (pp. 346–352).

Comaniciu, D., Ramesh, V. & Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. *CVPR*, *2*, 142–149.

Comaniciu, D., Ramesh, V. & Meer, P. (2003). Kernel-based object tracking. *IEEE PAMI*, *25(5)*, 564–575.

Coombs, D. & Brown, C. M. (1993). Real-time binocular smooth-pursuit. *IJCV*, *11(2)*, 147–165.

Cox, I. J. (1993). A review of statistical data association techniques for motion correspondence. *IJCV*, *10(1)*, 53–66.

Crowley, J. L., Stelmaszyk, P. & Discours, C. (1988). Measuring image flow by tracking edge-lines. In *ICCV* (pp. 658–664).

Dailey, D. J. & Li, L. (1999). An algorithm to estimate vehicle speed using uncalibrated cameras. In *International Conference on Intelligent Transportation Systems* (pp. 441–446).

Daniilidis, K., Krauss, C., Hansen, M. & Sommer, G. (1998). Real time tracking of moving objects with an active camera. *Real-Time Imaging*, *4(1)*, 3–20.

Delamarre, Q. & Faugeras, O. (1999). 3D articulated models and multi-view tracking with silhouettes. In *International Conference on Computer Vision* (pp. 716–721).

Dempster, A. P., Laird, N. M. & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, *39(1)*, 1–38.

Deriche, R. & Faugeras, O. (1991). Tracking line segments. *IVC*, *8(4)*, 261–270.

Derpanis, K. (2003). Vision based gesture recognition within a linguistics framework. Technical Report CS-2004-02, York University, Department of Computer Science.

Derpanis, K. (2006). Characterizing image motion. Technical Report CS-2006-06, York University, Department of Computer Science.

Derpanis, K. & Gryn, J. (2005). Three-dimensional nth derivative of Gaussian separable steerable filters. *ICIP*, *3*, 553–556.

Drummond, T. W. & Cipolla, R. (2002). Real-time tracking of complex structures with on-line camera calibration. *IVC*, *20*, 427–433.

Duda, R. O. & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, *15(1)*, 11–15.

Enzweiler, M., Wildes, R. & Herpers, R. (2005). Unified target detection and tracking using motion coherence. *Wrkshp. Motion & Video Comp.*, *2*, 66–71.

Fan, Z., Yang, M. & Wu, Y. (2007). Multiple collaborative kernel tracking. *PAMI*, *29(7)*, 1268–1273.

Fieguth, P. & Terzopoulos, D. (1997). Color-based tracking of heads and other mobile objects at video frame rates. In *CVPR* (pp. 21–27).

Fischler, M. A. & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24(6)*, 381–395.

Fisher, R. (2002). http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

Fleck, M., Forsyth, D. & Bregler, C. (1996). Finding naked people. In *ECCV* (pp. 592–602).

Forney, G. D. (1973). The Viterbi algorithm. *IEEE Proceedings*, *61(3)*, 268–278.

Fortmann, T. E., Bar-Shalom, Y. & Scheffe, M. (1983). Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, *8(3)*, 173–184.

Fua, P. & Lepetit, V. (2005). Vision based 3D tracking and pose estimation for mixed reality. In M. Haller, M. Billinghurst & B. H. Thomas (Eds.), *Emerging Technologies of Augmented Reality: Interfaces and Design*. IGI Global.

Fukunaga, K. & Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE IT*, *21(1)*, 32–40.

Gavrila, D. M. (1999). The visual analysis of human movement: a survey. *CVIU*, *73(1)*, 82–98.

Geman, D. & Jedynak, B. (1996). An active testing model for tracking roads in satellite images. *PAMI*, *18(1)*, 1–14.

Gennery, D. B. (1992). Visual tracking of known three-dimensional objects. *IJCV*, *7(3)*, 243–270.

Goldenshluger, A. & Zeevi, A. (2004). The Hough transform estimator. *The Annals of Statistics*, *32(5)*, 1908–1932.

Hager, G., Dewan, M. & Stewart, C. (2004). Multiple kernel tracking with SSD. *CVPR*, *1*, 790–797.

Hager, G. D. & Belhumeur, P. N. (1998). Efficient region tracking with parametric models of geometry and illumination. *PAMI*, *20(10)*, 1025–1039.

Han, B., Comaniciu, D., Zhu, Y. & Davis, L. S. (2008). Sequential kernel density approximation and its application to real-time visual tracking. *PAMI*, *30(7)*, 1186–1197.

Han, B., Joo, S. W. & Davis, L. S. (2007). Probabilistic fusion tracking using mixture kernel-based Bayesian filtering. In *ICCV* (pp. 1–8).

Haritaoglu, I., Harwood, D. & Davis, L. S. (1999). Hydra: multiple people detection and tracking using silhouettes. In *International Conference on Digital Object Identifier* (pp. 280–285).

Haritaoglu, L., Harwood, D. & Davis, L. (2000). W4: real-time surveillance of people and their activities. *PAMI*, *22(8)*, 809–830.

Harris, C. & Stephens, M. (1988). Combined corner and edge detector. In *The Fourth Alvey Vision Conference* (pp. 147–151).

Heath, M. D., Sarkar, S., Sanocki, T. & Bowyer, K. W. (1997). A robust visual method for assessing the relative performance of edge-detection algorithms. *PAMI*, *19(12)*, 1338–1359.

Heckenberg, D. (2006). Performance evaluation of vision-based high DOF human movement tracking: a survey and human computer interaction perspective. In *Workshop on Vision for Human Computer Interaction.*

Heijmans, H. J. A. M. (1994). *Morphological Image Operators.* Boston: Academic Press.

Herpers, R., Derpanis, K., MacLean, W. J., Verghese, G., Jenkin, M., Milios, E., Jepson, A. & Tsotsos, J. K. (2001). SAVI: an actively controlled teleconferencing system. *Image and Vision Computing*, *19*, 793–804.

Hogg, D. (1983). Model-based vision: A program to see a walking person. *Image and Vision Computing*, *1(1)*, 5–20.

Hough, P. (1962). Method and means for recognizing complex patterns. U.S. Patent 3 069 654.

Huttenlocher, D., Noh, J. & Rucklidge, W. (1993). Tracking non-rigid objects in complex scenes. In *ICCV* (pp. 93–101).

Intille, S. S., Davis, J. W. & Bobick, A. F. (1997). Real-time closed-world tracking. In *CVPR* (pp. 697–703).

Irani, M., Rousso, B. & Peleg, S. (1994). Computing occluding and transparent motions. *IJCV*, *12(1)*, 5–16.

Isard, M. & Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. *ECCV*, *1*, 343–354.

Isard, M. & MacCormick, J. (2001). BraMBLe: a Bayesian multiple-blob tracker. In *ICCV* (pp. 34–41).

Iverson, L. A. & Zucker, S. W. (1995). Logical/Linear operators for image curves. *PAMI*, *17(10)*, 982–996.

Jähne, B. (1990). Motion determination in space-time images. In *ECCV* (pp. 161–173).

Jain, R. (1984). Difference and accumulative difference pictures in dynamic scene analysis. *IVC*, *2(2)*, 99–108.

Jehan-Besson, S., Barlaud, M. & Aubert, G. (2001). Video object segmentation using Eulerian region-based active contours. In *International Conference on Digital Object Identifier* (pp. 353–360).

Jepson, A., Fleet, D. & El-Maraghi, T. (2003). Robust online appearance models for visual tracking. *IEEE PAMI*, *25(10)*, 1296–1311.

Johansson, G. (1975). Visual motion perception. *Scientific American*, *232*, 76–88.

Julier, S. J., Uhlmann, J. K. & Durrant-Whyte, H. F. (1995). A new approach for filtering nonlinear systems. In *American Control Conference* (pp. 1628–1632).

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, *82*, 32–45.

Kang, J., Cohen, I. & Medioni, G. (2003). Continuous tracking within and across camera streams. In *CVPR* (pp. 267–272).

Kass, M., Witkin, A. & Terzopoulos, D. (1988). Snakes: active contour models. *IJCV* (pp. 321–331).

Khan, Z., Balch, T. & Dellaert, F. (2006). MCMC data association and sparse factorization updating for real time multitarget tracking with merged and multiple measurements. *PAMI*, *28(12)*, 1960–1972.

Kim, Z. W. (2008). Real time object tracking based on dynamic feature grouping with background subtraction. In *CVPR*.

Kjeldsen, R. & Kender, J. (1996). Finding skin in color images. In *Automatic Face and Gesture Recognition* (pp. 312–317).

Koller, D., Daniilidis, K. & Nagel, H. H. (1993). Model-based object tracking in monocular image sequences of road traffic scenes. *IJCV*, *10(3)*, 257–281.

Koller, T., Gerig, G., Szekely, G. & Dettwiler, D. (1995). Multiscale detection of curvilinear structures in 2D and 3D image data. In *ICCV* (pp. 864–869).

Lee, L., Romano, R. & Stein, G. (2000). Monitoring activities from multiple video streams: establishing a common coordinate frame. *PAMI*, *22(8)*, 758–767.

Leichter, I., Lindenbaum, M. & Rivlin, E. (2004). A probabilistic framework for combining tracking algorithms. In *CVPR* (pp. 445–451).

Leichter, I., Lindenbaum, M. & Rivlin, E. (2007). Visual tracking by affine kernel fitting using color and objects boundary. In *ICCV* (pp. 1–6).

Leng, J. & Wang, H. (2004). Tracking as recognition: A stable 3D tracking framework. In *Conference on Control, Automation, Robotics and Vision* (pp. 2303–2307).

Lepetit, V., Lagger, P. & Fua, P. (2005). Randomized trees for real-time keypoint recognition. In *CVPR* (pp. 775–781).

Leung, A. & Gong, S. (2006). Mean-shift tracking with random sampling. *BMVC*, *2*, 729–738.

Leymarie, F. & Levine, M. D. (1993). Tracking deformable objects in the plane using an active contour model. *PAMI*, *15(6)*, 617–634.

Li, K., Miller, E. D., Weiss, L. E., Campbell, P. G. & Kanade, T. (2006). Online tracking of migrating and proliferating cells imaged with phase-contrast microscopy. In *Workshop on Mathematical Methods in Biomedical Image Analysis* (pp. 65–72).

Lin, K. C. & Tsai, M. C. (2000). Image feedback path tracking control using an uncalibrated CCD camera. *MVA*, *12(2)*, 53–58.

Lindeberg, T. (1998). Feature detection with automatic scale selection. *IJCV*, *30(2)*, 79–116.

Liou, S. P. & Jain, R. C. (1989). Motion detection in spatio-temporal space. *Computer Vision, Graphics, and Image Processing*, *45(2)*, 227–250.

Lipton, A., Fujiyoshi, H. & Patil, R. (1998). Moving target classification and tracking from real-time video. In *DARPA Image Understanding Workshop* (pp. 8–14).

Lowe, D. G. (1991). Fitting parameterized three-dimensional models to images. *PAMI*, *13(5)*, 441–450.

Lowe, D. G. (1992). Robust model-based motion tracking through the integration of search and estimation. *IJCV*, *8(2)*, 113–122.

Lowe, D. G. (1999). Recognition from local scale-invariant features. In *ICCV* (pp. 1150–1157).

Lucas, B. & Kanade, T. (1981). An iterative image registration technique with application to stereo vision. In *DARPA IUW* (pp. 121–130).

Maccormick, J. & Blake, A. (2000). A probabilistic exclusion principle for tracking multiple objects. *IJCV*, *39(1)*, 57–71.

Maintz, J. B., van den Elsen, P. & Viergever, M. A. (1996). Evaluation of ridge seeking operators for multimodality medical image matching. *PAMI*, *18(4)*, 353–365.

Mansouri, A. R. (2002). Region tracking via level set PDEs without motion correspondence. *PAMI*, *24(7)*, 947–961.

Marchand, E., Bouthemy, P. & Chaumette, F. (2001). A 2d-3d model-based approach to real-time visual tracking. *IVC*, *19(13)*, 941–955.

Martin, D., Fowlkes, C., Tal, D. & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV* (pp. 416–423).

Matthews, I., Ishikawa, T. & Baker, S. (2004). The template update problem. *PAMI*, *28(6)*, 810–815.

Matusita, K. (1955). Decision rules, based on the distance, for problems of fit, two samples, and estimation. *The Annals of Mathematical Statistics*, *26(4)*, 631–640.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N. & Teller, A. H. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, *21(6)*, 1087–1092.

Meyer, F. G. & Bouthemy, P. (1994). Region-based tracking using affine motion models in long image sequences. *CVGIP: Image Understanding*, *60(2)*, 119–140.

Mittal, A. & Davis, L. S. (2003). M_2 tracker: a multi-view approach to segmenting and tracking people in a cluttered scene. *IJCV*, *51(3)*, 189–203.

Moeslund, T. B. & Granum, E. (2000). Multiple cues used in model-based human motion capture. In *Conference on Automatic Face and Gesture Recognition.*

Moravec, H. (1980). Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical Report CMU-RI-TR-3, Carnegie-Mellon University, Robotics Institute.

Morefield, C. L. (1977). Application of 0-1 integer programming to multitarget tracking problems. *IEEE Transactions on Automatic Control*, *22(6)*, 302–312.

Moselund, T. B. & Granum, E. (2001). A survey of computer vision-based human motion capture. *CVIU*, *81*, 231–268.

Munkelt, O., Ridder, C., Hansel, D. & Hafner, W. (1998). A model driven 3D image interpretation system applied to person detection in video images. In *International Conference on Pattern Recognition* (pp. 70–73).

Murray, D., Bradshaw, K., McLauchlan, P., Reid, I. & Sharkey, P. (1995). Driving saccade to pursuit using image motion. *IJCV*, *16(3)*, 205–228.

Nalwa, V. S. & Binford, T. O. (1986). On detecting edges. *PAMI*, *8(6)*, 699–714.

Nejhum, S. M. S., Ho, J. & Yang, M. H. (2008). Visual tracking with histograms and articulating blocks. In *CVPR*.

Nguyen, H. T. (2006). Robust tracking using foreground-background texture discrimination. *IJCV*, *69(3)*, 277–293.

Nguyen, H. T., Worring, M. & Boomgaard, R. (2001). Occlusion robust adaptive template tracking. In *ICCV* (pp. 678–683).

Nummiaro, K., Koller-Meier, E. & Gool, L. V. (2002). An adaptive color-based particle filter. *Image and Vision Computing*, *21*, 99–110.

Ogata, K. (2002). *Modern Control Engineering*. Upper Saddle River, New Jersey: Prentice Hall, Inc.

Okuma, K., Taleghani, A., Freitas, N. D., Little, J. & Lowe, D. (2004). A boosted particle filter: multitarget detection and tracking. *ECCV*, *1*, 28–39.

Olson, C. F. (2000). Maximum-likelihood template matching. In *CVPR* (pp. 52–57).

Osher, S. & Fedkiw, R. (2003a). *Level Set Methods and Dynamic Implicit Surfaces*. New York, New York: Springer.

Osher, S. & Fedkiw, R. (2003b). *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. New York, New York: Springer.

Papanikolopoulos, N., Khosla, P. & Kanade, T. (1993). Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *RA*, *9*, 14–35.

Parag, T., Porikli, F. & Elgammal, A. (2008). Boosting adaptive linear weak classifiers for online learning and tracking. In *CVPR*.

Paragios, N. & Deriche, R. (2000). Geodesic active contours and level sets for the detection and tracking of moving objects. *PAMI*, *22(3)*, 265–280.

Parameswaran, V., Ramesh, V. & Zoghlami, I. (2006). Tunable kernels for tracking. In *CVPR* (pp. 2179–2186).

Pentland, A., Moghaddam, B. & Starner, T. (1994). View-based and modular eigenspaces for face recognition. In *CVPR* (pp. 84–91).

Peterfreund, N. (1997). The velocity snake. In *CVPR* (pp. 1–8).

Peterfreund, N. (1999). Robust tracking of position and velocity with kalman snakes. *PAMI*, *21(6)*, 564–569.

PETS (2000). http://www.cvg.rdg.ac.uk/PETS2000/.

PETS (2001). http://www.cvg.rdg.ac.uk/PETS2001/.

PETS (2002). http://www.cvg.rdg.ac.uk/PETS2002/.

PETS (2003). http://www.cvg.rdg.ac.uk/PETS2003/.

PETS (2004). http://www-prima.imag.fr/PETS04/index.html.

PETS (2005). http://www.cvg.rdg.ac.uk/PETS2005/.

PETS (2006). http://peipa.essex.ac.uk/ipa/pix/pets/.

PETS (2007). http://www.pets2007.net/.

Phillips, P. J., Moon, H., Rizvi, S. A. & Rauss, P. J. (2000). The FERET evaluation methodology for face-recognition algorithms. *PAMI*, *22(10)*, 1090–1104.

Porikli, F. (2005). Integral histogram: A fast way to extract histograms in Cartesian spaces. In *CVPR* (pp. 829–836).

Poritz, A. B. (1988). Hidden Markov models: a guided tour. In *International Conference on Acoustics, Speech, and Signal Processing* (pp. 7–13).

Rangarajan, K. & Shah, M. (1991). Establishing motion correspondence. *CVGIP*, *54(1)*, 56–73.

Rasmussen, C. & Hager, G. D. (2001). Probabilistic data association methods for tracking complex visual objects. *PAMI*, *23(6)*, 560–576.

Rathi, Y., Vaswani, N., Tannenbaum, A. & Yezzi, A. (2007). Tracking deforming objects using particle filtering for geometric active contours. *PAMI*, *29(8)*, 1470–1475.

Reid, D. B. (1979). An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, *24(6)*, 843–854.

Rittscher, J., Kato, J., Joga, S. & Blake, A. (2000). A probabilistic background model for tracking. In *ECCV* (pp. 336–350).

Ronfard, R. (1994). Region-based strategies for active contour models. *IJCV*, *13(2)*, 229–251.

Rosales, R. & Sclaroff, S. (1999). 3D trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *CVPR* (pp. 117–123).

Rosenbluth, M. N. & Rosenbluth, A. W. (1955). Monte carlo calculation of the average extension of molecular chains. *The Journal of Chemical Physics*, *23(2)*, 356–359.

Ross, D. A., Lim, J. & Lin, R. S. (2008). Incremental learning for robust visual tracking. *IJCV*, *77*, 125–141.

Rubner, Y., Tomasi, C. & Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *IJCV*, *40(2)*, 99–121.

Salari, V. & Sethi, I. K. (1990). Feature point correspondence in the presence of occlusion. *PAMI*, *12(1)*, 87–91.

Sarkar, S., Phillips, P. J., Liu, Z., Vega, I. R., Grother, P. & Bowyer, K. W. (2005). The HumanID gait challenge problem: data sets, performance, and analysis. *PAMI*, *27(2)*, 162–177.

Scharstein, D. & Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *ICCV* (pp. 7–42).

Schmid, C., Mohr, R. & Bauckhage, C. (2000). Evaluation of interest point detectors. *IJCV*, *37(2)*, 151–172.

Serby, D., Meier, E. K. & Gool, L. V. (2000). Probabilistic object tracking using multiple features. In *ICPR* (pp. 184–187).

Sethi, I. & Jain, R. (1987). Finding trajectories of feature points in monocular images. *PAMI*, *9(1)*, 56–73.

Shafique, K. & Shah, M. (2003). A non-iterative greedy algorithm for multi-frame point correspondence. In *ICCV* (pp. 110–115).

Shang, L., Jasiobedzki, P. & Greenspan, M. (2007). Model-based tracking by classification in a tiny discrete pose space. *PAMI*, *29(6)*, 976–989.

Shao, J., Porikli, F. & Chellappa, R. (2007). Estimation of contour motion and deformation for nonrigid object tracking. *Journal of Optical Society of America*, *24(8)*, 2109–2121.

Shi, J. & Malik, J. (2000). Normalized cuts and image segmentation. *PAMI*, *22(8)*, 888–905.

Shi, J. & Tomasi, C. (1994). Good features to track. *CVPR*, *1*, 593–600.

Simon, D. A., Hebert, M. & Kanade, T. (1994). Real-time 3-D pose estimation using a high-speed range sensor. In *Conference on Robotics and Automation* (pp. 2235–2241).

Simon, G. & Berger, M.-O. (1998). A two-stage robust statistical method for temporal registration from features of various type. In *ICCV* (pp. 261–266).

Singer, R. A. & Stein, J. J. (1971). An optimal tracking filter for processing sensor data of imprecisely determined origin in surveillance systems. In *Conference on Decision and Control* (pp. 171–175).

Sittler, R. W. (1964). An optimal data association problem in surveillance theory. *IEEE Transactions on Military Electronics*, *8*, 125–139.

Sizintsev, M., Derpanis, K. G. & Hogue, A. (2008). Histogram-based search: A comparative study. In *CVPR*.

Smith, P. & Buechler, G. (1975). A branching algorithm for discriminating and tracking multiple objects. *IEEE Transactions on Automatic Control*, *20*, 101–104.

Smith, S. M. & Brady, J. M. (1995). ASSET-2: real-time motion segmentation and shape tracking. *PAMI*, *17(8)*, 814–820.

Spengler, M. & Schiele, B. (2003). Towards robust multi-cue integration for visual tracking. *MVA*, *14*, 50–58.

Stauffer, C. & Grimson, W. (2000). Learning patterns of activity using real-time tracking. *PAMI*, *22(8)*, 747–757.

Steger, C. (1998). An unbiased detector of curvilinear structures. *PAMI*, *20(2)*, 113–125.

Swain, M. J. & Ballard, D. H. (1991). Color indexing. *IJCV*, *7(1)*, 11–32.

Takala, V. & Pietikainen, M. (2007). Multi-object tracking using color, texture and motion. In *ICCV* (pp. 1–7).

Tao, H., Sawhney, H. S. & Kumar, R. (2002). Object tracking with Bayesian estimation of dynamic layer representations. *PAMI*, *24(1)*, 75–89.

Terzopoulos, D. & Szeliski, R. (1992). Tracking with Kalman snakes. In A. Blake & A. Yuille (Eds.), *Active Vision* (pp. 553–556). MIT Press.

Thompson, D. W. & Mundy, J. L. (1987). Three-dimensional model matching from an unconstrained viewpoint. In *Conference on Robotics and Automation* (pp. 208–220).

Torresani, L., Yang, D. B., Alexander, E. J. & Bregler, C. (2001). Tracking and modeling non-rigid objects with rank constraints. In *CVPR*.

Trucco, E. & Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision*. Upper Saddle River, NJ: Prentice Hall, Inc.

Tsin, Y., Genc, Y., Zhu, Y. & Ramesh, V. (2007). Learn to track edges. In *ICCV* (pp. 1–8).

Turk, M. & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, *3(1)*, 71–86.

Uhlin, T., Nordlund, P., Maki, A. & Eklundh, J. O. (1995). Towards an active visual observer. In *ICCV* (pp. 679–686).

Vacchetti, L., Lepetit, V. & Fua, P. (2004). Combining edge and texture information for real-time accurate 3D camera tracking. In *International Symposium on Mixed and Augmented Reality* (pp. 48–56).

Veenman, C. J., Reinders, M. J. & Backer, E. (2001). Resolving motion correspondence for densely moving points. *PAMI*, *23(1)*, 54–72.

Verghese, G., Gale, K. L. & Dyer, C. R. (1990). Real-time, parallel motion tracking of three dimensional objects from spatiotemporal sequences. *Parallel Algorithms for Machine Intelligence and Vision* (pp. 310–339).

Vermaak, J., Doucet, A. & Perez, P. (2003). Maintaining multi-modality through mixture tracking. In *ICCV* (pp. 1110–1116).

Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *CVPR*, Volume 1 (pp. 511–518).

Wang, J. & Adelson, E. (1993). Layered representation for motion analysis. In *CVPR* (pp. 361–366).

Wang, J., Bebis, G. & Miller, R. (2006). Robust video-based surveillance by integrating target detection with tracking. In *Workshop on Object Tracking and Classification Beyond the Visible Spectrum.*

Wax, N. (1955). Signal-to-noise improvement and the statistics of track populations. *Journal of Applied Physics*, *26(5)*, 586–595.

Wilder, J., Mammone, R. J., Meer, P., Flanagan, A., Xu, K., Tsai, A., Weiner, S. & Zhang, X. Y. (1994). Projection-based face recognition. In *SPIE.*

Wildes, R. & Bergen, J. (2000). Qualitative spatiotemporal analysis using an oriented energy representation. *ECCV*, *2*, 784–796.

Wixson, L., Eledath, J., Hansen, M., Mandelbaum, R. & Mishra, D. (1998). Image alignment for precise camera fixation and aim. In *CVPR* (pp. 594–600).

Wong, K. Y. & Spetsakis, M. E. (2006). Tracking based motion segmentation under relaxed statistical assumptions. *CVIU*, *101*, 45–64.

Wren, C., Azarbayejani, A., Darrell, T. & Pentland, A. (1997). Pfinder: real-time tracking of the human body. *PAMI*, *19(7)*, 780–785.

Wu, H., Sankaranarayanan, A. C. & Chellappa, R. (2007). *In Situ* evaluation of tracking algorithms using time reversed chains. In *CVPR* (pp. 1–8).

Wu, Y. & Huang, T. (2001). A co-inference approach to robust visual tracking. In *ICCV* (pp. 26–33).

Wu, Y., Yu, T. & Hua, G. (2005). A statistical field model for pedestrian detection. In *CVPR*, Volume 1 (pp. 1023–1030).

Xu, N. & Ahuja, N. (2002). Object contour tracking using graph cuts based active contours. In *ICIP* (pp. 277–280).

Yang, H., Pollefeys, M., Welch, G., Frahm, J. M. & Ilie, A. (2007). Differential camera tracking through linearizing the local appearance manifold. In *CVPR* (pp. 1–8).

Yang, M., Wu, Y. & Hua, G. (2008). Context-aware visual tracking (to appear). *PAMI.*

Yang, T., Li, S. Z., Pan, Q. & Li, J. (2005). Real-time multiple objects tracking with occlusion handling in dynamic scenes. In *CVPR* (pp. 970–975).

Ye, Y. M., Tsotsos, J. K., Harley, E. & Bennet, K. (2000). Tracking a person with pre-recorded image database by a pan, tilt, and zoom camera. *MVA*, *12(1)*, 32–43.

Yilmaz, A. (2007). Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection. In *CVPR* (pp. 1–6).

Yilmaz, A., Javed, O. & Shah, M. (2006). Object tracking: a survey. *Comp. Surv.*, *38(4)*, 1–45.

Yilmaz, A., Lin, X. & Shah, M. (2004). Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *PAMI*, *26(11)*, 1531–1536.

Yin, Z. & Collins, R. (2007a). Belief propagation in a 3D spatio-temporal MRF for moving object detection. In *CVPR* (pp. 1–8).

Yin, Z. & Collins, R. (2007b). On-the-fly object modeling while tracking. In *ICCV* (pp. 1–8).

Yin, Z., Porikli, F. & Collins, R. T. (2008). Likelihood map fusion for visual object tracking. In *WACV* (pp. 1–7).

Zhang, Z. & Faugeras, O. D. (1992). Three-dimensional motion computation and object segmentation in a long sequence of stereo frames. *IJCV*, *7(3)*, 211–241.

Zhao, T. & Nevatia, R. (2004). Tracking multiple humans in complex situations. *PAMI*, *26(9)*, 1208–1221.

Zhou, Z., Hu, W., Chen, Y. & Hu, W. (2007). Markov random field modeled level sets method for object tracking with moving cameras. In *ACCV* (pp. 832–842).

Zhu, Z., Zhao, Y. & Lu, H. (2007). Sequential architecture for efficient car detection. In *CVPR* (pp. 1–8).