Object Tracking: A Survey

Alper Yilmaz

Ohio State University

Omar Javed

ObjectVideo, Inc.

and

Mubarak Shah

University of Central Florida

The goal of this article is to review the state-of-the-art tracking methods, classify them into different categories, and identify new trends. Object tracking, in general, is a challenging problem. Difficulties in tracking objects can arise due to abrupt object motion, changing appearance patterns of both the object and the scene, nonrigid object structures, object-to-object and object-to-scene occlusions, and camera motion. Tracking is usually performed in the context of higher-level applications that require the location and/or shape of the object in every frame. Typically, assumptions are made to constrain the tracking problem in the context of a particular application. In this survey, we categorize the tracking methods on the basis of the object and motion representations used, provide detailed descriptions of representative methods in each category, and examine their pros and cons. Moreover, we discuss the important issues related to tracking including the use of appropriate image features, selection of motion models, and detection of objects.

Categories and Subject Descriptors: I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Tracking*

General Terms: Algorithms

Additional Key Words and Phrases: Appearance models, contour evolution, feature selection, object detection, object representation, point tracking, shape tracking

ACM Reference Format:

Yilmaz, A., Javed, O., and Shah, M. 2006. Object tracking: A survey. *ACM Comput. Surv. 38*, 4, Article 13 (Dec. 2006), 45 pages. DOI = 10.1145/1177352.1177355 http://doi.acm.org/10.1145/1177352.1177355

This material is based on work funded in part by the US Government. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US Government.

Author's address: A. Yilmaz, Department of CEEGS, Ohio State University; email: yilmaz.15@osu.edu; O. Javed, ObjectVideo, Inc., Reston, VA 20191; email: ojaved@objectvideo.com; M. Shah, School of EECS, University of Central Florida; email: shah@cs.ucf.edu.

1. INTRODUCTION

Object tracking is an important task within the field of computer vision. The proliferation of high-powered computers, the availability of high quality and inexpensive video cameras, and the increasing need for automated video analysis has generated a great deal of interest in object tracking algorithms. There are three key steps in video analysis: detection of interesting moving objects, tracking of such objects from frame to frame, and analysis of object tracks to recognize their behavior. Therefore, the use of object tracking is pertinent in the tasks of:

—motion-based recognition, that is, human identification based on gait, automatic object detection, etc;
—automated surveillance, that is, monitoring a scene to detect suspicious activities or unlikely events;
—video indexing, that is, automatic annotation and retrieval of the videos in multimedia databases;
—human-computer interaction, that is, gesture recognition, eye gaze tracking for data input to computers, etc.;
—traffic monitoring, that is, real-time gathering of traffic statistics to direct traffic flow.
—vehicle navigation, that is, video-based path planning and obstacle avoidance capabilities.

In its simplest form, tracking can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene. In other words, a tracker assigns consistent labels to the tracked objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide object-centric information, such as orientation, area, or shape of an object. Tracking objects can be complex due to:

—loss of information caused by projection of the 3D world on a 2D image,
—noise in images,
—complex object motion,
—nonrigid or articulated nature of objects,
—partial and full object occlusions,
—complex object shapes,
—scene illumination changes, and
—real-time processing requirements.

One can simplify tracking by imposing constraints on the motion and/or appearance of objects. For example, almost all tracking algorithms assume that the object motion is smooth with no abrupt changes. One can further constrain the object motion to be of constant velocity or constant acceleration based on a priori information. Prior knowledge about the number and the size of objects, or the object appearance and shape, can also be used to simplify the problem.

Numerous approaches for object tracking have been proposed. These primarily differ from each other based on the way they approach the following questions: Which object representation is suitable for tracking? Which image features should be used? How should the motion, appearance, and shape of the object be modeled? The answers to these questions depend on the context/environment in which the tracking is performed and the end use for which the tracking information is being sought. A large number of tracking methods have been proposed which attempt to answer these questions for a variety of scenarios. The goal of this survey is to group tracking methods into broad

categories and provide comprehensive descriptions of representative methods in each category. We aspire to give readers, who require a tracker for a certain application, the ability to select the most suitable tracking algorithm for their particular needs. Moreover, we aim to identify new trends and ideas in the tracking community and hope to provide insight for the development of new tracking methods.

Our survey is focused on methodologies for tracking objects in general and not on trackers tailored for specific objects, for example, person trackers that use human kinematics as the basis of their implementation. There has been substantial work on tracking humans using articulated object models that has been discussed and categorized in the surveys by Aggarwal and Cai [1999], Gavrilla [1999], and Moeslund and Granum [2001]. We will, however, include some works on the articulated object trackers that are also applicable to domains other than articulated objects.

We follow a bottom-up approach in describing the issues that need to be addressed when one sets out to build an object tracker. The first issue is defining a suitable representation of the object. In Section 2, we will describe some common object shape representations, for example, points, primitive geometric shapes and object contours, and appearance representations. The next issue is the selection of image features used as an input for the tracker. In Section 3, we discuss various image features, such as color, motion, edges, etc., which are commonly used in object tracking. Almost all tracking algorithms require detection of the objects either in the first frame or in every frame. Section 4 summarizes the general strategies for detecting the objects in a scene. The suitability of a particular tracking algorithm depends on object appearances, object shapes, number of objects, object and camera motions, and illumination conditions. In Section 5, we categorize and describe the existing tracking methods and explain their strengths and weaknesses in a summary section at the end of each category. In Section 6, important issues relevant to object tracking are discussed. Section 7 presents future directions in tracking research. Finally, concluding remarks are sketched in Section 8.

2. OBJECT REPRESENTATION

In a tracking scenario, an object can be defined as anything that is of interest for further analysis. For instance, boats on the sea, fish inside an aquarium, vehicles on a road, planes in the air, people walking on a road, or bubbles in the water are a set of objects that may be important to track in a specific domain. Objects can be represented by their shapes and appearances. In this section, we will first describe the object shape representations commonly employed for tracking and then address the joint shape and appearance representations.

—*Points*. The object is represented by a point, that is, the centroid (Figure 1(a)) [Veenman et al. 2001] or by a set of points (Figure 1(b)) [Serby et al. 2004]. In general, the point representation is suitable for tracking objects that occupy small regions in an image.(see Section 5.1).

—*Primitive geometric shapes*. Object shape is represented by a rectangle, ellipse (Figure 1(c), (d) [Comaniciu et al. 2003], etc. Object motion for such representations is usually modeled by translation, affine, or projective (homography) transformation (see Section 5.2 for details). Though primitive geometric shapes are more suitable for representing simple rigid objects, they are also used for tracking nonrigid objects.

—*Object silhouette and contour*. Contour representation defines the boundary of an object (Figure 1(g), (h). The region inside the contour is called the silhouette of the object (see Figure 1(i)). Silhouette and contour representations are suitable for tracking complex nonrigid shapes [Yilmaz et al. 2004].

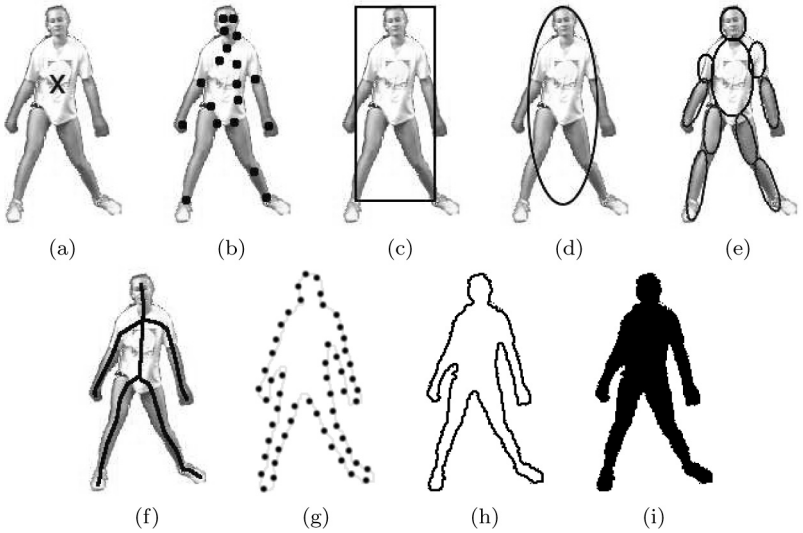


Fig. 1. Object representations. (a) Centroid, (b) multiple points, (c) rectangular patch, (d) elliptical patch, (e) part-based multiple patches, (f) object skeleton, (g) complete object contour, (h) control points on object contour, (i) object silhouette.

—*Articulated shape models.* Articulated objects are composed of body parts that are held together with joints. For example, the human body is an articulated object with torso, legs, hands, head, and feet connected by joints. The relationship between the parts are governed by kinematic motion models, for example, joint angle, etc. In order to represent an articulated object, one can model the constituent parts using cylinders or ellipses as shown in Figure 1(e).

—*Skeletal models.* Object skeleton can be extracted by applying medial axis transform to the object silhouette [Ballard and Brown 1982, Chap. 8]. This model is commonly used as a shape representation for recognizing objects [Ali and Aggarwal 2001]. Skeleton representation can be used to model both articulated and rigid objects (see Figure 1(f).

There are a number of ways to represent the appearance features of objects. Note that shape representations can also be combined with the appearance representations [Cootes et al. 2001] for tracking. Some common appearance representations in the context of object tracking are:

—*Probability densities of object appearance.* The probability density estimates of the object appearance can either be parametric, such as Gaussian [Zhu and Yuille 1996] and a mixture of Gaussians [Paragios and Deriche 2002], or nonparametric, such as Parzen windows [Elgammal et al. 2002] and histograms [Comaniciu et al. 2003]. The probability densities of object appearance features (color, texture) can be computed from the image regions specified by the shape models (interior region of an ellipse or a contour).

—*Templates.* Templates are formed using simple geometric shapes or silhouettes [Fieguth and Terzopoulos 1997]. An advantage of a template is that it carries both spatial and appearance information. Templates, however, only encode the object appearance generated from a single view. Thus, they are only suitable for tracking objects whose poses do not vary considerably during the course of tracking.

—*Active appearance models.* Active appearance models are generated by simultaneously modeling the object shape and appearance [Edwards et al. 1998]. In general, the object shape is defined by a set of landmarks. Similar to the contour-based representation, the landmarks can reside on the object boundary or, alternatively, they can reside inside the object region. For each landmark, an appearance vector is stored which is in the form of color, texture, or gradient magnitude. Active appearance models require a training phase where both the shape and its associated appearance is learned from a set of samples using, for instance, the principal component analysis.

—*Multiview appearance models.* These models encode different views of an object. One approach to represent the different object views is to generate a subspace from the given views. Subspace approaches, for example, Principal Component Analysis (PCA) and Independent Component Analysis (ICA), have been used for both shape and appearance representation [Mughadam and Pentland 1997; Black and Jepson 1998].

Another approach to learn the different views of an object is by training a set of classifiers, for example, the support vector machines [Avidan 2001] or Bayesian networks [Park and Aggarwal 2004]. One limitation of multiview appearance models is that the appearances in all views are required ahead of time.

In general, there is a strong relationship between the object representations and the tracking algorithms. Object representations are usually chosen according to the application domain. For tracking objects, which appear very small in an image, point representation is usually appropriate. For instance, Veenman et al. [2001] use the point representation to track the seeds in a moving dish sequence. Similarly, Shafique and Shah [2003] use the point representation to track distant birds. For the objects whose shapes can be approximated by rectangles or ellipses, primitive geometric shape representations are more appropriate. Comaniciu et al. [2003] use an elliptical shape representation and employ a color histogram computed from the elliptical region for modeling the appearance. In 1998, Black and Jepson used eigenvectors to represent the appearance. The eigenvectors were generated from rectangular object templates. For tracking objects with complex shapes, for example, humans, a contour or a silhouette-based representation is appropriate. Haritaoglu et al. [2000] use silhouettes for object tracking in a surveillance application.

3. FEATURE SELECTION FOR TRACKING

Selecting the right features plays a critical role in tracking. In general, the most desirable property of a visual feature is its uniqueness so that the objects can be easily distinguished in the feature space. Feature selection is closely related to the object representation. For example, color is used as a feature for histogram-based appearance representations, while for contour-based representation, object edges are usually used as features. In general, many tracking algorithms use a combination of these features. The details of common visual features are as follows.

—*Color.* The apparent color of an object is influenced primarily by two physical factors, 1) the spectral power distribution of the illuminant and 2) the surface reflectance properties of the object. In image processing, the RGB (red, green, blue) color space is usually used to represent color. However, the RGB space is not a perceptually uniform color space, that is, the differences between the colors in the RGB space do not correspond to the color differences perceived by humans [Paschos 2001]. Additionally, the RGB dimensions are highly correlated. In contrast, $L^*u^*v^*$ and $L^*a^*b^*$ are perceptually uniform color spaces, while HSV (Hue, Saturation, Value) is an approximately uniform color space. However, these color spaces are sensitive to noise [Song

et al. 1996]. In summary, there is no last word on which color space is more efficient, therefore a variety of color spaces have been used in tracking.

—*Edges.* Object boundaries usually generate strong changes in image intensities. Edge detection is used to identify these changes. An important property of edges is that they are less sensitive to illumination changes compared to color features. Algorithms that track the boundary of the objects usually use edges as the representative feature. Because of its simplicity and accuracy, the most popular edge detection approach is the Canny Edge detector [Canny 1986]. An evaluation of the edge detection algorithms is provided by Bowyer et al. [2001].

—*Optical Flow.* Optical flow is a dense field of displacement vectors which defines the translation of each pixel in a region. It is computed using the brightness constraint, which assumes brightness constancy of corresponding pixels in consecutive frames [Horn and Schunk 1981]. Optical flow is commonly used as a feature in motion-based segmentation and tracking applications. Popular techniques for computing dense optical flow include methods by Horn and Schunck [1981], Lucas and Kanade [1981], Black and Anandan [1996], and Szeliski and Couglan [1997]. For the performance evaluation of the optical flow methods, we refer the interested reader to the survey by Barron et al. [1994].

—*Texture.* Texture is a measure of the intensity variation of a surface which quantifies properties such as smoothness and regularity. Compared to color, texture requires a processing step to generate the descriptors. There are various texture descriptors: Gray-Level Cooccurrence Matrices (GLCM's) [Haralick et al. 1973] (a 2D histogram which shows the cooccurrences of intensities in a specified direction and distance), Law's texture measures [Laws 1980] (twenty-five 2D filters generated from five 1D filters corresponding to level, edge, spot, wave, and ripple), wavelets [Mallat 1989] (orthogonal bank of filters), and steerable pyramids [Greenspan et al. 1994]. Similar to edge features, the texture features are less sensitive to illumination changes compared to color.

Mostly features are chosen manually by the user depending on the application domain. However, the problem of automatic feature selection has received significant attention in the pattern recognition community. Automatic feature selection methods can be divided into *filter* methods and *wrapper* methods [Blum and Langley 1997]. The filter methods try to select the features based on a general criteria, for example, the features should be uncorrelated. The wrapper methods select the features based on the usefulness of the features in a specific problem domain, for example, the classification performance using a subset of features. Principal Component Analysis (PCA) is an example of the filter methods for the feature reduction. PCA involves transformation of a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called the principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. A wrapper method of selecting the discriminatory features for tracking a particular class of objects is the Adaboost [Tieu and Viola 2004] algorithm. Adaboost is a method for finding a strong classifier based on a combination of moderately inaccurate weak classifiers. Given a large set of features, one classifier can be trained for each feature. Adaboost, as discussed in Sections 4.4, will discover a weighted combination of classifiers (representing features) that maximize the classification performance of the algorithm. The higher the weight of the feature, the more discriminatory it is. One can use the first n highest-weighted features for tracking.

Table I. Object Detection Categories

Categories	Representative Work
Point detectors	Moravec's detector [Moravec 1979], Harris detector [Harris and Stephens 1988], Scale Invariant Feature Transform [Lowe 2004]. Affine Invariant Point Detector [Mikolajczyk and Schmid 2002].
Segmentation	Mean-shift [Comaniciu and Meer 1999], Graph-cut [Shi and Malik 2000], Active contours [Caselles et al. 1995].
Background Modeling	Mixture of Gaussians[Stauffer and Grimson 2000], Eigenbackground[Oliver et al. 2000], Wall flower [Toyama et al. 1999], Dynamic texture background [Monnet et al. 2003].
Supervised Classifiers	Support Vector Machines [Papageorgiou et al. 1998], Neural Networks [Rowley et al. 1998], Adaptive Boosting [Viola et al. 2003].

Among all features, color is one of the most widely used feature for tracking. Comaniciu et al. [2003] use a color histogram to represent the object appearance. Despite its popularity, most color bands are sensitive to illumination variation. Hence in scenarios where this effect is inevitable, other features are incorporated to model object appearance. Cremers et al. [2003] use optical flow as a feature for contour tracking. Jepson et al. [2003] use steerable filter responses for tracking. Alternatively, a combination of these features is also utilized to improve the tracking performance.

4. OBJECT DETECTION

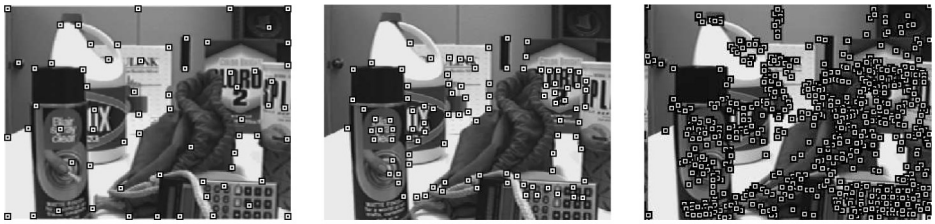
Every tracking method requires an object detection mechanism either in every frame or when the object first appears in the video. A common approach for object detection is to use information in a single frame. However, some object detection methods make use of the temporal information computed from a sequence of frames to reduce the number of false detections. This temporal information is usually in the form of frame differencing, which highlights changing regions in consecutive frames. Given the object regions in the image, it is then the tracker's task to perform object correspondence from one frame to the next to generate the tracks.

We tabulate several common object detection methods in Table I. Although the object detection itself requires a survey of its own, here we outline the popular methods in the context of object tracking for the sake of completeness.

4.1. Point Detectors

Point detectors are used to find interest points in images which have an expressive texture in their respective localities. Interest points have been long used in the context of motion, stereo, and tracking problems. A desirable quality of an interest point is its invariance to changes in illumination and camera viewpoint. In the literature, commonly used interest point detectors include Moravec's interest operator [Moravec 1979], Harris interest point detector [Harris and Stephens 1988], KLT detector [Shi and Tomasi 1994], and SIFT detector [Lowe 2004]. For a comparative evaluation of interest point detectors, we refer the reader to the survey by Mikolajczyk and Schmid [2003].

To find interest points, Moravec's operator computes the variation of the image intensities in a 4×4 patch in the horizontal, vertical, diagonal, and antidiagonal directions and selects the minimum of the four variations as representative values for the window. A point is declared interesting if the intensity variation is a local maximum in a 12×12 patch.



(a) (c) (c)

Fig. 2. Interest points detected by applying (a) the Harris, (b) the KLT, and (c) SIFT operators.

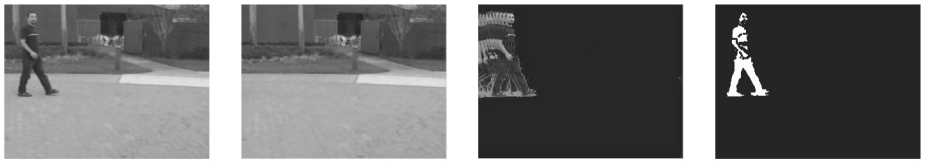
The Harris detector computes the first order image derivatives, (I_x, I_y), in x and y directions to highlight the directional intensity variations, then a second moment matrix, which encodes this variation, is evaluated for each pixel in a small neighborhood:

$$M = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix}. \tag{1}$$

An interest point is identified using the determinant and the trace of M which measures the variation in a local neighborhood $R = det(M) - k \cdot tr(M)^2$, where k is constant. The interest points are marked by thresholding R after applying nonmaxima suppression (see Figure 2(a) for results). The source code for Harris detector is available at HarrisSrc. The same moment matrix M given in Equation (1) is used in the interest point detection step of the KLT tracking method. Interest point confidence, R, is computed using the minimum eigenvalue of M, $\lambda_{\min}$. Interest point candidates are selected by thresholding R. Among the candidate points, KLT eliminates the candidates that are spatially close to each other (Figure 2(b)). Implementation of the KLT detector is available at KLTSrc.

Quantitatively both Harris and KLT emphasize the intensity variations using very similar measures. For instance, R in Harris is related to the characteristic polynomial used for finding the eigenvalues of M: $\lambda^2 + det(M) - \lambda \cdot tr(M) = 0$, while KLT computes the eigenvalues directly. In practice, both of these methods find almost the same interest points. The only difference is the additional KLT criterion that enforces a predefined spatial distance between detected interest points.

In theory, the M matrix is invariant to both rotation and translation. However, it is not invariant to affine or projective transformations. In order to introduce robust detection of interest points under different transformations, Lowe [2004] introduced the SIFT (Scale Invariant Feature Transform) method which is composed of four steps. First, a scale space is constructed by convolving the image with Gaussian filters at different scales. Convolved images are used to generate difference-of-Gaussians (DoG) images. Candidate interest points are then selected from the minima and maxima of the DoG images across scales. The next step updates the location of each candidate by interpolating the color values using neighboring pixels. In the third step, low contrast candidates as well as the candidates along the edges are eliminated. Finally, remaining interest points are assigned orientations based on the peaks in the histograms of gradient directions in a small neighborhood around a candidate point. SIFT detector generates a greater number of interest points compared to other interest point detectors. This is due to the fact that the interest points at different scales and different resolutions (pyramid) are accumulated. Empirically, it has been shown in Mikolajczyk and Schmid [2003] that SIFT outperforms most point detectors and is more resilient to image deformations. Implementation of the SIFT detector is available at SIFTSrc.



(a) (b) (c) (d)

Fig. 3. Mixture of Gaussian modeling for background subtraction. (a) Image from a sequence in which a person is walking across the scene. (b) The mean of the highest-weighted Gaussians at each pixels position. These means represent the most temporally persistent per-pixel color and hence should represent the stationary background. (c) The means of the Gaussian with the second-highest weight; these means represent colors that are observed less frequently. (d) Background subtraction result. The foreground consists of the pixels in the current frame that matched a low-weighted Gaussian.

4.2. Background Subtraction

Object detection can be achieved by building a representation of the scene called the background model and then finding deviations from the model for each incoming frame. Any significant change in an image region from the background model signifies a moving object. The pixels constituting the regions undergoing change are marked for further processing. Usually, a connected component algorithm is applied to obtain connected regions corresponding to the objects. This process is referred to as the *background subtraction.*

Frame differencing of temporally adjacent frames has been well studied since the late 70s [Jain and Nagel 1979]. However, background subtraction became popular following the work of Wren et al. [1997]. In order to learn gradual changes in time, Wren et al. propose modeling the color of each pixel, $I(x, y)$, of a stationary background with a single 3D (Y, U, and V color space) Gaussian, $I(x, y) \sim N(\mu(x, y), \Sigma(x, y))$. The model parameters, the mean $\mu(x, y)$ and the covariance $\Sigma(x, y)$, are learned from the color observations in several consecutive frames. Once the background model is derived, for every pixel (x, y) in the input frame, the likelihood of its color coming from $N(\mu(x, y), \Sigma(x, y))$ is computed, and the pixels that deviate from the background model are labeled as the foreground pixels. However, a single Gaussian is not a good model for outdoor scenes [Gao et al. 2000] since multiple colors can be observed at a certain location due to repetitive object motion, shadows, or reflectance. A substantial improvement in background modeling is achieved by using multimodal statistical models to describe per-pixel background color. For instance, Stauffer and Grimson [2000] use a mixture of Gaussians to model the pixel color. In this method, a pixel in the current frame is checked against the background model by comparing it with every Gaussian in the model until a matching Gaussian is found. If a match is found, the mean and variance of the matched Gaussian is updated, otherwise a new Gaussian with the mean equal to the current pixel color and some initial variance is introduced into the mixture. Each pixel is classified based on whether the matched distribution represents the background process. Moving regions, which are detected using this approach, along with the background models are shown in Figure 3.

Another approach is to incorporate region-based (spatial) scene information instead of only using color-based information. Elgammal and Davis [2000] use nonparametric kernel density estimation to model the per-pixel background. During the subtraction process, the current pixel is matched not only to the corresponding pixel in the background model, but also to the nearby pixel locations. Thus, this method can handle camera jitter or small movements in the background. Li and Leung [2002] fuse the texture and color features to perform background subtraction over blocks of

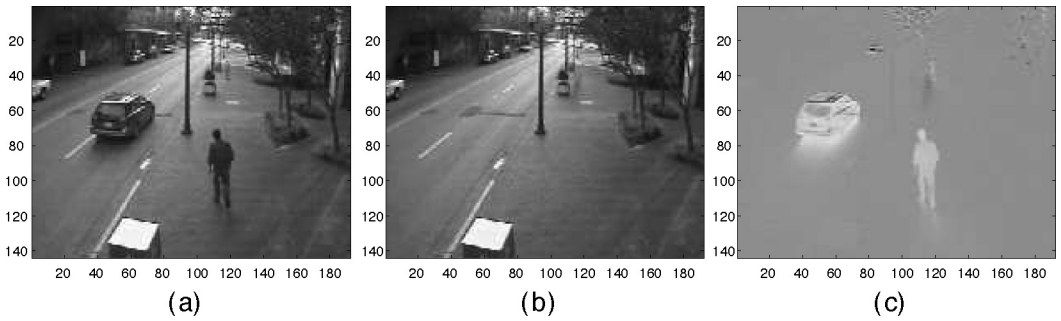


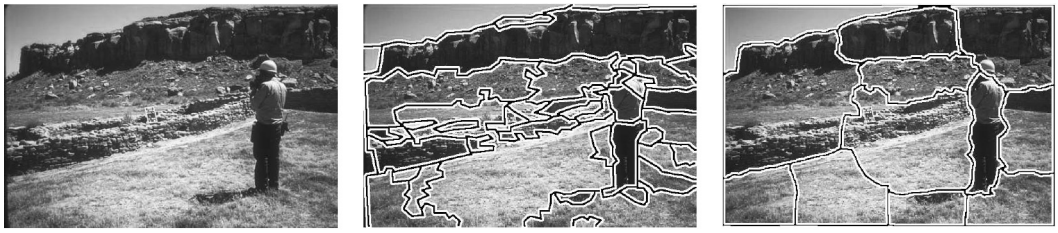
Fig. 4. Eigenspace decomposition-based background subtraction (space is constructed with objects in the FOV of camera): (a) an input image with objects, (b) reconstructed image after projecting input image onto the eigenspace, (c) difference image. Note that the foreground objects are clearly identifiable.

5×5 pixels. Since texture does not vary greatly with illumination changes, the method is less sensitive to illumination. Toyama et al. [1999] propose a three-tiered algorithm to deal with the background subtraction problem. In addition to the pixel-level subtraction, the authors use the region and the frame-level information. At the pixel level, the authors propose to use Wiener filtering to make probabilistic predictions of the expected background color. At the region level, foreground regions consisting of homogeneous color are filled in. At the frame level, if most of the pixels in a frame exhibit suddenly change, it is assumed that the pixel-based color background models are no longer valid. At this point, either a previously stored pixel-based background model is swapped in, or the model is reinitialized.

An alternate approach for background subtraction is to represent the intensity variations of a pixel in an image sequence as discrete states corresponding to the events in the environment. For instance, for tracking cars on a highway, image pixels can be in the background state, the foreground (car) state, or the shadow state. Rittscher et al. [2000] use Hidden Markov Models (HMM) to classify small blocks of an image as belonging to one of these three states. In the context of detecting light on and off events in a room, Stenger et al. [2001] use HMMs for the background subtraction. The advantage of using HMMs is that certain events, which are hard to model correctly using unsupervised background modeling approaches, can be learned using training samples.

Instead of modeling the variation of individual pixels, Oliver et al. [2000] propose a holistic approach using the eigenspace decomposition. For k input frames, $I^i : i = 1 \cdots k$, of size $n \times m$, a background matrix $\mathbf{B}$ of size $k \times l$ is formed by cascading m rows in each frame one after the other, where $l = (n \times m)$, and eigenvalue decomposition is applied to the covariance of $\mathbf{B}$, $\mathbf{C} = \mathbf{B}^T\mathbf{B}$. The background is then represented by the most descriptive η eigenvectors, $\mathbf{u}_i$, where $i < \eta < k$, that encompass all possible illuminations in the field of view (FOV). Thus, this approach is less sensitive to illumination. The foreground objects are detected by projecting the current image to the eigenspace and finding the difference between the reconstructed and actual images. We show detected object regions using the eigenspace approach in Figure 4.

One limitation of the aforementioned approaches is that they require a static background. This limitation is addressed by Monnet et al. [2003], and Zhong and Sclaroff [2003]. Both of these methods are able to deal with time-varying background (e.g., the waves on the water, moving clouds, and escalators). These methods model the image regions as autoregressive moving average (ARMA) processes which provide a way to learn and predict the motion patterns in a scene. An ARMA process is a time series model that is made up of sums of autoregressive and moving-average components, where an



(a) (b) (c)

Fig. 5. Segmentation of the image shown in (a), using mean-shift segmentation (b) and normalized cuts (c).

autoregressive process can be described as a weighted sum of its previous values and a white noise error.

In summary, most state-of-the-art tracking methods for fixed cameras, for example, Haritaoglu et al. [2000] and Collins et al. [2001] use background subtraction methods to detect regions of interest. This is because recent subtraction methods have the capabilities of modeling the changing illumination, noise, and the periodic motion of the background regions and, therefore, can accurately detect objects in a variety of circumstances. Moreover, these methods are computationally efficient. In practice, background subtraction provides incomplete object regions in many instances, that is, the objects may be spilled into several regions, or there may be holes inside the object since there are no guarantees that the object features will be different from the background features. The most important limitation of background subtraction is the requirement of stationary cameras. Camera motion usually distorts the background models. These methods can be applied to video acquired by mobile cameras by regenerating background models for small temporal windows, for instance, three frames, from scratch [Kanade et al. 1998] or by compensating sensor motion, for instance, creating background mosaics [Rowe and Blake 1996; Irani and Anandan 1998]. However, both of these solutions require assumptions of planar scenes and small motion in successive frames.

4.3. Segmentation

The aim of image segmentation algorithms is to partition the image into perceptually similar regions. Every segmentation algorithm addresses two problems, the criteria for a good partition and the method for achieving efficient partitioning [Shi and Malik 2000]. In this section, we will discuss recent segmentation techniques that are relevant to object tracking.

4.3.1. Mean-Shift Clustering. For the image segmentation problem, Comaniciu and Meer [2002] propose the mean-shift approach to find clusters in the joint spatial+color space, $[l, u, v, x, y]$, where $[l, u, v]$ represents the color and $[x, y]$ represents the spatial location. Given an image, the algorithm is initialized with a large number of hypothesized cluster centers randomly chosen from the data. Then, each cluster center is moved to the mean of the data lying inside the multidimensional ellipsoid centered on the cluster center. The vector defined by the old and the new cluster centers is called the *mean-shift vector*. The mean-shift vector is computed iteratively until the cluster centers do not change their positions. Note that during the mean-shift iterations, some clusters may get merged. In Figure 5(b), we show the segmentation using the mean-shift approach generated using the source code available at MeanShiftSegmentSrc.

Mean-shift clustering is scalable to various other applications such as edge detection, image regularization [Comaniciu and Meer 2002], and tracking [Comaniciu et al. 2003].

Mean-shift based segmentation requires fine tuning of various parameters to obtain better segmentation, for instance, selection of the color and spatial kernel bandwidths, and the threshold for the minimum size of the region considerably effect the resulting segmentation.

4.3.2. Image Segmentation Using Graph-Cuts. Image segmentation can also be formulated as a graph partitioning problem, where the vertices (pixels), $\mathbf{V} = \{u, v, \ldots\}$, of a graph (image), $\mathbf{G}$, are partitioned into N disjoint subgraphs (regions), $A_i, \bigcup_{i=1}^{N} A_i = \mathbf{V}$, $A_i \cap A_j = \emptyset$, $i \neq j$, by pruning the weighted edges of the graph. The total weight of the pruned edges between two subgraphs is called a *cut*. The weight is typically computed by color, brightness, or texture similarity between the nodes. Wu and Leahy [1993] use the minimum cut criterion, where the goal is to find the partitions that minimize a cut. In their approach, the weights are defined based on the color similarity. One limitation of minimum cut is its bias toward oversegmenting the image. This effect is due to the increase in cost of a cut with the number of edges going across the two partitioned segments.

Shi and Malik [2000] propose the *normalized cut* to overcome the oversegmentation problem. In their approach, the cut not only depends on the sum of edge weights in the cut, but also on the ratio of the total connection weights of nodes in each partition to all nodes of the graph. For image-based segmentation, the weights between the nodes are defined by the product of the color similarity and the spatial proximity. Once the weights between each pair of nodes are computed, a weight matrix $\mathbf{W}$ and a diagonal matrix $\mathbf{D}$, where $\mathbf{D}_{i,i} = \sum_{j=1}^{N} \mathbf{W}_{i,j}$, are constructed. The segmentation is performed first by computing the eigenvectors and the eigenvalues of the generalized eigensystem $(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$, then the second-smallest eigenvector is used to divide the image into two segments. For each new segment, this process is recursively performed until a threshold is reached. In Figure 5(c), we show the segmentation results obtained by the normalized cuts approach.

In normalized cuts-based segmentation, the solution to the generalized eigensystem for large images can be expensive in terms of processing and memory requirements. However, this method requires fewer manually selected parameters, compared to mean-shift segmentation. Normalized cuts have also been used in the context of tracking object contours [Xu and Ahuja 2002].

4.3.3. Active Contours. In an active contour framework, object segmentation is achieved by evolving a closed contour to the object's boundary, such that the contour tightly encloses the object region. Evolution of the contour is governed by an energy functional which defines the fitness of the contour to the hypothesized object region. Energy functional for contour evolution has the following common form:

$$E(\Gamma) = \int_0^1 E_{int}(\mathbf{v}) + E_{im}(\mathbf{v}) + E_{ext}(\mathbf{v}) ds, \tag{2}$$

where s is the arc-length of the contour Γ, E_{int} includes regularization constraints, E_{im} includes appearance-based energy, and E_{ext} specifies additional constraints. E_{int} usually includes a curvature term, first-order $(\nabla \mathbf{v})$ or second-order $(\nabla^2 \mathbf{v})$ continuity terms to find the shortest contour. Image-based energy, E_{im}, can be computed locally or globally. Local information is usually in the form of an image gradient and is evaluated around the contour [Kass et al. 1988; Caselles et al. 1995]. In contrast, global features are computed inside and outside of the object region. Global features include color [Zhu and Yuille 1996; Yilmaz et al. 2004; Ronfard 1994] and texture [Paragios and Deriche 2002].

Different researchers have used different energy terms in Equation (2). In 1995, Caselles et al. exclude the E_{ext} and use only the image gradient as the image energy $E_{im} = g(|\nabla I|)$, where g is the sigmoid function. Compared to the gradient, a function of the gradient defines the object contour as a geodesic curve in the Riemannian space [Caselles et al. 1995]. However, image gradients provide very local information and are sensitive to local minima. In order to overcome this problem, researchers introduced region-based image energy terms. In 1996, Zhu and Yuille proposed using region information instead of the image gradient. However, the use of regional terms in the energy functional does not result in good localization of the object contour. Recently, methods that combine both region-based and gradient-based image energy have become popular. Paragios and Deriche [2002] propose using a convex combination of the gradient and region-based energies, $E_{image} = \lambda E_{boundary} + (1 - \lambda)E_{region}$. In particular, the authors model the appearance in E_{region} by mixture of Gaussians. Contour evolution is first performed globally, then locally by varying the α from 0 to 1 at each iteration.

An important issue in contour-based methods is the contour initialization. In image gradient-based approaches, a contour is typically placed outside the object region and shrunk until the object boundary is encountered [Kass et al. 1988; Caselles et al. 1995]. This constraint is relaxed in region-based methods such that the contour can be initialized either inside or outside the object so that the contour can either expand or shrink, respectively, to fit the object boundary. However, these approaches require prior object or background knowledge [Paragios and Deriche 2002]. Using multiple frames or a reference frame, initialization can be performed without building region priors. For instance, in Paragios and Deriche [2000], the authors use background subtraction to initialize the contour.

Besides the selection of the energy functional and the initialization, another important issue is selecting the right contour representation. Object contour, Γ, can be represented either explicitly (control points, $\mathbf{v}$) or implicitly (level sets, ϕ). In the explicit representation, the relation between the control points are defined by spline equations. In the level sets representation, the contour is represented on a spatial grid which encodes the signed distances of the grids from the contour with opposite signs for the object and the background regions. The contour is implicitly defined as the zero crossings in the level set grid. The evolution of the contour is governed by changing the grid values according to the energy computed using Equation (2), evaluated at each grid position. The changes in the grid values result in new zero crossings, hence, a new contour positions (more details are given in Section 5.3). The source code for generic level sets, which can be used for various applications by specifying the contour evolution speed, for instance, segmentation, tracking, heat flow etc., is available at LevelSetSrc. The most important advantage of implicit representation over the explicit representation is its flexibility in allowing topology changes (split and merge).

4.4. Supervised Learning

Object detection can be performed by learning different object views automatically from a set of examples by means of a supervised learning mechanism. Learning of different object views waives the requirement of storing a complete set of templates. Given a set of learning examples, supervised learning methods generate a function that maps inputs to desired outputs. A standard formulation of supervised learning is the classification problem where the learner approximates the behavior of a function by generating an output in the form of either a continuous value, which is called *regression*, or a class label, which is called *classification*. In context of object detection, the learning examples are composed of pairs of object features and an associated object class where both of these quantities are manually defined.

Selection of features plays an important role in the performance of the classification, hence, it is important to use a set of features that discriminate one class from the other. In addition to the features discussed in Section 3, it is also possible to use other features such as object area, object orientation, and object appearance in the form of a density function, for example, histogram. Once the features are selected, different appearances of an object can be learned by choosing a supervised learning approach. These learning approaches include, but are not limited to, neural networks [Rowley et al. 1998], adaptive boosting [Viola et al. 2003], decision trees [Grewe and Kak 1995], and support vector machines [Papageorgiou et al. 1998]. These learning methods compute a hypersurface that separates one object class from the other in a high dimensional space.

Supervised learning methods usually require a large collection of samples from each object class. Additionally, this collection must be manually labeled. A possible approach to reducing the amount of manually labeled data is to accompany cotraining with supervised learning [Blum and Mitchell 1998]. The main idea behind cotraining is to train two classifiers using a small set of labeled data where the features used for each classifier are independent. After training is achieved, each classifier is used to assign unlabeled data to the training set of the other classifier. It was shown that, starting from a small set of labeled data with two sets of statistically independent features, cotraining can provide a very accurate classification rule [Blum and Mitchell 1998]. Cotraining has been successfully used to reduce the amount of manual interaction required for training in the context of adaboost [Levin et al. 2003] and support vector machines [Kockelkorn et al. 2003]. Following we will discuss the adaptive boosting and the support vector machines due to their applicability to object tracking.

4.4.1. Adaptive Boosting. Boosting is an iterative method of finding a very accurate classifier by combining many base classifiers, each of which may only be moderately accurate [Freund and Schapire 1995]. In the training phase of the Adaboost algorithm, the first step is to construct an initial distribution of weights over the training set. The boosting mechanism then selects a base classifier that gives the least error, where the error is proportional to the weights of the misclassified data. Next, the weights associated with the data misclassified by the selected base classifier are increased. Thus the algorithm encourages the selection of another classifier that performs better on the misclassified data in the next iteration. For interested readers tutorials on boosting are available at http://www.boosting.org.

In the context of object detection, weak classifiers can be simple operators such as a set of thresholds, applied to the object features extracted from the image. In 2003, Viola et al. used the Adaboost framework to detect pedestrians. In their approach, perceptrons were chosen as the weak classifiers which are trained on image features extracted by a combination of spatial and temporal operators. The operators for feature extraction are in the form of simple rectangular filters, and are shown in Figure 6. The operators in the temporal domain are in the form of frame differencing which encode some form of motion information. Frame differencing, when used as an operator in the temporal domain, reduces the number of false detections by enforcing object detection in the regions where the motion occurs.

4.4.2. Support Vector Machines. As a classifier, Support Vector Machines (SVM) are used to cluster data into two classes by finding the maximum marginal hyperplane that separates one class from the other [Boser et al. 1992]. The margin of the hyperplane, which is maximized, is defined by the distance between the hyperplane and the closest data points. The data points that lie on the boundary of the margin of the hyperplane are called the support vectors. In the context of object detection, these classes correspond

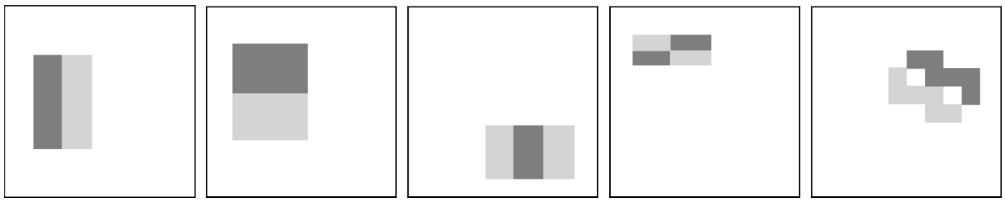


Fig. 6. A set of rectangular filters used by Viola et al. [2003] to extract features used in the Adaboost framework. Each filter is composed of three regions: white, light gray, and dark gray, with associated weights 0, −1, and 1 respectively. In order to compute the feature in a window, these filters are convolved with the image.

to the object class (positive samples) and the nonobject class (negative samples). From manually generated training examples labeled as object and nonobject, computation of the hyperplane from among an infinite number of possible hyperplanes is carried out by means of quadratic programming.

Despite being a linear classifier, SVM can also be used as a nonlinear classifier by applying the kernel trick to the input feature vector extracted from the input. Application of the kernel trick to a set of data that is not linearly separable, transforms the data to a higher dimensional space which is likely to be separable. The kernels used for kernel trick are polynomial kernels or radial basis functions, for example, Gaussian kernel and two-layer perceptron, for instance, a sigmoid function. However, the selection of the right kernel for the problem at hand is not easy. Once a kernel is chosen, one has to test the classification performance for a set of parameters which may not work as well when new observations are introduced to the sample set.

In the context of object detection, Papageorgiou et al. [1998] use SVM for detecting pedestrians and faces in images. The features used to discriminate between the classes are extracted by applying Haar wavelets to the sets of positive and negative training examples. In order to reduce the search space, temporal information is utilized by computing the optical flow field in the image. Particularly, the discontinuities in the optical flow field are used to initiate the search for possible objects resulting in a decreased number of false positives.

5. OBJECT TRACKING

The aim of an object tracker is to generate the trajectory of an object over time by locating its position in every frame of the video. Object tracker may also provide the complete region in the image that is occupied by the object at every time instant. The tasks of detecting the object and establishing correspondence between the object instances across frames can either be performed separately or jointly. In the first case, possible object regions in every frame are obtained by means of an object detection algorithm, and then the tracker corresponds objects across frames. In the latter case, the object region and correspondence is jointly estimated by iteratively updating object location and region information obtained from previous frames. In either tracking approach, the objects are represented using the shape and/or appearance models described in Section 2. The model selected to represent object shape limits the type of motion or deformation it can undergo. For example, if an object is represented as a point, then only a translational model can be used. In the case where a geometric shape representation like an ellipse is used for the object, parametric motion models like affine or projective transformations are appropriate. These representations can approximate the motion of rigid objects in the scene. For a nonrigid object, silhouette or contour is the most descriptive representation and both parametric and nonparametric models can be used to specify their motion.

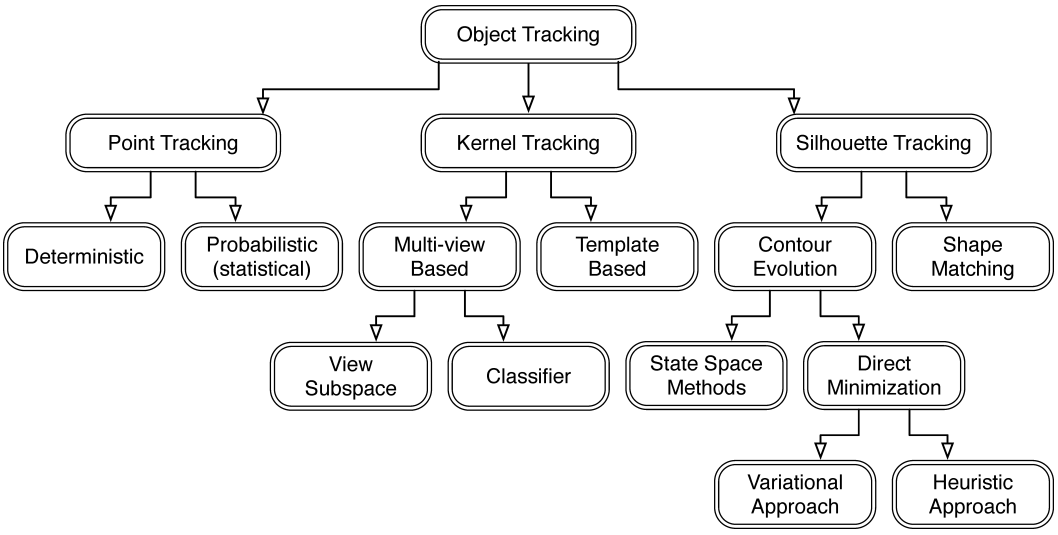


Fig. 7. Taxonomy of tracking methods.

Table II. Tracking Categories

Categories	Representative Work
Point Tracking	
• Deterministic methods	MGE tracker [Salari and Sethi 1990], GOA tracker [Veenman et al. 2001].
• Statistical methods	Kalman filter [Broida and Chellappa 1986], JPDAF [Bar-Shalom and Foreman 1988], PMHT [Streit and Luginbuhl 1994].
Kernel Tracking	
• Template and density based appearance models	Mean-shift [Comaniciu et al. 2003], KLT [Shi and Tomasi 1994], Layering [Tao et al. 2002].
• Multi-view appearance models	Eigentracking [Black and Jepson 1998], SVM tracker [Avidan 2001].
Silhouette Tracking	
• Contour evolution	State space models [Isard and Blake 1998], Variational methods [Bertalmio et al. 2000], Heuristic methods [Ronfard 1994].
• Matching shapes	Hausdorff [Huttenlocher et al. 1993], Hough transform [Sato and Aggarwal 2004], Histogram [Kang et al. 2004].

In view of the aforementioned discussion, we provide a taxonomy of tracking methods in Figure 7. Representative work for each category is tabulated in Table II. We now briefly introduce the main tracking categories, followed by a detailed section on each category.

—*Point Tracking*. Objects detected in consecutive frames are represented by points, and the association of the points is based on the previous object state which can include object position and motion. This approach requires an external mechanism to detect the objects in every frame. An example of object correspondence is shown in Figure 8(a).

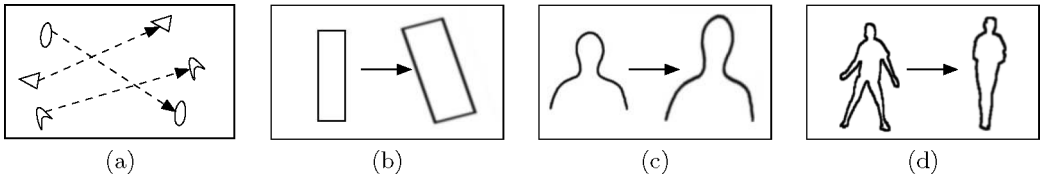


Fig. 8. (a) Different tracking approaches. Multipoint correspondence, (b) parametric transformation of a rectangular patch, (c, d) Two examples of contour evolution.

—*Kernel Tracking*. Kernel refers to the object shape and appearance. For example, the kernel can be a rectangular template or an elliptical shape with an associated histogram. Objects are tracked by computing the motion of the kernel in consecutive frames (Figure 8(b)). This motion is usually in the form of a parametric transformation such as translation, rotation, and affine.

—*Silhouette Tracking*. Tracking is performed by estimating the object region in each frame. Silhouette tracking methods use the information encoded inside the object region. This information can be in the form of appearance density and shape models which are usually in the form of edge maps. Given the object models, silhouettes are tracked by either shape matching or contour evolution (see Figure 8(c), (d)). Both of these methods can essentially be considered as object segmentation applied in the temporal domain using the priors generated from the previous frames.

5.1. Point Tracking

Tracking can be formulated as the correspondence of detected objects represented by points across frames. Point correspondence is a complicated problem-specially in the presence of occlusions, misdetections, entries, and exits of objects. Overall, point correspondence methods can be divided into two broad categories, namely, deterministic and statistical methods. The deterministic methods use *qualitative motion heuristics* [Veenman et al. 2001] to constrain the correspondence problem. On the other hand, probabilistic methods explicitly take the object measurement and take uncertainties into account to establish correspondence.

5.1.1. Deterministic Methods for Correspondence. Deterministic methods for point correspondence define a cost of associating each object in frame $t-1$ to a single object in frame t using a set of motion constraints. Minimization of the correspondence cost is formulated as a combinatorial optimization problem. A solution, which consists of one-to-one correspondences (Figure 9(b)) among all possible associations (Figure 9(a)), can be obtained by optimal assignment methods, for example, Hungarian algorithm, [Kuhn 1955] or greedy search methods. The correspondence cost is usually defined by using a combination of the following constraints.

—*Proximity* assumes the location of the object would not change notably from one frame to other (see Figure 10(a)).

—*Maximum velocity* defines an upper bound on the object velocity and limits the possible correspondences to the circular neighborhood around the object (see Figure 10(b)).

—*Small velocity change* (smooth motion) assumes the direction and speed of the object does not change drastically (see Figure 10(c)).

—*Common motion* constrains the velocity of objects in a small neighborhood to be similar (see Figure 10(d)). This constraint is suitable for objects represented by multiple points.

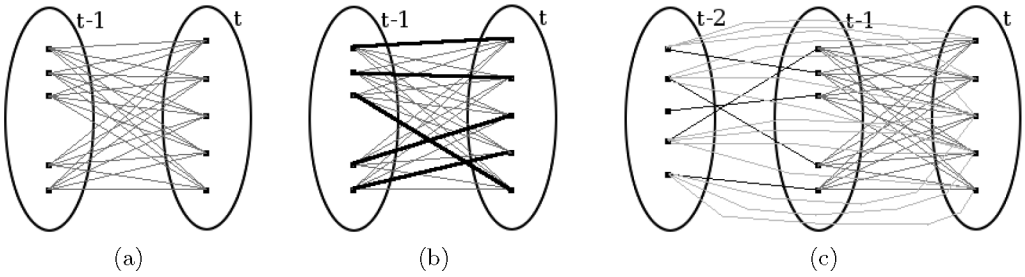


Fig. 9. Point correspondence. (a) All possible associations of a point (object) in frame $t-1$ with points (objects) in frame t, (b) unique set of associations plotted with bold lines, (c) multiframe correspondences.

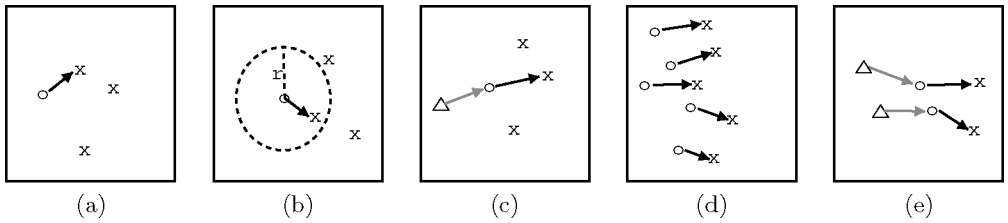


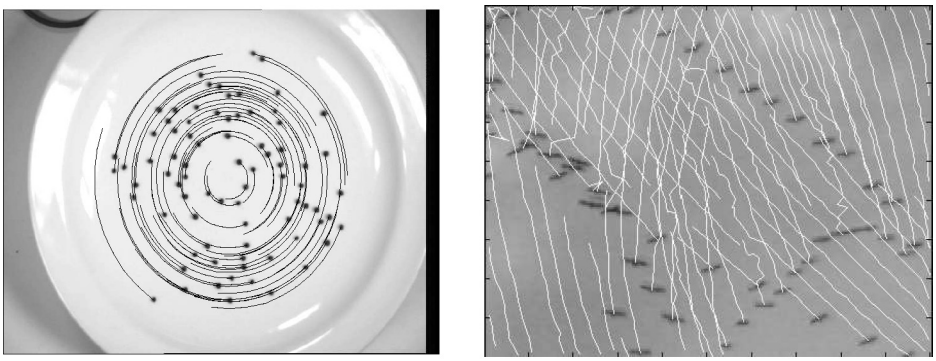
Fig. 10. Different motion constraints. (a) proximity, (b) maximum velocity (r denotes radius), (c) small velocity-change, (d) common motion, (e) rigidity constraints. $\triangle$ denotes object position at frame $t-2$, $\circ$ denotes object position at frame $t-1$, and finally $\times$ denotes object position at frame t.

—*Rigidity* assumes that objects in the 3D world are rigid, therefore, the distance between any two points on the actual object will remain unchanged (see Figure 10(e)).

—*Proximal uniformity* is a combination of the proximity and the small, velocity change constraints.

We should, however, note that these constraints are not specific to the deterministic methods, and they can also be used in the context of point tracking using statistical methods.

Here we present a sample of different methods proposed in the literature in this category. Sethi and Jain [1987] solve the correspondence by a greedy approach based on the proximity and rigidity constraints. Their algorithm considers two consecutive frames and is initialized by the nearest neighbor criterion. The correspondences are exchanged iteratively to minimize the cost. A modified version of the same algorithm which computes the correspondences in the backward direction (from the last frame to the first frame) in addition to the forward direction is also analyzed. This method cannot handle occlusions, entries, or exits. Salari and Sethi [1990] handle these problems, by first establishing correspondence for the detected points and then extending the tracking of the missing objects by adding a number of hypothetical points. Rangarajan and Shah [1991] propose a greedy approach, which is constrained by proximal uniformity. Initial correspondences are obtained by computing optical flow in the first two frames. The method does not address entry and exit of objects. If the number of detected points decrease, occlusion or misdetection is assumed. Occlusion is handled by establishing the correspondence for the detected objects in the current frame. For the remaining objects, position is predicted based on a constant velocity assumption. In the work by Intille et al. [1997], which uses a slightly modified version of Rangarajan



(a) (b)

Fig. 11. Results of two point correspondence algorithms. (a) Tracking using the algorithm proposed by Veenman et al. [2001] in the rotating dish sequence color segmentation was used to detect black dots on a white dish (©2001 IEEE). (b) Tracking birds using the algorithm proposed by Shafique and Shah [2003]; birds are detected using background subtraction (©2003 IEEE).

and Shah [1991] for matching object centroids, the objects are detected by using background subtraction. The authors explicitly handle the change in the number of objects by examining specific regions in the image, for example, a door, to detect entries/exits before computing the correspondence.

Veenman et al. [2001] extend the work of Sethi and Jain [1987], and Rangarajan and Shah [1991] by introducing the *common motion* constraint for correspondence. The common motion constraint provides a strong constraint for coherent tracking of points that lie on the same object; however, it is not suitable for points lying on isolated objects moving in different directions. The algorithm is initialized by generating the initial tracks using a two-pass algorithm, and the cost function is minimized by Hungarian assignment algorithm in two consecutive frames. This approach can handle occlusion and misdetection errors, however, it is assumed that the number of objects are the same throughout the sequence, that is, no object entries or exits. See Figure 11(a) for tracking results.

Shafique and Shah [2003] propose a multiframe approach to preserve temporal coherency of the speed and position (Figure 9(c)). They formulate the correspondence as a graph theoretic problem. Multiple frame correspondence relates to finding the best unique path $P_i = \{\mathbf{x}^0, \ldots, \mathbf{x}^k\}$ for each point (the superscript represents the frame number). For misdetected or occluded objects, the path will consist of missing positions in corresponding frames. The directed graph, which is generated using the points in k frames, is converted to a bipartite graph by splitting each node (object) into two ($+$ and $-$) nodes and representing directed edges as undirected edges from $+$ to $-$ nodes. The correspondence is then established by a greedy algorithm. They use a window of frames during point correspondence to handle occlusions whose durations are shorter than the temporal window used to perform matching. See Figure 11(b) for results on this algorithm for the tracking of birds.

5.1.2. Statistical Methods for Correspondence. Measurements obtained from video sensors invariably contain noise. Moreover, the object motions can undergo random perturbations, for instance, maneuvering vehicles. Statistical correspondence methods solve these tracking problems by taking the measurement and the model uncertainties into account during object state estimation. The statistical correspondence methods use the

state space approach to model the object properties such as position, velocity, and acceleration. Measurements usually consist of the object position in the image, which is obtained by a detection mechanism. Followings, we will discuss the state estimation methods in the context of point tracking, however, it should be noted that these methods can be used in general to estimate the state of any time varying system. For example, these methods have extensively been used for tracking contours [Isard and Blake 1998], activity recognition [Vaswani et al. 2003], object identification [Zhou et al. 2003], and structure from motion [Matthies et al. 1989].

Consider a moving object in the scene. The information representing the object, for example, location, is defined by a sequence of states $X^t : t = 1, 2, \ldots$. The change in state over time is governed by the dynamic equation,

$$X^t = f^t(X^{t-1}) + W^t, \tag{3}$$

where $W^t : t = 1, 2, \ldots$ is white noise. The relationship between the measurement and the state is specified by the measurement equation $Z^t = h^t(X^t, N^t)$, where N^t is the white noise and is independent of W^t. The objective of tracking is to estimate the state X^t given all the measurements up to that moment or, equivalently, to construct the probability density function $p(X^t|Z^{1,\ldots,t})$. A theoretically optimal solution is provided by a recursive Bayesian filter which solves the problem in two steps. The *prediction* step uses a dynamic equation and the already computed pdf of the state at time $t-1$ to derive the prior pdf of the current state, that is, $p(X^t|Z^{1,\ldots,t-1})$. Then, the *correction* step employs the likelihood function $p(Z^t|X^t)$ of the current measurement to compute the posterior pdf $p(X^t|Z^{1,\ldots,t})$. In the case where the measurements only arise due to the presence of a single object in the scene, the state can be simply estimated by the two steps as defined. On the other hand, if there are multiple objects in the scene, measurements need to be associated with the corresponding object states. We now discuss the two cases.

5.1.2.1. Single Object State Estimation. For the single object case, if f^t and h^t are linear functions and the initial state X^1 and noise have a Gaussian distribution, then the optimal state estimate is given by the Kalman Filter. In the general case, that is, object state is not assumed to be a Gaussian, state estimation can be performed using particle filters [Tanizaki 1987].

—*Kalman Filters*. A Kalman filter is used to estimate the state of a linear system where the state is assumed to be distributed by a Gaussian. Kalman filtering is composed of two steps, prediction and correction. The prediction step uses the state model to predict the new state of the variables:

$$\begin{aligned} \overline{X}^t &= \mathbf{D}X^{t-1} + W, \\ \overline{\Sigma}^t &= \mathbf{D}\Sigma^{t-1}\mathbf{D}^T + Q^t, \end{aligned}$$

where $\overline{X^t}$ and $\overline{\Sigma^t}$ are the state and the covariance predictions at time t. $\mathbf{D}$ is the state transition matrix which defines the relation between the state variables at time t and $t-1$. Q is the covariance of the noise W. Similarly, the correction step uses the current observations Z^t to update the object's state:

$$K^t = \overline{\Sigma}^t\mathbf{M}^T[\mathbf{M}\overline{\Sigma}^t\mathbf{M}^T + R^t]^{-1}, \tag{4}$$

$$X^t = \overline{X}^t + K^t\underbrace{[Z^t - \mathbf{M}\overline{X}^t]}_{v}, \tag{5}$$

$$\Sigma^t = \overline{\Sigma}^t - K^t\mathbf{M}\overline{\Sigma}^t,$$

where v is called the innovation, $\mathbf{M}$ is the measurement matrix, K is the Kalman gain, which is the Riccati Equation (4) used for propagation of the state models. Note that the updated state, X^t is still distributed by a Gaussian. In case the functions f^t and h^t are nonlinear, they can be linearized using the Taylor series expansion to obtain the extended Kalman filter [Bar-Shalom and Foreman 1988]. Similar to the Kalman filter, the extended Kalman filter assumes that the state is distributed by a Gaussian.

The Kalman filter has been extensively used in the vision community for tracking. Broida and Chellappa [1986] used the Kalman filter to track points in noisy images. In stereo camera-based object tracking, Beymer and Konolige [1999] use the Kalman filter for predicting the object's position and speed in $x - z$ dimensions. Rosales and Sclaroff [1999] use the extended Kalman filter to estimate 3D trajectory of an object from 2D motion. A Matlab toolbox for Kalman filtering is available at KalmanSrc.

—*Particle Filters*. One limitation of the Kalman filter is the assumption that the state variables are normally distributed (Gaussian). Thus, the Kalman filter will give poor estimations of state variables that do not follow Gaussian distribution. This limitation can be overcome by using particle filtering [Tanizaki 1987]. In particle filtering, the conditional state density $p(X_t|Z_t)$ at time t is represented by a set of samples $\{s_t^{(n)} : n = 1, \ldots, N\}$ (particles) with weights $\pi_t^{(n)}$ (sampling probability). The weights define the importance of a sample, that is, its observation frequency [Isard and Blake 1998]. To decrease computational complexity, for each tuple $(s^{(n)}, \pi^{(n)})$, a cumulative weight $c^{(n)}$ is also stored, where $c^{(N)} = 1$. The new samples at time t are drawn from $\mathbf{S}_{t-1} = \{(s_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}) : n = 1, \ldots, N\}$ at the previous time $t - 1$ step based on different sampling schemes [MacKay 1998]. The most common sampling scheme is *importance sampling* which can be stated as follows.

(1) Selection. Select N random samples $\hat{s}_t^{(n)}$ from $\mathbf{S}_{t-1}$ by generating a random number $r \in [0, 1]$, finding the smallest j such that $c_{t-1}^{(j)} > r$ and setting $\hat{s}_t^{(n)} = s_{t-1}^{(j)}$.

(2) Prediction. For each selected sample $\hat{s}_t^{(n)}$, generate a new sample by $s_t^{(n)} = f(\hat{s}_t^{(n)}, W_t^{(n)})$, where $W_t^{(n)}$ is a zero mean Gaussian error and f is a non-negative function, i.e. $f(s) = s$.

(3) Correction. Weights $\pi_t^{(n)}$ corresponding to the new samples $s_t^{(n)}$ are computed using the measurements z_t by $\pi_t^{(n)} = p(z_t|x_t = s_t^{(n)})$, where $p(.)$ can be modeled as a Gaussian density.

Using the new samples $\mathbf{S}_t$, one can estimate the new object position by $\varepsilon_t = \sum_{n=1}^{N} \pi_t^{(n)} f(s_t^{(n)}, W)$. Particle filter-based trackers can be initialized by either using the first measurements, $s_0^{(n)} \sim X_0$, with weight $\pi_0^{(n)} = \frac{1}{N}$ or by training the system using sample sequences. In addition to keeping track of the best particles, an additional resampling is usually employed to eliminate samples with very low weights. Note that the posterior density does not have to be a Gaussian. Particle filters recently became popular in computer vision. They are especially used for object detection and tracking. A Matlab toolbox for tracking using particle filtering is available at ParticleFltSrc.

Note that the Kalman filter and particle filter described assume a single measurement at each time instant, that is, the state of a single object is estimated. Tracking multiple objects requires a joint solution of data association and state estimation problems.

5.1.2.2. Multiobject Data Association and State Estimation. When tracking multiple objects using Kalman or particle filters, one needs to deterministically associate the most

likely measurement for a particular object to that object's state, that is, the correspondence problem needs to be solved before these filters can be applied. The simplest method to perform correspondence is to use the nearest neighbor approach. However, if the objects are close to each other, then there is always a chance that the correspondence is incorrect. An incorrectly associated measurement can cause the filter to fail to converge. There exist several statistical data association techniques to tackle this problem. A detailed review of these techniques can be found in the book by Fortmann and Bar-Shalom [1988] or in the survey by Cox [1993]. Joint Probability Data Association Filtering (JPDAF) and Multiple Hypothesis Tracking (MHT) are two widely used techniques for data association. We give a brief description of these techniques in the following.

—*Joint Probability Data Association Filter*. Let a track be defined as a sequence of measurements that are assumed to originate from the same object. Suppose we have N tracks and, at time t, $Z(t) = z_1(t), \cdots z_{m_t}(t)$ are the m measurements. We need to assign these measurements to the existing tracks. Let η be a set of assignments. It is assumed that the number of tracks will remain constant over time. Let $v_{i,l}$ be the innovation (see the discussion on the Kalman Filter) associated with the track l due to the measurement z_i. The JPDAF associates all measurements with each track. The combined weighted innovation is given by

$$v^l = \sum_{i=1}^{m_k} \beta_i^l v_{i,l}, \tag{6}$$

where β_i^l is the posterior probability that the measurement i originated from the object associated with track l and is given as:

$$\beta_i^l = \sum_{\eta} P[\eta_l(k)|Z^t]\tau_{i,l}(\eta), \tag{7}$$

where $\tau_{i,l}$ is the indicator variable, with $i = 1, \ldots, m_k$ and $l = 1, \ldots, N$. It is equal to one if the measurement $z_i(k)$ is associated with track l, otherwise it is zero. The weighted innovation given in Equation (6) can be plugged into the Kalman filter update Equations (5) for each track l.

JPDAF is used by Chang and Aggarwal [1991] to perform 3D structure reconstruction from a video sequence. Rasmussen and Hager [2001] use a constrained JPDAF filter to track regions. The major limitation of the JPDAF algorithm is its inability to handle new objects entering the field of view (FOV) or already tracked objects exiting the FOV. Since the JPDAF algorithm performs data association of a fixed number of objects tracked over two frames, serious errors can arise if there is a change in the number of objects. The MHT algorithm, which is explained next, does not have this shortcoming.

—*Multiple Hypothesis Tracking* (*MHT*). If motion correspondence is established using only two frames, there is always a finite chance of an incorrect correspondence. Better tracking results can be obtained if the correspondence decision is deferred until several frames have been examined. The MHT algorithm maintains several correspondence hypotheses for each object at each time frame [Reid 1979]. The final track of the object is the most likely set of correspondences over the time period of its observation. The algorithm has the ability to create new tracks for objects entering the FOV and terminate tracks for objects exiting the FOV. It can also handle occlusions, that is, continuation of a track even if some of the measurements from an object are missing.

MHT is an iterative algorithm. An iteration begins with a set of current track hypotheses. Each hypothesis is a collection of disjoint tracks. For each hypothesis, a prediction of each object's position in the next frame is made. The predictions are then compared

with actual measurements by evaluating a distance measure. A set of correspondences (associations) are established for each hypothesis based on the distance measure which introduces new hypotheses for the next iteration. Each new hypothesis represents a new set of tracks based on the current measurements. Note that each measurement can belong to a new object entering the FOV, a previously tracked object, or a spurious measurement. Moreover, a measurement may not be assigned to an object because the object may have exited the FOV, or a measurement corresponding to an object may not be obtained. The latter happens because either the object is occluded or it is not detected due to noise.

Note that MHT makes associations in a deterministic sense and exhaustively enumerates all possible associations. To reduce the computational load, Streit and Luginbuhl [1994] propose a probabilistic MHT (PMHT) in which the associations are considered to be statistically independent random variables and thus there is no requirement for exhaustive enumeration of associations. Recently, particle filters that handle multiple measurements to track multiple objects have been proposed by Hue et al. [2002]. In their method, data association is handled in a similar way as in PMHT, however, the state estimation is achieved through particle filters.

The MHT algorithm is computationally exponential both in memory and time. To overcome this limitation, Cox and Hingorani [1996] use Murty's [1968] algorithm to determine k-best hypotheses in polynomial time for tracking interest points. Cham and Rehg [1999] use the multiple hypothesis framework to track the complete human body.

5.1.3. Discussion. Point tracking methods can be evaluated on the basis of whether they generate correct point trajectories. Given a ground truth, the performance can be evaluated by computing *precision* and *recall* measures. In the context of point tracking, precision and recall measures can be defined as:

$$\text{precision} = \frac{\text{\# of correct correspondences}}{\text{\# of established correspondences}}, \tag{8}$$

$$\text{recall} = \frac{\text{\# of correct correspondences}}{\text{\# of actual correspondences}}, \tag{9}$$

where actual correspondences denote the correspondences available in the ground truth. Additionally, a qualitative comparison of object trackers can be made based on their ability to

—deal with entries of new objects and exits of existing objects,

—handle the missing observations (occlusion), and

—provide an optimal solution to the cost function minimization problem used for establishing correspondence.

In Table III, we provide a qualitative comparison based on these properties.

One important issue in the context of point trackers is the handling of missing or noisy observations. To address these problems, deterministic point trackers often use a combination of motion-based constraints addressed in Section 5.1.1, that is, common motion [Veenman et al. 2001] or proximal uniformity [Rangarajan and Shah 1991]. Statistical point tracking methods explicitly handle noise by taking model uncertainties into consideration. These uncertainties are usually assumed to be in the form of normally distributed noise. However, the assumption that measurements are normally distributed around their predicted position may not hold. Moreover, in many cases, the noise parameters are not known. In the case of valid assumptions on distributions and

Table III. Qualitative Comparison of Point Trackers (**#**: number of objects, **M**: multiple objects, **S** single object. Symbols $\surd$ and $\times$ denote whether the tracker can or cannot handle occlusions, object entries object exits, and provide the optimal solution.)

	#	Entry	Exit	Occlusion	Optimal
GE [Sethi and Jain 1987]	M	$\times$	$\times$	$\times$	$\times$
MGE [Salari and Sethi 1990]	M	$\surd$	$\surd$	$\surd$	$\times$
GOA [Veenman et al. 2001]	M	$\times$	$\times$	$\surd$	$\surd$
MFT [Shafique and Shah 2003]	M	$\surd$	$\surd$	$\surd$	$\times$
Kalman [Bar-Shalom and Foreman 1988]	S	$\times$	$\times$	$\times$	$\surd$
JPDAF [Bar-Shalom and Foreman 1988]	M	$\times$	$\times$	$\times$	$\times$
MHT [Cox and Hingorani 1996]	M	$\surd$	$\surd$	$\surd$	$\surd$

noise, Kalman filters [Bar-Shalom and Foreman 1988] and MHT [Reid 1979] give optimal solutions. Another possible approach to handling noise and missing observations is to enforce constraints that define the object's 3D structure. For instance, multibody factorization methods can be used for handling noisy observations by enforcing the object points to fit into the 3D object shape. This is addressed for the nonrigid object by Bregler et al. [Torresani and Bregler 2002; Bregler et al. 2000] where the authors first define a set of shape bases from a set of reliable tracks which has minimum or no appearance error on the points trajectory. Computed shape basis then serves as a constraint on the remaining point trajectories that are labeled as unreliable.

Point trackers are suitable for tracking very small objects which can be represented by a single point (single point representation). Multiple points are needed to track larger objects. In the context of tracking objects using multiple points, automatic clustering of points that belong to the same object is an important problem. This is due to the need to distinguish between multiple objects and, between objects and background. Motion-based clustering or segmentation approaches [Vidal and Ma 2004; Black and Anandan 1996; Wang and Adelson 1994] usually assume that the points being tracked lie on rigid bodies in order to simplify the segmentation problem.

5.2. Kernel Tracking

Kernel tracking is typically performed by computing the motion of the object, which is represented by a primitive object region, from one frame to the next. The object motion is generally in the form of parametric motion (translation, conformal, affine, etc.) or the dense flow field computed in subsequent frames. These algorithms differ in terms of the appearance representation used, the number of objects tracked, and the method used to estimate the object motion. We divide these tracking methods into two subcategories based on the appearance representation used, namely, templates and density-based appearance models, and multiview appearance models.

5.2.1. Tracking Using Template and Density-Based Appearance Models. Templates and density-based appearance models (see Section 2) have been widely used because of the their relative simplicity and low computational cost. We divide the trackers in this category into two subcategories based on whether the objects are tracked individually or jointly.

5.2.1.1. Tracking single objects. The most common approach in this category is template *matching*. Template matching is a brute force method of searching the image, I_w, for a region similar to the object template, O_t defined in the previous frame. The position of the template in the current image is computed by a similarity measure, for example, cross correlation: $\arg\max_{dx,dy} \frac{\Sigma_x \Sigma_y (O_t(x,y) \times I_w(x+dx, y+dy))}{\sqrt{\Sigma_x \Sigma_y O_t^2(x,y)}}$, where $(d\,x, d\,y)$ specify the candidate template position. Usually image intensity or color features are used to form the

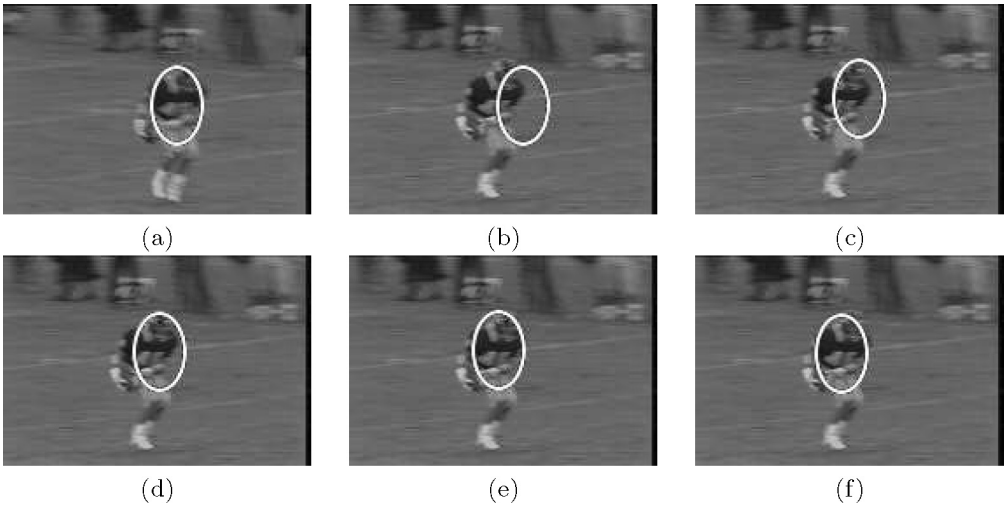


Fig. 12. Mean-shift tracking iterations. (a) estimated object location at time $t-1$, (b) frame at time t with initial location estimate using the previous object position, (c), (d), (e) location update using mean-shift iterations, (f) final object position at time t.

templates. Since image intensity is very sensitive to illumination changes, image gradients [Birchfield 1998] can also be used as features. A limitation of template matching is its high computation cost due to the brute force search. To reduce the computational cost, researchers usually limit the object search to the vicinity of its previous position. Also, more efficient algorithms for template matching have been proposed [Schweitzer et al. 2002].

Note that instead of templates, other object representations can be used for tracking, for instance, color histograms or mixture models can be computed by using the appearance of pixels inside the rectangular or ellipsoidal regions. Fieguth and Terzopoulos [1997] generate object models by finding the mean color of the pixels inside the rectangular object region. To reduce computational complexity, they search the object in eight neighboring locations. The similarity between the object model, M, and the hypothesized position, H, is computed by evaluating the ratio between the color means computed from M and H. The position which provides the highest ratio is selected as the current object location.

Comaniciu and Meer [2003] use a weighted histogram computed from a circular region to represent the object. Instead of performing a brute force search for locating the object, they use the mean-shift procedure (Section 4.3). The mean-shift tracker maximizes the appearance similarity iteratively by comparing the histograms of the object, Q, and the window around the hypothesized object location, P. Histogram similarity is defined in terms of the Bhattacharya coefficient, $\sum_{u=1}^{b} P(u)Q(u)$, where b is the number of bins. At each iteration, the mean-shift vector is computed such that the histogram similarity is increased. This process is repeated until convergence is achieved, which usually takes five to six iterations. For histogram generation, the authors use a weighting scheme defined by a spatial kernel which gives higher weights to the pixels closer to the object center. Comaniciu [2002] extended the mean-shift tracking approach used a joint spatial-color histogram (Section 4.3) instead of just a color histogram. An example of mean-shift tracking is given in Figure 12. An obvious advantage of the mean-shift tracker over the standard template matching is the elimination of a brute force search, and the computation of the translation of the object patch in a small number of

iterations. However, mean-shift tracking requires that a portion of the object is inside the circular region upon initialization (part of the object has to be inside the white ellipse in Figure 12(b)). Implementation of the mean-shift tracker is available in OpenCV as CAMSHIFT at MeanShiftSegmentSrc.

Jepson et al. [2003] propose an object tracker that tracks an object as a three-component mixture, consisting of the stable appearance features, transient features and noise process. The stable component identifies the most reliable appearance for motion estimation, that is, the regions of the object whose appearance does not quickly change over time. The transient component identifies the quickly changing pixels. The noise component handles the outliers in the object appearance that arise due to noise. An online version of the EM algorithm is used to learn the parameters of this three-component mixture. The authors use the phase of the steerable filter responses as features for appearance representation. The object shape is represented by an ellipse. The motion of the object is computed in terms of warping the tracked region from one frame to the next one. The warping transformation consists of translation, (t_x, t_y), rotation (a, b), and scale, s, parameters:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = s \begin{pmatrix} a & b \\ -b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}. \tag{10}$$

A weighted combination of the stable and transient components is used to determine the warping parameters. The advantage of learning stable and transient features is that one can give more weight to stable features for tracking, for example, if the face of a person who is talking is being tracked, then the forehead or nose region can give a better match to the face in the next frame as opposed to the mouth of the person (see Figure 13).

Another approach to track a region defined by a primitive shape is to compute its translation by use of an optical flow method. Optical flow methods are used for generating dense flow fields by computing the flow vector of each pixel under the brightness constancy constraint, $I(x, y, t) - I(x + dx, y + dy, t + dt) = 0$ [Horn and Schunk 1981]. This computation is always carried out in the neighborhood of the pixel either algebraically [Lucas and Kanade. 1981] or geometrically [Schunk 1986]. Extending optical flow methods to compute the translation of a rectangular region is trivial. In 1994, Shi and Tomasi proposed the KLT tracker which iteratively computes the translation (du, dv) of a region (e.g., 25×25 patch) centered on an interest point (for interest point detection, see Section 4.1):

$$\begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix} = \begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}.$$

This equation is similar in construction to the optical flow method proposed by Lucas and Kanade [1981]. Once the new location of the interest point is obtained, the KLT tracker evaluates the quality of the tracked patch by computing the affine transformation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}, \tag{11}$$

between the corresponding patches in consecutive frames. If the sum of square difference between the current patch and the projected patch is small, they continue tracking the feature, otherwise the feature is eliminated. The implementation of the KLT tracker is available at KLTSrc . The results obtained by the KLT tracker are shown in Figure 14.

Fig. 13. Results of the robust online tracking method by Jepson et al. [2003] (a) The target region in different frames. (b) The mixing probability of the stable component. Note that the probabilities around the mouth and eyebrow regions change, while they remain the same in the other regions (©2003 IEEE).

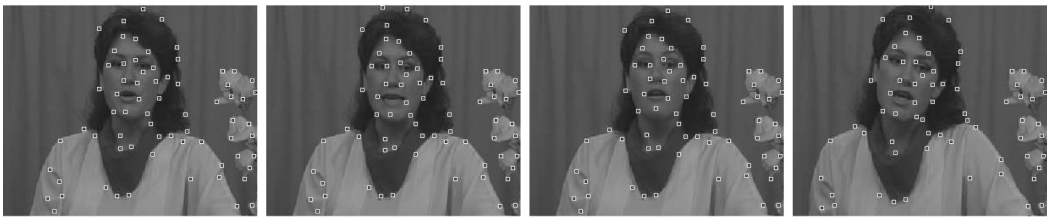


Fig. 14. Tracking features using the KLT tracker.

5.2.1.2. Tracking Multiple Objects. Modeling objects individually does not take into account the interaction between multiple objects and between objects and background during the course of tracking. An example interaction between objects can be one object partially or completely occluding the other. The tracking methods given in the following model the complete image, that is, the background and all moving objects are explicitly tracked.

Tao et al. [2002] propose an object tracking method based on modeling the whole image, I^t, as a set of layers. This representation includes a single background layer and one layer for each object. Each layer consists of shape priors (ellipse), Φ, motion model (translation and rotation), Θ, and layer appearance, A, (intensity modeled using a single Gaussian). Layering is performed by first compensating the background motion modeled by projective motion such that the object's motion can be estimated from the compensated image using 2D parametric motion. Then, each pixel's probability of belonging to a layer (object), p_l, is computed based on the object's previous motion and shape characteristics. Any pixel far from a layer is assigned a uniform background probability, p_b. Later, the object's appearance (intensity, color) probability p_a is coupled with p_l to obtain the final layer estimate. The model parameters (Φ_t, Θ_t, A_t) that maximize observing a layer at time t are estimated iteratively using an expectation maximization algorithm. However, due to the difficulty in simultaneous estimation of the parameters, the authors individually estimate one set, while fixing the others. For instance, they first estimate layer ownership using the intensity for each pixel, then they estimate the motion (rotation and translation) using appearance probabilities, and finally update layer ownership using this motion. The unknowns for each object are iteratively estimated until the layer ownership probabilities are maximized.

Isard and MacCormick [2001] propose joint modeling of the background and foreground regions for tracking. The background appearance is represented by a mixture of Gaussians. Appearance of all foreground objects is also modeled by mixture of Gaussians. The shapes of objects are modeled as cylinders. They assume the ground plane is known, thus the 3D object positions can be computed. Tracking is achieved by using particle filters where the state vector includes the 3D position, shape and the velocity of all objects in the scene. They propose a modified prediction and correction scheme for particle filtering which can increase or decrease the size of the state vector to include or remove objects. The method can also tolerate occlusion between objects. However, the maximum number of objects in the scene is required to be predefined. Another limitation of the approach is the use of the same appearance model for all foreground objects, and it requires training to model the foreground regions.

5.2.2. Tracking Using Multiview Appearance Models. In the previous tracking methods, the appearance models, that is, histograms, templates etc., are usually generated online. Thus these models represent the information gathered about the object from the most recent observations. The objects may appear different from different views, and if the object view changes dramatically during tracking, the appearance model may no longer be valid, and the object track might be lost. To overcome this problem, different views of the object can be learned offline and used for tracking.

In 1998, Black and Jepson proposed a subspace-based approach, that is, eigenspace, to compute the affine transformation from the current image of the object to the image reconstructed using eigenvectors. First, a subspace representation of the appearance of an object is built using Principal Component Analysis (PCA), then the transformation from the image to the eigenspace is computed by minimizing the so-called subspace constancy equation which evaluates the difference between the image reconstructed using the eigenvectors and the input image. Minimization is performed in two steps: finding subspace coefficients and computing affine parameters. In the first step, the

affine parameters are fixed, and the subspace coefficients are computed. In the second step, using the new subspace coefficients, affine parameters are computed. Based on this, tracking is performed by estimating the affine parameters iteratively until the difference between the input image and the projected image is minimized. Note that the use of eigenspace for similarity computation is a useful alternative to standard template matching techniques such as SSD and normalized correlation. The eigenspace-based similarity computation is equivalent to matching with a linear combination of eigen templates. This allows for distortions in the templates, for example, distortion caused by illumination changes in images.

In a similar vein, Avidan [2001] used a Support Vector Machine (SVM) classifier for tracking. SVM is a general classification scheme that, given a set of positive and negative training examples, finds the best separating hyperplane between the two classes [1998]. During testing, the SVM gives a score to the test data indicating the degree of membership of the test data to the positive class. For SVM-based trackers, the positive examples consist of the images of the object to be tracked, and the negative examples consist of all things that are not to be tracked. Generally, negative examples consist of background regions that could be confused with the object. Avidan's tracking method, instead of minimizing the intensity difference of a template from the image regions, maximizes the SVM classification score over image regions in order to estimate the position of the object. One advantage of this approach is that knowledge about background objects (negative examples that are not to be tracked) is explicitly incorporated in the tracker.

5.2.3. Discussion. The main goal of the trackers in this category is to estimate the object motion. With the region-based object representation, computed motion implicitly defines the object region as well as the object orientation in the next frame since, for each point of the object in the current frame, its location in the next frame can be determined using the estimated motion model. Depending on the context in which these trackers are being used, only one of these three properties might be more important. For instance, in the case of analyzing the object behavior based on the object trajectory, only the motion is adequate. However, to identify an object, the region it encompasses is also important. In order to evaluate the performance of the trackers in this category, one can define measures based on what is expected from the tracker. In the case when the tracker is expected to provide only object motion, the evaluation can be performed by computing a distance measure between the estimated and actual motion parameters. An example of a distance measure can be the angular distance, $d = \frac{\mathbf{A}.\mathbf{B}}{|\mathbf{A}||\mathbf{B}|}$, between the motion vectors, $\mathbf{A}$ and $\mathbf{B}$. For applications when the tracker is required to provide the correct object region in addition to its trajectory, the tracker's performance can be evaluated by computing the precision and the recall measures. Both the precision and the recall measure are defined in terms of the intersection of the hypothesized and correct object region. In particular, precision is the ratio of the intersection to the hypothesized regions. The recall is the ratio of the intersection to the ground truth.

A qualitative comparison of kernel trackers can be obtained based on

—tracking single or multiple objects,
—ability to handle occlusion,
—requirement of training,
—type of motion model, and
—requirement of a manual initialization.

In Table IV, we provide the qualitative comparison of the methods discussed in this section.

Table IV. Qualitative Comparison of Geometric Model-Based Trackers (Init. denotes initialization. **#**: number of objects, **M**: multiple objects, **S**: single object respectively, **A**: affine or homography, **T**: translational motion, **S**: scaling, **R**: rotation, **P**: partial occlusion, **F**: full occlusion. Symbols $\surd$ and $\times$ respectively denote if the tracker requires or does not require training or initialization.)

	#	Motion	Training	Occ.	Init.
Simple template matching	S	T	$\times$	P	$\surd$
Mean-shift [Comaniciu et al. 2003]	S	T + S	$\times$	P	$\surd$
KLT [Shi and Tomasi 1994]	S	A	$\times$	P	$\surd$
Appearance Tracking [Jepson et al. 2003]	S	T + S + R	$\times$	P	$\surd$
Layering [Tao et al. 2002]	M	T + S + R	$\times$	F	$\times$
Bramble [Isard and MacCormick 2001]	M	T + S + R	$\surd$	F	$\times$
EigenTracker [Black and Jepson 1998]	S	A	$\surd$	P	$\surd$
SVM [Avidan 2001]	S	T	$\surd$	P	$\surd$

Use of primitive geometric shapes to represent objects is very common due to the real-time applicability of the state-of-the-art methods. Because of the rigidity constraint, tracking methods in this category compute parametric motion of the object. This motion is usually in the form of translation, conformal affine, affine, or projective. Motion of the object can be estimated by maximizing the object appearance similarity between the previous and current frame. The estimation process can be in the form of a brute force search, or by using gradient ascent (descent)-based maximization (minimization) process. Object trackers, based on the gradient ascent (descent) approach, require that some part of the object is at least visible inside the chosen shape whose location is defined by the previous object position. To eliminate such requirements, a possible approach is to use Kalman filtering or particle filtering discussed in the context of point trackers to predict the location of the object in the next frame. Given the object state defined in terms of velocity and acceleration of the object centroid these filters will estimate the position of the object centroid such that the likelihood of observing part of the object inside the kernel is increased [Comaniciu et al. 2003]. This requirement can also be met by performing global motion compensation, assuming that the objects are farther from the camera and camera motion can be estimated by affine or projective transformation [Yilmaz et al. 2003].

One of the limitations of primitive geometric shapes for object representation is that parts of the objects may be left outside of the defined shape while parts of the background may reside inside it. This phenomena can be observed for both the rigid objects (when the object pose changes) and nonrigid objects (when local motion results in changes in object appearance). In such cases, the object motion estimated by maximizing model similarity may not be correct. To overcome this limitation, one approach is to force the kernel to reside inside the object rather than encapsulating the complete shape. Another approach is to model the object appearance by probability density functions of color/texture and assign weights to the pixels residing inside the primitive shape based on the conditional probability of observed color/texture.

5.3. Silhouette Tracking

Objects may have complex shapes, for example, hands, head, and shoulders (see Figure 15(a)) that cannot be well described by simple geometric shapes. Silhouette-based methods provide an accurate shape description for these objects. The goal of a silhouette-based object tracker is to find the object region in each frame by means of an object model generated using the previous frames. This model can be in the form of a color histogram, object edges or the object contour. We divide silhouette trackers into two categories, namely, shape matching and contour tracking. Shape matching approaches search for the object silhouette in the current frame. Contour tracking approaches, on the other hand, evolve an initial contour to its new position in the current

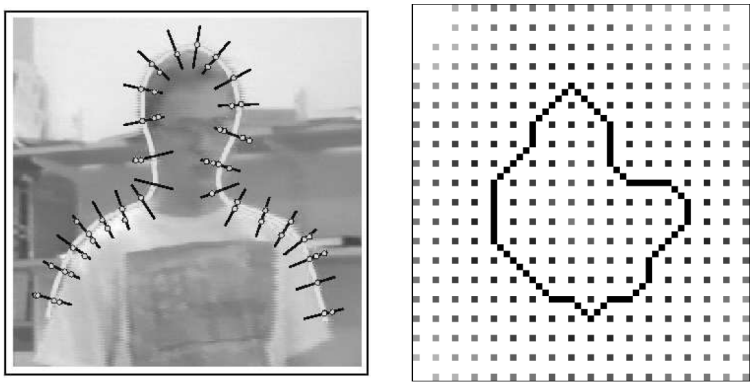


Fig. 15. (a) Edge observations along the contour normals (©1998 Kluwer). (b) Level set contour representation, each grid position encodes the Euclidean distance between a grid point and the point on the contour; gray levels represent the values of the grid.

frame by either using the state space models or direct minimization of some energy functional.

5.3.1. Shape Matching. Shape matching can be performed similar to tracking based on template matching (Section 5.2), where an object silhouette and its associated model is searched in the current frame. The search is performed by computing the similarity of the object with the model generated from the hypothesized object silhouette based on previous frame. In this approach, the silhouette is assumed to only translate from the current frame to the next, therefore nonrigid object motion is not explicitly handled. The object model, which is usually in the form of an edge map, is reinitialized to handle appearance changes in every frame after the object is located. This update is required to overcome tracking problems related to viewpoint and lighting condition changes as well as nonrigid object motion. In 1993, Huttenlocher et al. performed shape matching using an edge-based representation. The authors use the Hausdorff distance to construct a correlation surface from which the minimum is selected as the new object position. The Hausdorff metric is a mathematical measure for comparing two sets of points $A = \{a_1, a_2, \cdots, a_n\}$ and $B = \{b_1, b_2, \cdots, b_m\}$ in terms of the least similar members [Hausdorff 1962]:

$$H(A, B) = \max\{h(A, B), h(b, A)\}, \tag{12}$$

where $h(A, B) = \sup_{a \in A} \inf_{b \in B} \| a - b \|$ and $\| . \|$ is the norm of choice. In the context of matching using an edge-based model, Hausdorff distance measures the most mismatched edges. Due to this, the method emphasize parts of the edge map that are not drastically affected by object motion. For instance, in the case of a walking person, the head and the torso do not change their shape much, whereas the motion of the arms and legs will result in drastic shape change, such that removing the edges corresponding to arms and legs will improve the tracking performance. In a similar vein, Li et al. [2001] propose using the Hausdorff distance for verification of the trajectories and pose estimation problem. Tracking is achieved by evaluating the optical flow vector computed inside the hypothesized silhouette such that the average flow provides the new object position. For verification, an edge-based model is kept for each tracked object for all possible object poses. The authors then apply distance transform to the edges in the

hypothesized object edges to speed up the computation of L_2 norm in the Hausdorff distance computation.

Another approach to match shapes is to find corresponding silhouettes detected in two consecutive frames. Establishing silhouette correspondence, or in short silhouette matching, can be considered similar to point matching discussed in Section 5.1. However, the main difference between silhouette matching and point matching is the object representations and the object models used. In particular, silhouette matching uses the complete object region in, contrast to using points. In addition, silhouette matching makes use of an objects appearance features, whereas point matching uses only motion and position-based features. Silhouette detection is usually carried out by background subtraction (see Section 4.2 for discussion). Once the object silhouettes are extracted, matching is performed by computing some distance between the object models associated with each silhouette. Object models are usually in the form of density functions (color or edge histograms), silhouette boundary (closed or open object contour), object edges or a combination of these models. In 2004, Kang et al. used histograms of color and edges as the object models. In contrast to traditional histograms, they proposed generating histograms from concentric circles with various radii centered on a set of control points on a reference circle. The reference circle is chosen as the smallest circle encapsulating the object silhouette. Use of concentric circles implicitly encodes the spatial information which in regular histogram is only possible when the spatial (x, y) coordinates are included in the observation vector [Comaniciu and Meer 2002]. Resulting color and edge histograms are rotation, translation, and scale invariant, hence, provide the same matching score for objects transformed by conformal affine transform. The matching score can be computed using several distance measures including cross-correlation, Bhattacharya distance, and Kullback-Leibler divergence. Among these three measures, the authors conclude that the Bhattacharya distance and the Kullback-Leibler divergence perform similarly, and both peform better than the correlation-based measure. To match silhouettes in consecutive frames, Haritaoglu et al. [2000] model the object appearance by the edge information obtained inside the object silhouette. In particular, the edge model is used to refine the translation of the object using the constant velocity assumption. This refinement is carried out by performing binary correlation between the object edge in the consecutive frames.

In contrast to looking for possible silhouette matches in consecutive frames, tracking silhouettes can be performed by computing the flow vectors for each pixel inside the silhouette such that the flow that is dominant over the entire silhouette is used to generate the silhouette trajectory. Following this observation, Sato and Aggarwal [2004] proposed to generating object tracks by applying Hough transform in the velocity space to the object silhouettes in consecutive frames. Binary object silhouettes are detected using background subtraction (see Section 4.2). Then, from a spatio-temporal window around each moving region pixel, a velocity Hough transform is applied to compute voting matrices for the vertical flow v and the horizontal flow u. These voting matrices provides the so-called Temporal Spatio-Velocity (TSV) image in 4D (x, y, u, v) per frame. TSV image encodes the dominant motion of a moving region pixel and its likelihood in terms of number of votes such that a thresholding operation will provide regions with similar motion patterns. In contrast to appearance-based matching of silhouettes, TSV provides a motion-based matching of the object silhouettes and is less sensitive to appearance variations, due to different object views (e.g., front and back of the object may look different).

5.3.2. Contour Tracking. Contour tracking methods, in contrast to shape matching methods. iteratively evolve an initial contour in the previous frame to its new position in the current frame. This contour evolution requires that some part of the object

in the current frame overlap with the object region in the previous frame. Tracking by evolving a contour can be performed using two different approaches. The first approach uses state space models to model the contour shape and motion. The second approach directly evolves the contour by minimizing the contour energy using direct minimization techniques such as gradient descent.

5.3.2.1. Tracking Using State Space Models. The object's state is defined in terms of the shape and the motion parameters of the contour. The state is updated at each time instant such that the contour's a posteriori probability is maximized. The posterior probability depends on the prior state and the current likelihood which is usually defined in terms of the distance of the contour from observed edges.

Terzopoulos and Szeliski [1992] define the object state by the dynamics of the control points. The dynamics of the control points are modeled in terms of a *spring model*, which moves the control points based on the spring stiffness parameters. The new state (spring parameters) of the contour is predicted using the Kalman filter. The correction step uses the image observations which are defined in terms of the image gradients. In 1998, Isard and Blake defined the object state in terms of spline shape parameters and affine motion parameters. The measurements consist of image edges computed in the normal direction to the contour (see Figure 15(a)). The state is updated using a particle filter. In order to obtain initial samples for the filter, they compute the state variables from the contours extracted in consecutive frames during a training phase. During the testing phase, the current state variables are estimated through particle filtering based on the edge observations along normal lines at the control points on the contour.

In 2000, MacCormick and Blake extended the particle filter-based object tracker in Isard and Blake [1998] to track multiple objects by including the *exclusion principle* for handling occlusion. The exclusion principle integrates into the sampling step of the particle filtering framework such that, for two objects, if a feature lies in the observation space of both objects, then it contributes more to the samples of the object which is occluding the other object. Since the exclusion principle is only defined between two objects, this approach can track at most two objects undergoing occlusion at any time instant.

Chen et al. [2001] propose a contour tracker where the contour is parameterized as an ellipse. Each contour node has an associated HMM and the states of each HMM is defined by the points lying on the lines normal to the contour control point. The observation likelihood of the contour depends on the background and the foreground partitions defined by the edge along the normal line on contour control points. The state transition probabilities of the HMM are estimated using the JPDAF. Given the observation likelihood and the state transition probabilities, the current contour state is estimated using the Viterbi algorithm [1967]. After the contour is approximated, an ellipse is fit to enforce elliptical shape constraint.

The methods just discussed above represent the contours using explicit representation, for example, parametric spline. Explicit representations do not allow topology changes such as region split or merge [Sethian 1999]. Next, we will discuss contour tracking methods based on direct minimization of energy functional. These methods can use implicit representations and allow topology changes.

5.3.2.2. Tracking by Direct Minimization of Contour Energy Functional. In the context of contour evolution, there is an analogy between the segmentation methods discussed in Section 4.3 and the contour tracking methods in this category. Both the segmentation and tracking methods minimize the energy functional either by greedy methods or by gradient descent. The contour energy is defined in terms of temporal information in the form of either the temporal gradient (optical flow) [Bertalmio et al. 2000; Mansouri

Fig. 16. Car tracking using the level sets method (©2002. IEEE)

2002; Cremers and Schnorr 2003], or appearance statistics generated from the object and the background regions [Yilmaz et al. 2004; Ronfard 1994].

Contour tracking using temporal image gradients is motivated by the extensive work on computing the optical flow. The optical flow constraint is derived from the brightness constancy constraint: $I^{t+1}(x, y) - I^t(x - u, y - v) = 0$, where I is the image, t is the time, and (u, v) is the flow vector in the x and the y directions. Bertalmio et al. [2000] use this constraint to evolve the contour in consecutive frames. Their objective was to compute u and v iteratively for each contour position using the level set representation (see Figure 15(b)). At each iteration, contour speed in the normal direction, $\overrightarrow{n}$, is computed by projecting the gradient magnitude $|\nabla I_t|$ on $\overrightarrow{n}$. The authors use two energy functionals, one for contour tracking, E_t, and another one for intensity morphing, E_m: $E_m(\Gamma) = \int_0^1 E_{im}(\mathbf{v})ds$ and $E_t(\Gamma) = \int_0^1 E_{ext}(\mathbf{v})ds$, where E_{ext} is computed based on E_m. The intensity morphing functional, which minimizes intensity changes in the current and the previous frames, $\nabla I_t = I_t - I_{t-1}$, on the hypothesized object contour, $\frac{\partial F(x,y)}{\partial t} = \nabla I_t(x, y) \parallel \nabla F(x, y) \parallel$, is coupled with the contour tracking equation, $\frac{\partial \phi(x,y)}{\partial t} = \nabla I_t(x, y)\vec{n_F}\vec{n_\phi} \parallel \nabla \phi(x, y) \parallel$, and both functionals are minimized simultaneously. For instance, if $\nabla I_t(x, y) \gg 0$, then the contour moves with the maximum speed in its normal direction, and $I^{t-1}(x, y)$ is morphed into $I^t(x, y)$. On the other hand, if $\nabla I_t(x, y) \approx 0$, then the evolution speed will be zero.

Similarly, Mansouri [2002] also uses the optical flow constraint for contour tracking. In contrast to Bertalmio et al. [2000] which computes the flow only on the object boundary, his approach is motivated by computing the flow vector for each pixel inside the complete object region in a circular neighborhood with radius r using a brute force search. Once the flow vectors are computed, the contour energy, which is based on the brightness constancy constraint, is evaluated. This process is iteratively performed until the energy is minimized. In Figure 16, we show the results of the tracking method proposed by Mansouri [2002] in a car sequence.

In 2003, Cremers and Schnorr also used the optical flow for contour evolution, and constraint such that an object can only have homogeneous flow vectors inside the region. Their energy is a modified form of the common Mumford-Shah energy [Mumford and Shah 1989], which evolves the contour until a region with homogeneous flow vectors is achieved. They also incorporated the shape priors to better estimate the object shape. The shape priors are generated from a set of object contours such that each control point on the contour has an associated Gaussian with a mean and standard deviation of the spatial positions of the corresponding control points on all the contours.

An alternative to using the optical flow is to exploit the consistency of the statistics computed inside and outside the object region from one frame to the next. This approach requires initialization of the contour in the current frame with its previous position. In this context, Ronfrad [1994] defines the energy functional governing the contour evolution based on the piecewise stationary image models formulated as Ward distances. Ward distance can be considered as a measure of image contrast [Beaulieu

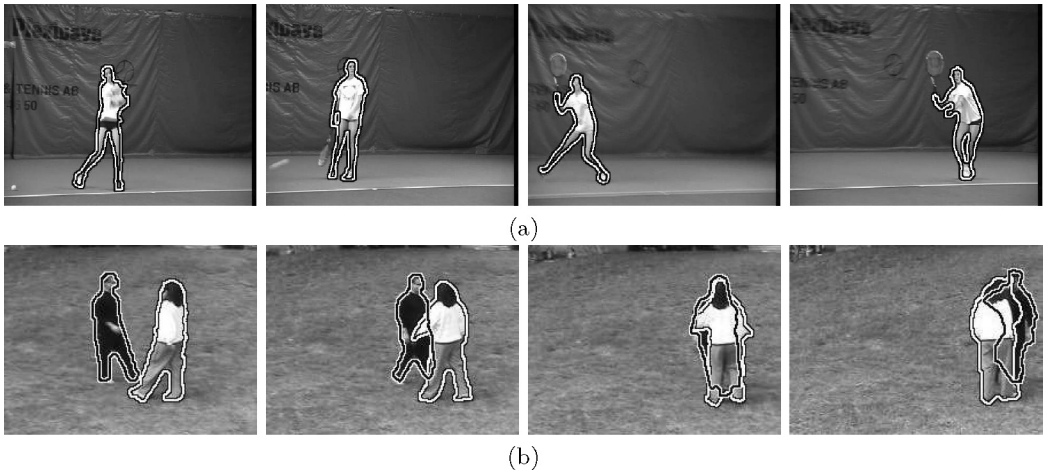


Fig. 17. Contour tracking results. (a) tracking of a tennis player, (b) tracking in presence of occlusion, using the method proposed by Yilmaz and Shah [2004] (©2004 IEEE).

and Goldberg 1989]. However, Ward distance can not be analitycally defined, hence, Ronfard's approach individually evolves each contour point based on its local neighborhood. In a similar vein, Yilmaz and Shah [2004] evolve an object contour using the color and texture models generated in a band around the object's boundary (see Figure 17(a)). The width of the band serves as a means to combine region and boundary-based contour tracking methods into a single framework. In contrast to the aforementioned methods, Yilmaz et al. [2004] model the object shape and its changes by means of a level set-based shape model. In this model, the grid points of the level set hold the means and the standard deviations of the distances of points from the object boundary. The level set-based shape model resolves the object occlusions during the course of tracking (see Figure 17(b)).

5.3.3. Discussion. Silhouette tracking is employed when tracking of the complete region of an object is required. In the context of region tracking, the precision and recall measures are defined in terms of the intersection of the hypothesized and correct object regions. The precision is the ratio of the intersection to the hypothesized region and recall is the ratio of the intersection to the ground truth. Important factors to distinguish different silhouette trackers are: What features are used? How is occlusion handled? Is training required or not? Moreover some algorithms only use information about the silhouette boundary for tracking, while others use the complete region inside the silhouette. Generally the region-based approaches are more resilient to noise. A qualitative comparison of contour-based silhouette tracking approaches is given in Table V.

The most important advantage of tracking silhouettes is their flexibility to handle a large variety of object shapes. Silhouettes can be represented in different ways. The most common silhouette representation is in the form of a binary indicator function, which marks the object region by ones and the nonobject regions by zeros. For contour-based methods, the silhouette is represented either explicitly or implicitly (see Figure 15). Explicit representation defines the boundary of the silhouette by a set of control points. Implicit representation defines the silhouette by means of a function defined on a grid. The most common implicit contour representation is the level sets representation.

Table V. Qualitative Comparison of Silhouette Trackers (Occ. denotes occlusion handling and Trn. denotes training. **#**: number of objects, **S**: single, **M**: multiple, **P**: partial, **F**: full. Symbols $\surd$ and $\times$ denote whether the tracker can or cannot handle occlusions, and requires or does not require training.)

	#	Occ.	Trn.	Features	Technique
Shape Matching					
[Huttenlocher et al. 1993]	S	$\times$	$\times$	Edge template	Template matching
[Li et al. 2001]	S	$\times$	$\surd$	Edge template	Template matching
[Kang et al. 2004]	S	$\times$	$\times$	Color histogram	Histogram matching
[Sato and Aggarwal 2004]	S	$\surd$	$\times$	Silhouette	Hough transform
Contour Evolution using State Space Models					
[Terzopoulos and Szeliski 1992]	S	$\times$	$\surd$	Gradient mag.	Kalman filtering
[Isard and Blake 1998]	S	$\times$	$\surd$	Gradient mag.	Particle filtering
[MacCormick and Blake 2000]	M	F	$\surd$	Gradient mag.	Particle filtering
[Chen et al. 2001]	S	$\times$	$\surd$	Gradient mag.	JPDAF
Contour Evolution by Direct Minimization					
[Bertalmio et al. 2000]	S	$\times$	$\times$	Temporal gradient	Gradient descent
[Mansouri 2002]	S	$\times$	$\times$	Temporal gradient	Gradient descent
[Paragios and Deriche 2002]	S	$\times$	$\times$	Temporal gradient	Gradient descent
[Cremers et al. 2002]	S	P	$\surd$	Region statistics	Gradient descent
[Yilmaz et al. 2004]	M	F	$\times$	Region statistics	Gradient descent

The representations chosen by the silhouette-based object trackers can be in the form of motion models (similar to point trackers), appearance models (similar to kernel trackers), or shape models or a combination of these. Object appearance is usually modeled by parametric or nonparametric density functions such as mixture of Gaussians or histograms. Object shape can be modeled in the form of contour subspace where a subspace is generated from a set of possible object contours obtained from different object poses [Blake and Isard 2000]. Additionally, object shape can be implicitly modeled via a level set function where the grid positions are assigned at the distance generated from different level set functions corresponding to different object poses [Yilmaz et al. 2004]. Appearance-based shape representations are also commonly used by researchers who employ a brute force silhouette search. For edge-based shape representation, Hausdorff distance is the most widely used measure. However, Hausdorff measure is known for its sensitivity to noise. Hence, instead of using the maximum of distances, researchers have considered using an average of the distances [Baddeley 1992].

Occlusion handling is another important aspect of silhouette tracking methods. Usually methods do not address the occlusion problem explicitly. A common approach is to assume constant motion or constant acceleration where, during occlusion, the object silhouette from the previous frame is translated to its hypothetical new position. Few methods explicitly handle object occlusions by enforcing shape constraints [MacCormick and Blake 2000; Yilmaz et al. 2004].

Another important issue related to silhouette trackers is their capability for dealing with object split and merge. For instance, while tracking a silhouette of a person carrying an object, when the person leaves an object, a part of the person's contour will be placed on the left object (region split). These topology changes of region split or merge can be handled well by implicit contour representations.

6. RELATED ISSUES

In this section, we discuss issues that arise in tracking objects in realistic scenarios. These include locating objects as they undergo occlusion and keeping unique tracks of objects as they are viewed through multiple cameras.

6.1. Resolving Occlusion

Occlusion can be classified into three categories: self occlusion, interobject occlusion, and occlusion by the background scene structure. Self occlusion occurs when one part of the object occludes another. This situation most frequently arises while tracking articulated objects. Interobject occlusion occurs when two objects being tracked occlude each other. Similarly, occlusion by the background occurs when a structure in the background occludes the tracked objects. Generally, for interobject occlusion, the multiobject trackers like MacCormick and Blake [2000] and Elgammal et al. [2002] can exploit the knowledge of the position and the appearance of the occluder and occludee to detect and resolve occlusion. Partial occlusion of an object by a scene structure is hard to detect since it is difficult to differentiate between the object changing its shape and the object getting occluded.

A common approach to handle complete occlusion during tracking is to model the object motion by linear dynamic models or by nonlinear dynamics and, in the case of occlusion, to keep on predicting the object location until the object reappears. For example, a linear velocity model is used in Beymer and Konolige [1999] and a Kalman filter is used for estimating the location and motion of objects. A nonlinear dynamic model is used in Isard and MacCormick [2001] and a particle filter employed for state estimation.

Researchers have also utilized other features to resolve occlusion, for example, silhouette projections [Haritaoglu et al. 2000] (to locate persons' heads during partial occlusion), and optical flow [Dockstader and Tekalp 2001b] (assuming that two objects move in opposite directions). Occlusion can also be implicitly resolved during generation of object tracks. In 2004, Sato and Aggarwal fit a slant cylindrical to the TSV (see Section 5.2 for more details) in the spatio-temporal space to recover from occlusions. In particular, two continuous slant cylinders, which are fit to the tracked object silhouette before the occlusion and after the occlusion, will intersect with each other for the frames whre the occlusion occurred. This intersection provides continuous trajectory before, during, and after the occlusion. Free-form object contour trackers employ a different occlusion resolution approach. These methods usually address occlusion by using shape priors which are either built ahead of time [Cremers et al. 2002] or built online [Yilmaz et al. 2004]. In particular, Cremers et al. [2002] build a shape model from subspace analysis (PCA) of possible object shapes to fill in missing contour parts. Yilmaz et al. [2004] build online shape priors using a mixture model based on the level set contour representation. Their approach is able to handle complete object occlusion.

The chance of occlusion can be reduced by an appropriate selection of camera positions. For instance, if the cameras are mounted on airborne vehicles, that is, when a birds-eye view of the scene is available, occlusions between objects on the ground do not occur. However, oblique view cameras are likely to encounter multiple object occlusions and require occlusion handling mechanisms. Multiple cameras viewing the same scene can also be used to resolve object occlusions during tracking [Dockstader and Tekalp 2001a; Mittal and Davis 2003]. These methods are discussed in the next section.

6.2. Multiple Camera Tracking

The need for using multiple cameras for tracking arises for two reasons. The first reason is the use of depth information for tracking and occlusion resolution. The second reason for using multiple cameras is to increase the area under view since it is not possible for a single camera to observe large areas because of a finite sensor field-of-view. An important issue in using multiple cameras is the relationship between the different camera views which can be manually defined [Collins et al. 2001; Cai and Aggarwal 1999] or computed automatically [Lee et al. 2000; Khan and Shah 2003] from the

observations of the objects moving in the scene. For the tracking algorithms that require the depth estimation, high computational cost is another concern. However, due to the availability of successful commercial products, off-the-shelf real-time depth recovery systems are available which can be employed. In addition, methods like Mittal and Davis [2003] do not perform dense depth estimation but compute a sparse depth map (a single depth estimate for each object using a region-based stereo method) which also reduces the computational load. Multi-camera tracking methods like Dockstader and Tekalp [2001a] and Mittal and Davis [2003] have demonstrated superior tracking results as compared to single camera trackers in the case of persistent occlusion between the objects.

The aforementioned multi-camera tracking methods assume stationary cameras. Recently, Kang et al. [2003] used a combination of stationary and pan-tilt-zoom cameras with overlapping views for tracking. In many situations, it is not possible to have overlapping camera views due to limited resources or large areas of interest. Methods for tracking in such a scenario inherently have to deal with sparse object observations due to nonoverlapping views. Therefore some assumptions have to be made about the object speed and the path in order to obtain the correspondences across cameras [Huang and Russell 1997; Kettnaker and Zabih 1999; Javed et al. 2003]. Note that these methods, which establish object correspondence across nonoverlapping cameras, assume 1) the cameras are stationary and 2) the object tracks within each camera are available. The performance of these algorithms depends greatly on how much the objects follow the established paths and expected time intervals across cameras. For scenarios in which spatio-temporal constraints cannot be used, for example, objects moving arbitrarily in the nonoverlap region, the only tracking-by-recognition approach can be employed, which uses the appearance and the shape of the object to recognize it when it reappears in a camera view.

7. FUTURE DIRECTIONS

Significant progress has been made in object tracking during the last few years. Several robust trackers have been developed which can track objects in real time in simple scenarios. However, it is clear from the papers reviewed in this survey that the assumptions used to make the tracking problem tractable, for example, smoothness of motion, minimal amount of occlusion, illumination constancy, high contrast with respect to background, etc., are violated in many realistic scenarios and therefore limit a tracker's usefulness in applications like automated surveillance, human computer interaction, video retrieval, traffic monitoring, and vehicle navigation. Thus, tracking and associated problems of feature selection, object representation, dynamic shape, and motion estimation are very active areas of research and new solutions are continuously being proposed.

One challenge in tracking is to develop algorithms for tracking objects in unconstrained videos, for example, videos obtained from broadcast news networks or home videos. These videos are noisy, compressed, unstructured, and typically contain edited clips acquired by moving cameras from multiple views. Another related video domain is of formal and informal meetings. These videos usually contain multiple people in a small field of view. Thus, there is severe occlusion, and people are only partially visible. One interesting solution in this context is to employ audio in addition to video for object tracking. There are some methods being developed for estimating the point of location of audio source, for example, a person's mouth, based on four or six microphones. This audio-based localization of the speaker provides additional information which then can be used in conjunction with a video-based tracker to solve problems like severe occlusion.

In general, an important issue that has been neglected in the development of tracking algorithms is integration of contextual information. For example, in a vehicle tracking application, the location of vehicles should be constrained to paths on the ground as opposed to vertical walls or the sky. Recent work in the area of object recognition [Torralba 2003; Kumar and Hebert 2003] has shown that exploiting contextual information is helpful in recognition. In addition, advances in classifiers [Friedman et al. 2000; Tipping 2001] have made accurate detection of scene context possible, for example, man made structures, paths of movement, class of objects, etc. A tracker that takes advantage of contextual information to incorporate general constraints on the shape and motion of objects will usually perform better than one that does not exploit this information. This is because a tracker designed to give the best average performance in a variety of scenarios can be less accurate for a particular scene than a tracker that is attuned (by exploiting context) to the characteristics of that scene.

The use of a particular feature set for tracking can also greatly affect the performance. Generally, the features that best discriminate between multiple objects and, between the object and background are also best for tracking the object. Many tracking algorithms use a weighted combination of multiple features assuming that a combination of preselected features will be discriminative. A wide range of feature selection algorithms have been investigated in the machine learning and pattern recognition communities. However, these algorithms require offline training information about the target and/or the background. Such information is not always available. Moreover, as the object appearance or background varies, the discriminative features also vary. Thus, there is a need for online selection of discriminative features. Some work has been done in this area for online selection of individual features [Collins and Liu 2003; Stern and Efros 2002]. However, the problem of efficient online estimation of discriminative feature sets remains unresolved. One promising direction to achieve this goal is the use of the online boosting methods [Oza 2002] for feature selection.

In a similar vein, most tracking algorithms use prespecified models for object representation. The capability to learn object models online will greatly increase the applicability of a tracker. Motion-based segmentation [Vidal and Ma 2004; Black and Anandan 1996; Wang and Adelson 1994] and multibody factorization [Costeira and Kanade 1998; Gear 1998] methods have been used to learn models for multiple objects moving in a scene. However, these approaches assume rigid body motion. Unsupervised learning of object models for multiple nonrigid moving objects from a single camera remains an unsolved problem. One interesting direction that has largely been unexplored is the use of semisupervised learning techniques for modeling objects. These techniques (cotraining [Levin et al. 2003; Blum and Mitchell 1998], transductive SVMs [Joachims 1999], constrained graph cuts [Yu and Shi 2004]) do not require prohibitive amounts of training data. Moreover, they can not only learn nonrigid shapes and/or appearance, but they can also encode the knowledge of the background in the form of negative training data.

Probabilistic state-space methods including Kalman Filters [Bar-Shalom and Foreman 1988], JPDAFs [Cox 1993], HMMs [Rabiner 1989], and Dynamic Bayesian Networks (DBNs) [Jensen 2001] have been extensively used to estimate object motion parameters. Among these methods, DBNs are probably the most general method for representation of conditional dependencies between multiple variables and/or image observations. They also provide a principled framework for fusing information from different sources. However, there is a need for more efficient solutions for inference before DBNs are more commonly used in tracking applications.

Overall, we believe that additional sources of information, in particular prior and contextual information, should be exploited whenever possible to attune the tracker to the particular scenario in which it is used. A principled approach to integrate these

disparate sources of information will result in a general tracker that can be employed with success in a variety of applications.

8. CONCLUDING REMARKS

In this article, we present an extensive survey of object tracking methods and also give a brief review of related topics. We divide the tracking methods into three categories based on the use of object representations, namely, methods establishing point correspondence, methods using primitive geometric models, and methods using contour evolution. Note that all these classes require object detection at some point. For instance, the point trackers require detection in every frame, whereas geometric region or contours-based trackers require detection only when the object first appears in the scene. Recognizing the importance of object detection for tracking systems, we include a short discussion on popular object detection methods. We provide detailed summaries of object trackers, including discussion on the object representations, motion models, and the parameter estimation schemes employed by the tracking algorithms. Moreover, we describe the context of use, degree of applicability, evaluation criteria, and qualitative comparisons of the tracking algorithms. We believe that, this article, the first survey on object tracking with a rich bibliography content, can give valuable insight into this important research topic and encourage new research.

REFERENCES

AGGARWAL, J. K. AND CAI, Q. 1999. Human motion analysis: A review. *Comput. Vision Image Understand. 73,* 3, 428–440.

ALI, A. AND AGGARWAL, J. 2001. Segmentation and recognition of continuous human activity. In *IEEE Workshop on Detection and Recognition of Events in Video*. 28–35.

AVIDAN, S. 2001. Support vector tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 184–191.

BADDELEY, A. 1992. Errors in binary images and an l version of the haus- dorff metric. *Nieuw Archief voor Wiskunde 10*, 157–183.

BALLARD, D. AND BROWN, C. 1982. *Computer Vision*. Prentice-Hall.

BAR-SHALOM, Y. AND FOREMAN, T. 1988. *Tracking and Data Association*. Academic Press Inc.

BARRON, J., FLEET, D., AND BEAUCHEMIN, S. 1994. Performance of optical flow techniques. *Int. J. Comput. Vision 12*, 43–77.

BEAULIEU, J. AND GOLDBERG, M. 1989. Hierarchy in picture image segmentation: A step wise optimization approach. *IEEE Trans. Patt. Analy. Mach. Intell. 11*, 150–163.

BERTALMIO, M., SAPIRO, G., AND RANDALL, G. 2000. Morphing active contours. *IEEE Trans. Patt. Analy. Mach. Intell. 22*, 7, 733–737.

BEYMER, D. AND KONOLIGE, K. 1999. Real-time tracking of multiple people using continuous detection. In *IEEE International Conference on Computer Vision (ICCV) Frame-Rate Workshop.*.

BIRCHFIELD, S. 1998. Elliptical head tracking using intensity gradients and color histograms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 232–237.

BLACK, M. AND ANANDAN, P. 1996. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Comput. Vision Image Understand. 63*, 1, 75–104.

BLACK, M. AND JEPSON, A. 1998. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Int. J. Comput. Vision 26*, 1, 63–84.

BLAKE, A. AND ISARD, M. 2000. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer.

BLUM, A. AND MITCHELL, T. 1998. Combining labeled and unlabeled data with co-training. In *11th Annual Conference on Computational Learning Theory*. 92–100.

BLUM, A. L. AND LANGLEY, P. 1997. Selection of relevant features and examples in machine learning. *Artific. Intell. 97*, 1-2, 245–271.

BOSER, B., GUYON, I. M., AND VAPNIK, V. 1992. A training algorithm for optimal margin classifiers. In *ACM Workshop on Conference on Computational Learning Theory (COLT)*. 142–152.

BOWYER, K., KRANENBURG, C., AND DOUGHERTY, S. 2001. Edge detector evaluation using empirical roc curve. *Comput. Vision Image Understand. 10*, 77–103.

BREGLER, C., HERTZMANN, A., AND BIERMANN, H. 2000. Recovering nonrigid 3d shape from image streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 690–696.

BROIDA, T. AND CHELLAPPA, R. 1986. Estimation of object motion parameters from noisy images. *IEEE Trans. Patt. Analy. Mach. Intell. 8*, 1, 90–99.

CAI, Q. AND AGGARWAL, J. 1999. Tracking human motion in structured environments using a distributed camera system. *IEEE Trans. Patt. Analy. Mach. Intell. 2*, 11, 1241–1247.

CANNY, J. 1986. A computational approach to edge detection. *IEEE Trans. Patt. Analy. Mach. Intell. 8*, 6, 679–698.

CASELLES, V., KIMMEL, R., AND SAPIRO, G. 1995. Geodesic active contours. In *IEEE International Conference on Computer Vision (ICCV)*. 694–699.

CHAM, T. AND REHG, J. M. 1999. A multiple hypothesis approach to figure tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition*. 239–245.

CHANG, Y. L. AND AGGARWAL, J. K. 1991. 3d structure reconstruction from an ego motion sequence using statistical estimation and detection theory. In *Workshop on Visual Motion*. 268–273.

CHEN, Y., RUI, Y., AND HUANG, T. 2001. Jpdaf based hmm for real-time contour tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 543–550.

COLLINS, R., LIPTON, A., FUJIYOSHI, H., AND KANADE, T. 2001. Algorithms for cooperative multisensor surveillance. *Proceedings of IEEE 89*, 10, 1456–1477.

COMANICIU, D. 2002. Bayesian kernel tracking. In *Annual Conference of the German Society for Pattern Recognition*. 438–445.

COMANICIU, D. AND MEER, P. 1999. Mean shift analysis and applications. In *IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 1197–1203.

COMANICIU, D. AND MEER, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Patt. Analy. Mach. Intell. 24*, 5, 603–619.

COMANICIU, D., RAMESH, V., AND MEER, P. 2003. Kernel-based object tracking. *IEEE Trans. Patt. Analy. Mach. Intell. 25*, 564–575.

COOTES, T., EDWARDS, G., AND TAYLOR, C. 2001. Robust real-time periodic motion detection, analysis, and applications. *IEEE Trans. Patt. Analy. Mach. Intell. 23*, 6, 681–685.

COSTEIRA, J. AND KANADE, T. 1998. A multibody factorization method for motion analysis. *Int. J. Comput. Vision 29*, 3, 159–180.

COX, I. AND HINGORANI, S. 1996. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Trans. Patt. Analy. Mach. Intell. 18*, 2, 138–150.

COX, I. J. 1993. A review of statistical data association techniques for motion correspondence. *Int. J. Comput. Vision 10*, 1, 53–66.

CREMERS, D., KOHLBERGER, T., AND SCHNORR, C. 2002. Non-linear shape statistics in mumford-shah based segmentation. In *European Conference on Computer Vision (ECCV)*.

CREMERS, D. AND SCHNORR, C. 2003. Statistical shape knowledge in variational motion segmentation. *I. Srael Nent. Cap. J. 21*, 77–86.

DOCKSTADER, S. AND TEKALP, A. M. 2001a. Multiple camera tracking of interacting and occluded human motion. *Proceedings of the IEEE 89*, 1441–1455.

DOCKSTADER, S. AND TEKALP, M. 2001b. On the tracking of articulated and occluded video object motion. *Real Time Image 7*, 5, 415–432.

EDWARDS, G., TAYLOR, C., AND COOTES, T. 1998. Interpreting face images using active appearance models. In *International Conference on Face and Gesture Recognition*. 300–305.

ELGAMMAL, A., DURAISWAMI, R., HARWOOD, D., AND DAVIS, L. 2002. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of IEEE 90*, 7, 1151–1163.

ELGAMMAL, A., HARWOOD, D., AND DAVIS, L. 2000. Non-parametric model for background subtraction. In *European Conference on Computer Vision (ECCV)*. 751–767.

FIEGUTH, P. AND TERZOPOULOS, D. 1997. Color-based tracking of heads and other mobile objects at video frame rates. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 21–27.

FREUND, Y. AND SCHAPIRE, R. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. *Computat. Learn. Theory*. 23–37.

FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. 2000. Additive logistic regression: A statistical view of boosting. annals of statistics. *Ann. Stat. 38*, 2, 337–374.

GAO, X., BOULT, T., COETZEE, F., AND RAMESH, V. 2000. Error analysis of background adaption. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 503–510.

GAVRILA, D. M. 1999. The visual analysis of human movement: A survey. *Comput. Vision Image Understand. 73*, 1, 82–98.

GEAR, C. W. 1998. Multibody grouping from motion images. *Int. J. Comput. Vision 29*, 2, 133–150.

GREENSPAN, H., BELONGIE, S., GOODMAN, R., PERONA, P., RAKSHIT, S., AND ANDERSON, C. 1994. Overcomplete steerable pyramid filters and rotation invariance. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 222–228.

GREWE, L. AND KAK, A. 1995. Interactive learning of a multi-attribute hash table classifier for fast object recognition. *Comput. Vision Image Understand. 61*, 3, 387–416.

HARALICK, R., SHANMUGAM, B., AND DINSTEIN, I. 1973. Textural features for image classification. *IEEE Trans. Syst. Man Cybern. 33*, 3, 610–622.

HARITAOGLU, I., HARWOOD, D., AND DAVIS, L. 2000. W4: real-time surveillance of people and their activities. *IEEE Trans. Patt. Analy. Mach. Intell. 22*, 8, 809–830.

HARRIS, C. AND STEPHENS, M. 1988. A combined corner and edge detector. In *4th Alvey Vision Conference*. 147–151.

HARRISSRC. Harris Source Code. http://www.cs.uwa.edu.au/~pk/Research/MatlabFns/Spatial/harris.m.

HAUSDORFF, F. 1962. *Set Theory*. Chelsea, New York, NY.

HORN, B. AND SCHUNK, B. 1981. Determining optical flow. *Artific. Intell. 17*, 185–203.

HUANG, T. AND RUSSELL, S. 1997. Object identification in a bayesian context. In *Proceedings of International Joint Conference on Artificial Intelligence*. 1276–1283.

HUE, C., CADRE, J. L., AND PREZ, P. 2002. Sequential monte carlo methods for multiple targettracking and data fusion. *IEEE Trans. Sign. Process. 50*, 2, 309–325.

HUTTENLOCHER, D., NOH, J., AND RUCKLIDGE, W. 1993. Tracking nonrigid objects in complex scenes. In *IEEE International Conference on Computer Vision (ICCV)*. 93–101.

INTILLE, S., DAVIS, J., AND BOBICK, A. 1997. Real-time closed-world tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 697–703.

IRANI, M. AND ANANDAN, P. 1998. Video indexing based on mosaic representations. *IEEE Trans. Patt. Analy. Mach. Intell. 20*, 6, 577–589.

ISARD, M. AND BLAKE, A. 1998. Condensation - conditional density propagation for visual tracking. *Int. J. Comput. Vision 29*, 1, 5–28.

ISARD, M. AND MACCORMICK, J. 2001. Bramble: A bayesian multiple-blob tracker. In *IEEE International Conference on Computer Vision (ICCV)*. 34–41.

JAIN, R. AND NAGEL, H. 1979. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Trans. Patt. Analy. Mach. Intell. 1*, 2, 206–214.

JAVED, O., RASHEED, Z., SHAFIQUE, K., AND SHAH, M. 2003. Tracking across multiple cameras with disjoint views. In *IEEE International Conference on Computer Vision (ICCV)*. 952–957.

JENSEN, F. V. 2001. *Bayesian Networks and Decision Graphs*. Springer.

JEPSON, A., FLEET, D., AND ELMARAGHI, T. 2003. Robust online appearance models for visual tracking. *IEEE Trans. Patt. Analy. Mach. Intell. 25*, 10, 1296–1311.

JOACHIMS, T. 1999. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning*. 200–209.

KALMANSRC. Kalman Filtering Source Code. http://www.ai.mit.edu/~murphyk/Software/index.html.

KANADE, T., COLLINS, R., LIPTON, A., BURT, P., AND WIXSON, L. 1998. Advances in cooperative multi-sensor video surveillance. *Darpa IU Workshop*. 3–24.

KANG, J., COHEN, I., AND MEDIONI, G. 2003. Continuous tracking within and across camera streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 267–272.

KANG, J., COHEN, I., AND MEDIONI, G. 2004. Object reacquisition using geometric invariant appearance model. In *International Conference on Pattern Recongnition (ICPR)*. 759–762.

KASS, M., WITKIN, A., AND TERZOPOULOS, D. 1988. Snakes: active contour models. *Int. J. Comput. Vision 1*, 321–332.

KETTNAKER, V. AND ZABIH, R. 1999. Bayesian multi-camera surveillance. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 117–123.

KHAN, S. AND SHAH, M. 2003. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Trans. Patt. Analy. Mach. Intell. 25*, 10, 1355–1360.

KLTSRC. KLT Source Code. http://www.ces.clemson.edu/~stb/klt/.

KOCKELKORN, M., LUNEBURG, A., AND SCHEFFER, T. 2003. Using transduction and multiview learning to answer emails. In *European Conference on Principle and Practice of Knowledge Discovery in Databases*. 266–277.

KUHN, H. 1955. The hungarian method for solving the assignment problem. *Naval Research Logistics Quart. 2*, 83–97.

KUMAR, S. AND HEBERT, M. 2003. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *IEEE International Conference on Computer Vision (ICCV)*. 1150–1157.

LAWS, K. 1980. Textured image segmentation. PhD thesis, Electrical Engineering, University of Southern California.

LEE, L., ROMANO, R., AND STEIN, G. 2000. Monitoring activities from multiple video streams: Establishing a common coordinate frame. *IEEE Trans. Patt. Recogn. Mach. Intell. 22*, 8 (Aug.), 758–768.

LEVELSETSRC. Level Set Source Code. http://www.cs.utah.edu/~whitaker/vispack/.

LEVIN, A., VIOLA, P., AND FREUND, Y. 2003. Unsupervised improvement of visual detectors using co-training. In *IEEE International Conference on Computer Vision (ICCV)*. 626–633.

LI, B., CHELLAPPA, R., ZHENG, Q., AND DER, S. 2001. Model-based temporal object verification using video. *IEEE Trans. Image Process. 10*, 6, 897–908.

LIYUAN, L. AND MAYLOR, L. 2002. Integrating intensity and texture differences for robust change detection. *IEEE Trans. Image Process. 11*, 2, 105–112.

LOWE, D. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2, 91–110.

LUCAS, B. D. AND KANADE., T. 1981. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*.

MACCORMICK, J. AND BLAKE, A. 2000. Probabilistic exclusion and partitioned sampling for multiple object tracking. *Int. J. Comput. Vision 39*, 1, 57–71.

MACKAY, D. J. C. 1998. Introduction to Monte Carlo methods. In *Learning in Graphical Models*, M. I. Jordan, Ed. NATO Science Series. Kluwer Academic Press, 175–204.

MALLAT, S. 1989. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Patt. Analy. Mach. Intell. 11*, 7, 674–693.

MANSOURI, A. 2002. Region tracking via level set pdes without motion computation. *IEEE Trans. Patt. Analy. Mach. Intell. 24*, 7, 947–961.

MATTHIES, L., SZELISKI, R., AND KANADE, T. 1989. Kalman filter-based algorithms for estimating depth from image sequences. *Int. J. Comput. Vision 3*, 3, 209–238.

MEANSHIFTSEGMENTSRC. Mean-Shift Segmentation Source Code. http://www.caip.rutgers.edu/riul/research/code.html.

MEANSHIFTTRACKSRC. Mean-Shift Tracking Source Code. http://www.intel.com/technology/computing/opencv/index.htm.

MIKOLAJCZYK, K. AND SCHMID, C. 2002. An affine invariant interest point detector. In *European Conference on Computer Vision (ECCV)*. Vol. 1. 128–142.

MIKOLAJCZYK, K. AND SCHMID, C. 2003. A performance evaluation of local descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1615–1630.

MITTAL, A. AND DAVIS, L. 2003. M2 tracker: A multiview approach to segmenting and tracking people in a cluttered scene. *Int. J. Comput. Vision 51*, 3, 189–203.

MOESLUND, T. AND GRANUM, E. 2001. A survey of computer vision-based human motion capture. *Comput. Vision Image Understand. 81*, 3, 231–268.

MONNET, A., MITTAL, A., PARAGIOS, N., AND RAMESH, V. 2003. Background modeling and subtraction of dynamic scenes. In *IEEE International Conference on Computer Vision (ICCV)*. 1305–1312.

MORAVEC, H. 1979. Visual mapping by a robot rover. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 598–600.

MUGHADAM, B. AND PENTLAND, A. 1997. Probabilistic visual learning for object representation. *IEEE Trans. Patt. Analy. Mach. Intell. 19*, 7, 696–710.

MUMFORD, D. AND SHAH, J. 1989. Optimal approximations by piecewise smooth functions and variational problems. *Comm. Pure Appl. Mathemat. 42*, 5, 677–685.

MURTY, K. 1968. An algorithm for ranking all the assignments in order of increasing cost. *Operations Resear. 16*, 682–686.

OLIVER, N., ROSARIO, B., AND PENTLAND, A. 2000. A bayesian computer vision system for modeling human interactions. *IEEE Trans. Patt. Analy. Mach. Intell. 22*, 8, 831–843.

OZA, N. C. 2002. Online ensemble learning. PhD Thesis, University of California, Berkeley.

PAPAGEORGIOU, C., OREN, M., AND POGGIO, T. 1998. A general framework for object detection. In *IEEE International Conference on Computer Vision (ICCV)*. 555–562.

PARAGIOS, N. AND DERICHE, R. 2000. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Trans. Patt. Analy. Mach. Intell. 22*, 3, 266–280.

PARAGIOS, N. AND DERICHE, R. 2002. Geodesic active regions and level set methods for supervised texture segmentation. *Int. J. Comput. Vision 46*, 3, 223–247.

PARK, S. AND AGGARWAL, J. K. 2004. A hierarchical bayesian network for event recognition of human actions and interactions. *Multimed. Syst. 10*, 2, 164–179.

PARTICLEFLTSRC. Particle Filtering Source Code. http://www-sigproc.eng.cam.ac.uk/smc/software.html.

PASCHOS, G. 2001. Perceptually uniform color spaces for color texture analysis: an empirical evaluation. *IEEE Trans. Image Process. 10*, 932–937.

RABINER, L. R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2, 257–286.

RANGARAJAN, K. AND SHAH, M. 1991. Establishing motion correspondence. *Conference Vision Graphies Image Process 54*, 1, 56–73.

RASMUSSEN, C. AND HAGER, G. 2001. Probabilistic data association methods for tracking complex visual objects. *IEEE Trans. Patt. Analy. Mach. Intell. 23*, 6, 560–576.

COLLINS, R. AND LIU, Y.. 2003. On-line selection of discriminative tracking features. In *IEEE International Conference on Computer Vision (ICCV)*. 346–352.

REID, D. B. 1979. An algorithm for tracking multiple targets. *IEEE Trans. Autom. Control 24*, 6, 843–854.

RITTSCHER, J., KATO, J., JOGA, S., AND BLAKE, A. 2000. A probabilistic background model for tracking. In *European Conference on Computer Vision (ECCV)*. Vol. 2. 336–350.

RONFARD, R. 1994. Region based strategies for active contour models. *Int. J. Comput. Vision 13*, 2, 229–251.

ROSALES, R. AND SCLAROFF, S. 1999. 3d trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 117–123.

ROWE, S. AND BLAKE, A. 1996. Statistical mosaics for tracking. *Israel Verj. Cap. J. 14*, 549–564.

ROWLEY, H., BALUJA, S., AND KANADE, T. 1998. Neural network-based face detection. *IEEE Trans. Patt. Analy. Mach. Intell. 20*, 1, 23–38.

SALARI, V. AND SETHI, I. K. 1990. Feature point correspondence in the presence of occlusion. *IEEE Trans. Patt. Analy. Mach. Intell. 12*, 1, 87–91.

SATO, K. AND AGGARWAL, J. 2004. Temporal spatio-velocity transform and its application to tracking and interaction. *Comput. Vision Image Understand. 96*, 2, 100–128.

SCHUNK, B. 1986. The image flow constraint equation. *Comput. Visison Graphics Image Process. 35*, 20–46.

SCHWEITZER, H., BELL, J. W., AND WU, F. 2002. Very fast template matching. In *European Conference on Computer Vision (ECCV)*. 358–372.

SERBY, D., KOLLER-MEIER, S., AND GOOL, L. V. 2004. Probabilistic object tracking using multiple features. In *IEEE International Conference of Pattern Recognition (ICPR)*. 184–187.

SETHI, I. AND JAIN, R. 1987. Finding trajectories of feature points in a monocular image sequence. *IEEE Trans. Patt. Analy. Mach. Intell. 9*, 1, 56–73.

SETHIAN, J. 1999. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics Computer Vision and Material Sciences*. Cambridge University Press.

SHAFIQUE, K. AND SHAH, M. 2003. A non-iterative greedy algorithm for multi-frame point correspondence. In *IEEE International Conference on Computer Vision (ICCV)*. 110–115.

SHI, J. AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Trans. Patt. Analy. Mach. Intell. 22*, 8, 888–905.

SHI, J. AND TOMASI, C. 1994. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 593–600.

SIFTSRC. SIFT Source Code. http://www.cs.ucla.edu/∼vedaldi/code/siftpp/assets/siftpp/versions/.

SONG, K. Y., KITTLER, J., AND PETROU, M. 1996. Defect detection in random color textures. *Israel Verj. Cap. J. 14*, 9, 667–683.

STAUFFER, C. AND GRIMSON, W. 2000. Learning patterns of activity using real time tracking. *IEEE Trans. Patt. Analy. Mach. Intell. 22*, 8, 747–767.

STENGER, B., RAMESH, V., PARAGIOS, N., COETZEE, F., AND BUHMANN, J. 2001. Topology free hidden markov models: Application to background modeling. In *IEEE International Conference on Computer Vision (ICCV)*. 294–301.

STERN, H. AND EFROS, B. 2002. Adaptive color space switching for face tracking in multi-colored lighting environments. In *IEEE International Conference on Automatic Face and Gesture Recognition*. 0249.

STREIT, R. L. AND LUGINBUHL, T. E. 1994. Maximum likelihood method for probabilistic multi-hypothesis tracking. In *Proceedings of the International Society for Optical Engineering (SPIE.)* vol. 2235. 394–405.

SZELISKI, R. AND COUGHLAN, J. 1997. Spline-based image registration. *Int. J. Comput. Vision 16*, 1-3, 185–203.

TANIZAKI, H. 1987. Non-gaussian state-space modeling of nonstationary time series. *J. Amer. Statist. Assoc. 82*, 1032–1063.

TAO, H., SAWHNEY, H., AND KUMAR, R. 2002. Object tracking with bayesian estimation of dynamic layer representations. *IEEE Trans. Patt. Analy. Mach. Intell. 24*, 1, 75–89.

TERZOPOULOS, D. AND SZELISKI, R. 1992. Tracking with kalman snakes. In *Active Vision*, A. Blake and A. Yuille, Eds. MIT Press.

TIEU, K. AND VIOLA, P. 2004. Boosting image retrival. *Int. J. Comput. Vision 56*, 1, 17–36.

TIPPING, M. E. 2001. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Resear. 1*, 1, 211–244.

TORRALBA, A. 2003. Contextual priming for object detection. *Int. J. Comput. Vision 53*, 2, 169–191.

TORRESANI, L. AND BREGLER, C. 2002. Space-time tracking. In *European Conference on Computer Vision (ECCV)*. 801–812.

TOYAMA, K., J. KRUMM, B. B., AND MEYERS, B. 1999. Wallflower: Principles and practices of background maintenance. In *IEEE International Conference on Computer Vision (ICCV)*. 255–261.

VAPNIK, V. 1998. *Statistical Learning Theory*. John Wiley NY.

VASWANI, N., ROYCHOWDHURY, A., AND CHELLAPPA, R. 2003. Activity recognition using the dynamics of the configuration ofinteracting objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 633–640.

VEENMAN, C., REINDERS, M., AND BACKER, E. 2001. Resolving motion correspondence for densely moving points. *IEEE Trans. Patt. Analy. Mach. Intell. 23*, 1, 54–72.

VIDAL, R. AND MA, Y. 2004. A unified algebraic approach to 2-d and 3-d motion segmentation. In *European Conference on Computer Vision (ECCV)*. 1–15.

VIOLA, P., JONES, M., AND SNOW, D. 2003. Detecting pedestrians using patterns of motion and appearance. In *IEEE International Conference on Computer Vision (ICCV)*. 734–741.

VITERBI, A. J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory 13*, 260–269.

WANG, J. AND ADELSON, E. 1994. Representing moving images with layers. *IEEE Image Process. 3*, 5, 625–638.

WREN, C., AZARBAYEJANI, A., AND PENTLAND, A. 1997. Pfinder: Real-time tracking of the human body. *IEEE Trans. Patt. Analy. Mach. Intell. 19*, 7, 780–785.

WU, Z. AND LEAHY, R. 1993. An optimal graph theoretic approach to data clustering: Theory and its applications to image segmentation. *IEEE Trans. Patt. Analy. Mach. Intell. 11*, 1101–1113.

XU, N. AND AHUJA, N. 2002. Object contour tracking using graph cuts based active contours. In *IEEE International Conference on Image Processing (ICIP)*. 277–280.

YILMAZ, A., LI, X., AND SHAH, M. 2004. Contour based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Trans. Patt. Analy. Mach. Intell. 26*, 11, 1531–1536.

YILMAZ, A., SHAFIQUE, K., AND SHAH, M. 2003. Target tracking in airborne forward looking imagery. *J. Image Vision Comput. 21*, 7, 623–635.

YU, S. X. AND SHI, J. 2004. Segmentation given partial grouping constraints. *IEEE Trans. Patt. Analy. Mach. Intell. 26*, 2, 173–183.

ZHONG, J. AND SCLAROFF, S. 2003. Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In *IEEE International Conference on Computer Vision (ICCV)*. 44–50.

ZHOU, S., CHELLAPA, R., AND MOGHADAM, B. 2003. Adaptive visual tracking and recognition using particle filters. In *Proceedings IEEE International Conference on Multimedia and Expo (ICME)*. 349–352.

ZHU, S. AND YUILLE, A. 1996. Region competition: unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Trans. Patt. Analy. Mach. Intell. 18*, 9, 884–900.

Received June 2004; revised January 2006; accepted July 2006