Object Detection & Tracking

Fatih Porikli and Alper Yilmaz

Abstract Detecting and tracking objects are among the most prevalent and challenging tasks that a surveillance system has to accomplish in order to determine meaningful events and suspicious activities, and automatically annotate and retrieve video content. Under the business intelligence notion, an object can be a face, a head, a human, a queue of people, a crowd as well as a product on an assembly line. In this chapter we introduce the reader to main trends and provide taxonomy of popular methods to give an insight to underlying ideas as well as to show their limitations in the hopes of facilitating integration of object detection and tracking for more effective business oriented video analytics.

Key words: detection, tracking, representations, descriptors, features

1 Introduction

Visual surveillance in dynamic business environments attempts to detect, track, and recognize objects of interest from multiple videos, and more generally to interpret object behaviors and actions. For instance, it aims to automatically compute the flux of people at public areas such as stores and travel sites, and then attain congestion and demographic analysis to assist in crowd traffic management and targeted advertisement. Such intelligent systems would replace the traditional surveillance setups where the number of cameras exceeds the capacity of costly human operators to monitor them.

Proceeding with a low-level image features to high-level event understanding approach, there are three main steps of visual analytics: detection of objects and agents, tracking of such objects and indicators from frame to frame, and evaluating tracking results to describe and infer semantic events and latent phenomena. This analogy can be extended to other applications including motion-based recognition, access control, video indexing, human-computer interaction, and vehicle traffic monitoring and navigation. This chapter reviews fundamental aspects of the detection and tracking steps to support a deeper appreciation of many applications presented in the rest of the book.

Imagine waiting for your turn in a shopping line at a busy department store. Your visual system can easily sense humans and identify different layers of their interactions. As with other tasks that our brain does effortlessly, visual analytics has turned long out to be entangled for machines. Not surprisingly, this is also an open problem for visual perception.

F. Porikli
Mitsubishi Electric Research Laboratories, USA e-mail: `fatih@merl.com`

A. Yilmaz
The Ohio State University, USA e-mail: `yilmaz.15@osu.edu`

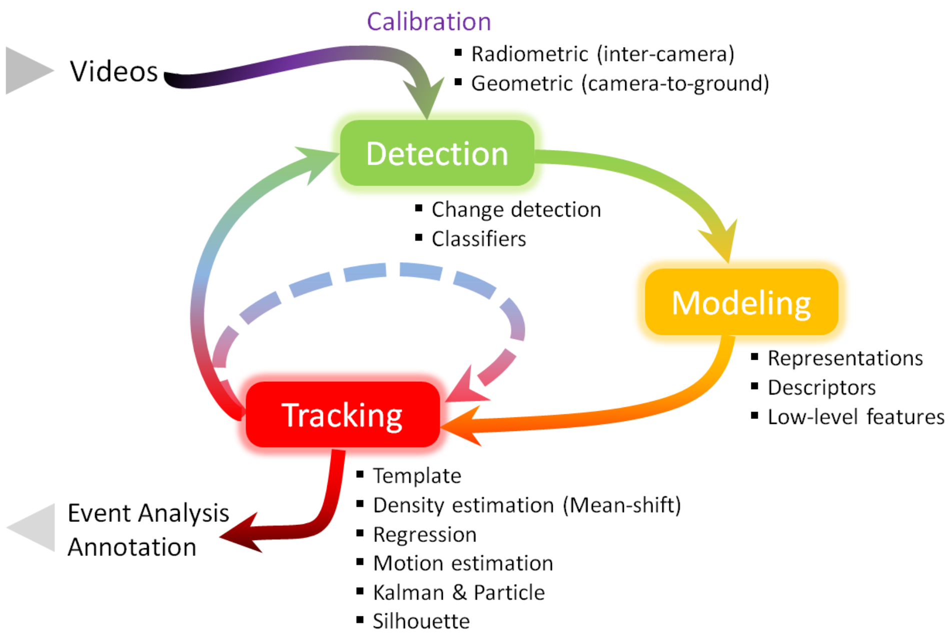

Fig. 1 Fundamental tasks of a video analytics framework based on object detection and tracking.

The main challenge is the problem of variability. A visual detection and tracking system needs to generalize across huge variations in object appearance such due for instance to viewpoint, pose, facial expressions, lighting conditions, imaging quality or occlusions while maintaining specificity to not claim everything it sees are objects of interest. In addition, these tasks should preferably be performed in real-time on conventional computing platforms.

One can ask the question which of the two tasks is easier and which comes first? Within detection, motion changes and appearance cues can be used to distinguish objects, which typically renders it relatively easily, and tracking techniques are often triggered by detection results. Combination of statistical analysis of visual features and temporal motion information usually lead to more robust approaches. For systems that face noisy conditions, however, tracking is suggested to be followed by detection to gather sufficient statistic as several track-before-detect algorithms propose. Besides, tracking steer to choose detection regions, source and sink areas. In any case, it has been common in the past few years, to assume that different strategies are required for these different tasks. Here we take the theoretical view that detection and tracking, rather than being two distinct tasks, represent two points in a spectrum of generalization levels.

Figure **??** illustrates a procedural flow of these reciprocal tasks intertwined with the object modeling. In the following sections, we attempt to give an overview of the popular choices for the object detection, modeling, and tracking stages for which a plethora of solutions have been evidently and inevitably produced over the past several decades.

2 Object Detection

Object detection is essential to initialize tracking process. It is continually applied in every frame. A common approach for moving object detection is to use temporal information extracted from a sequence of images, for instance by computing inter-frame difference, learning a static background scene model and comparing it with the current scene, or finding high motion areas. Another popular

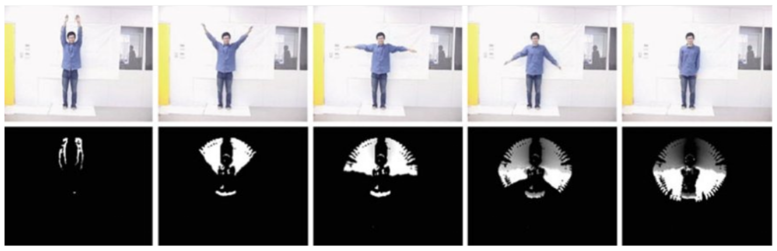

Fig. 2 Motion history images obtained by aggregating frame differences (© A. Ahad, 2010, Springer).

approach to object detection is to slide a window across the image (possibly at multiple scales), and to classify each such local window as containing the target or background. Alternatively, local interest points are extracted from the image, and then each of the regions around these points can be classified, rather than looking at all possible subwindows.

2.1 Change Detection

Change detection is the identification of changes in the state of a pixel through the examination of the appearance values between sets of video frames. Some of the most commonly used changed detection techniques are frame differencing, background subtraction, motion segmentation, and matrix decomposition.

2.1.1 Frame Differencing & Motion History Image

Frame differencing is the intensity dissimilarity between two frames assuming that intensity change of a pixel apparently indicate something changing in the image, e.g. a moving object. It simply subtracts the current image from the previous or a reference image. It has been well studied since the late 70s. In general, the frame difference method is sensitive to illumination changes and it cannot detect an object once it becomes stationary.

Motion History Image [**?**] is obtained by successive layering of frame differences. For each new frame, existing frame differences are decreased in value subject to some threshold and the new silhouette (if any) is overlaid at maximal brightness. Motion history can also be reconstructed by foreground masks. It has the advantage that a range of times from frame-to-frame to several seconds may be encoded in a single image. A sample motion history image is shown in Figure **??**.

2.1.2 Background Subtraction

Detection can be achieved by building a representation of the scene, called the background model, and then observing deviations from this model for each incoming frame. Any gradual change from the background model is assumed to signify a moving object.

Earlier approaches use simple filters to make a prediction of background pixel intensities. In [**?**] Kalman filtering is used to model background dynamics. Similarly, in [**?**] Wiener filtering is used to make a linear predictions at the pixel level. Foreground regions consisting of homogeneous color are filled in at the region level, and in case most of the pixels in a frame exhibit sudden change, the background models are assumed no longer valid at the frame level. At this point either a previously stored pixel based background model is swapped in, or the model is reinitialized.

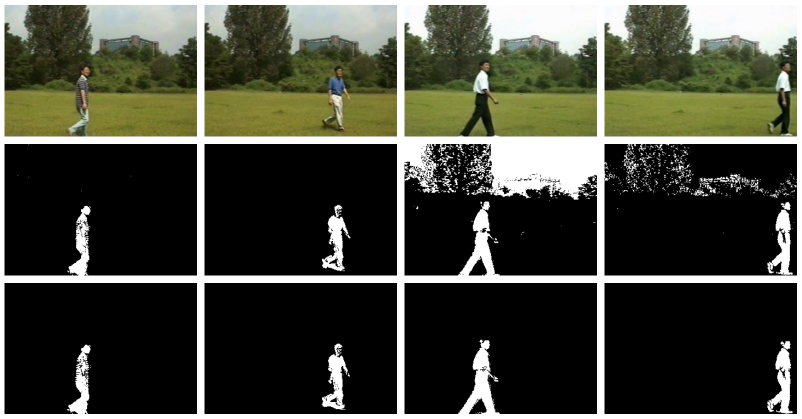

Fig. 3 Sample foreground detection results for GMM (middle row) and Layered model with Bayesian Update (bottom row). When the background changes more than the preset variance of the GMM it produces false alarms.

To learn changes in time, [**?**] proposed to model each pixel with a single Gaussian distribution. Once this background model is derived by updating in several consecutive frames, the likelihood of current pixel color coming from the corresponding model is computed, and the pixels that deviate significantly from their models are labeled as the foreground pixels.

However, a single Gaussian is not a good model for dynamic scenes [**?**] as multiple colors may be observed at a pixel due to repetitive object motion, shadows or reflectance. A substantial improvement is achieved by using multi-modal statistical models to describe background color. For instance, in [**?**] use a Gaussian Mixture Model (GMM) to represent a background pixel. GMM compares a pixel in the current frame with every model in the mixture 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.

As an alternative to mixture models, a Bayesian approach that model each pixel as a combination of layered Gaussians is proposed in [**?**]. Recursive Bayesian estimation is performed to update the background parameters to better preserve the multi-modality of the background model than the conventional expectation maximization fitting, and to automatically determine the number of active layers for each pixel. Moving regions, which are detected using the GMM and layered approaches are shown in Figure **??**.

Instead of relying only a pixel, GMM's can be trained to incorporate extended spatial information. In [**?**] a non-parametric kernel density estimation is used to refit a density function every time to multiple previous pixel values. During the subtraction process the current pixel is matched not only to the corresponding background model but also to the nearby pixel locations. Thus, some robustness against camera jitter or small movements in the background is obtained. Similar affects can be achieved by extending the support to larger blocks and using texture features that are less sensitive to inter-frame illumination variations. Although nonparametric models are robust against small changes they are computationally and memory-wise expensive. Besides, extending the support causes small foreground objects to disappear.

A shortcoming of above background methods is that they neglect the temporal correlation among the previous values of a pixel. This prevents them detecting a structured or near-periodic changes, for example the alternating signals in an intersection, the motion of plants driven by wind, the action of waves on a beach, and the appearance of rotating objects.

A frequency decomposition based background generation that explicitly harnesses the scene dynamics is proposed in [**?**]. To capture the cyclostationary behavior of each pixel, the frequency coefficients of the temporal variation of pixel intensity are computed in temporal windows and a background model that is composed of frequency coefficients is maintained and fused with distance maps to eliminate trail effects.

An alternate approach is to represent the intensity variations of a pixel in an image sequence as discrete states and using Hidden Markov Model (HMM), and switching among these states with the observations to classify pixels [**?**]. The advantage of using HMMs is that certain events, which may not be modeled correctly by unsupervised algorithms, can be learned using the provided training samples.

2.1.3 Motion Segmentation

Motion segmentation refers to the assignment of groups of pixels to various classes based on the speed and direction of their movements. Most approaches to motion segmentation first seek to compute the optical flow of the image sequence. Discontinuities in the optical flow can help in segmenting images into regions that correspond to different objects. Optical flow can arise from relative motion of objects and the camera. Using the rate of change of the image intensity and by assuming that the brightness function changes smoothly, the flow velocity is attained by minimizing a global error function. By assuming a locally constant flow, robustness against noise at the expense of the resolution of their optical field can be improved. From representing moving objects using sets of overlapping layers obtained by k-means clustering to variational methods that jointly solves the problems of motion estimation and segmentation for two consecutive frames in a sequence, many different flow based motion segmentation methods exist.

One way is to estimate the consistency of optical flow over a short duration of time [**?**] where the significant variation of accumulated local optical flows represent the dynamic features of nonstationary background objects. Towards the modeling of the dynamic characteristics, in [**?**] optical flow is computed and utilized as a feature in a higher dimensional space. In order to properly utilize the uncertainties in the features, a kernel based multivariate density estimation technique that adapts the bandwidth according the uncertainties in the test and sample measurements is incorporated.

Optical flow computation will be in error if the constant brightness and velocity smoothness assumptions are violated. In real imagery, their violation is quite common. Typically, the optical flow changes dramatically in highly textured regions, around moving boundaries, at depth discontinuities, etc. Resulting errors propagate across the entire optical flow solution.

2.1.4 Matrix Decomposition

Instead of modeling the variation of individual pixels, the whole image can be vectorized and used in background modeling. In [**?**] a holistic approach using eigenspace decomposition is proposed. For a certain number of input frames, a background matrix is formed by cascading the rows in each frame one after the other, and eigenvalue decomposition is applied to the covariance of the matrix. The background is then represented by the most descriptive eigenvectors, that encompass all possible illuminations to decrease sensitivity 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.

Instead of the conventional background and foreground definition, an intrinsic image inspired method that decomposes a scene into time-varying background and foreground intrinsic images are proposed in [**?**]. The multiplication of these images reconstructs the scene. First, a set of previous images are formed into a temporal scale and their spatial gradients are computed. By taking advantage of the sparseness of the filter outputs, the background is estimated by median filtering of the gradients, the corresponding foreground is found using the background. The intrinsic background/foreground decomposition is robust even under sudden and severe illumination changes, yet computationally expensive.

A learning-based background subtraction approach based on the theory of sparse representation and dictionary learning is proposed in [**?**]. This method makes the following two important assump-

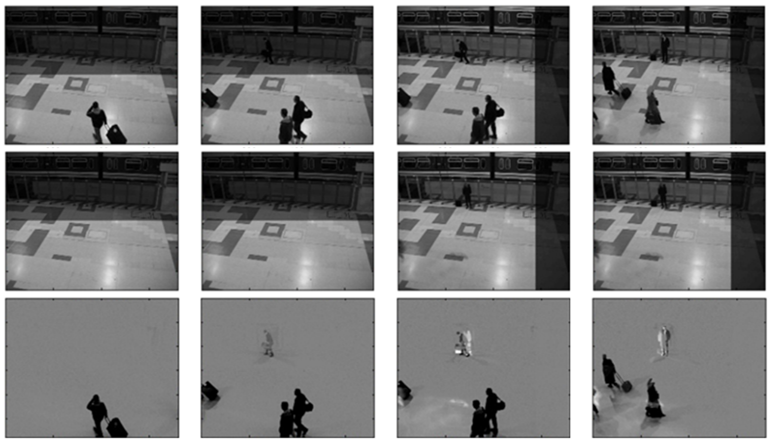

Fig. 4 Background and foreground decompositions are obtained by matrix decomposition.

tions: (1) the background of a scene has a sparse linear representation over a learned dictionary; (2) the foreground is sparse in the sense that majority pixels of the frame belong to the background. These two assumptions enable handling both sudden and gradual background changes. To build a correct background model when training samples are not foreground-free, a robust dictionary learning algorithm is used.

Learning based foreground and background decompositions for a sequence where the background significantly changes (upper and right half of the images are distorted) are shown in Figure **??**.

2.2 Classifiers

Object (e.g. human) detection can be performed by learning a classification function that captures the variation in object appearances and views from a set of labeled training examples in a supervised framework. The input of the classification function is a test region and the output is the estimated label. i.e. object or not. 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. Once the features are selected, different learning approaches can be applied including, but are not limited to, neural networks, boosting, decision trees, and support vector machines. These learning methods compute a hypersurface that separates one object class from the other in a high dimensional space.

Boosting is an iterative method of finding a very accurate classifier by combining many base classifiers, each of which may only be moderately accurate [**?**]. 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. 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 windows.

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 [**?**]. 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. Despite being a linear classifier, SVM can also be used as a non-linear classifier by applying the kernel

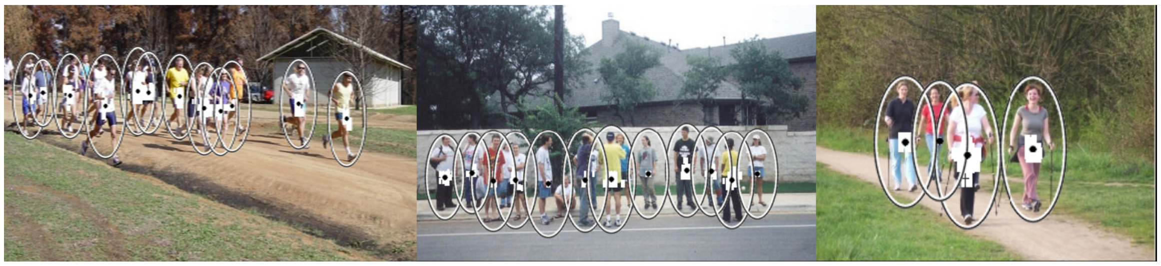

Fig. 5 Detection examples. The classifier is trained on the INRIA data set. White dots show all the detection results. Black dots are the modes generated by mean shift smoothing, and the ellipses are average detection window sizes. There are extremely few false positives and negatives.

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, e.g. Gaussian kernel, and two layer perceptron, e.g. 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.

Leading approaches in classification based object detection can be separated into two groups based on the search technique. The first group of methods is based on sequentially applying a classifier at all the possible subwindows in a given image. In [**?**], a polynomial support vector machine (SVM) was learned using Haar wavelets as object descriptors. Later, the work was extended to multiple classifiers trained to detect object parts, and the responses inside the detection window are combined to give the final decision [**?**]. In [**?**], a human detector was described by training an SVM classifier using densely sampled histogram of oriented gradients inside the detection window. In a similar approach [**?**], near real time detection performances were achieved by training a cascade model using histogram of oriented gradients (HOG) features by taking advantage of the integral histograms [**?**]. Similar to still images, in [**?**], a moving human detection algorithm was described using Haar wavelet descriptors but extracted from space-time differences in video. The operators in the space-time domain are in the form of frame differencing which encode some form of motion information, and frame differencing, when used as an operator in the temporal domain, is assumed to reduce the number of false detections by enforcing object detection in the regions where the motion occurs.

Instead of Haar wavelets or HOG features, region covariance matrices [**?**] are utilized as object descriptors in [**?**]. A region was represented by the covariance matrix of image features, such as spatial location, intensity, higher order derivatives, etc. Similarly, an object is modeled with several covariance matrices of overlapping regions. Since these descriptors do not lie on a vector space, conventional machine learning techniques are not adequate to learn the classifiers. The space of nonsingular covariance matrices can be represented as a connected Riemannian manifold. For classification of points lying on a Riemannian manifold a manifold learning method is presented by incorporating the a priori information about the geometry of the space. Typical human detection results generated by this method is shown in Figure **??**.

The second group of methods is based on detecting object parts [**?**] or common shapes [**?**] and assembling these local features according to geometric constraints to form the final model. In [**?**], parts were represented by co-occurrences of local orientation features and separate detectors were trained for each part using AdaBoost. Object location was determined by maximizing the joint likelihood of part occurrences combined according to the geometric relations. A human detection system for crowded scenes was described in [**?**]. The approach combined local appearance features and their geometric relations with global cues by top-down segmentation based on per pixel likelihoods. Other approaches include using silhouette information either in matching [**?**] or in classification framework [**?**].

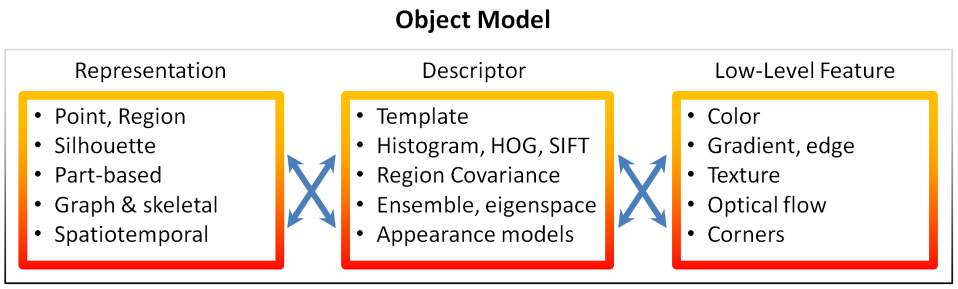

Fig. 6 Different layers of object modeling can be categorized under the representations, descriptors, and low-level features.

Supervised learning methods usually require a large collection of samples from each object class. A possible approach to reduce the amount of manually labeled data is to accompany co-training to supervised learning [?]. The main idea behind co-training 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. Co-training has been used to reduce the amount of manual interaction required for training in the context of AdaBoost [?] and SVM [?].

An alternative classifier based detection of the foreground regions is attempted using corner-based background models [?]. Instead of processing every pixel, this algorithm detects a certain number of feature points using a Harris corner detector and a scale-invariant feature point descriptor. It dynamically learns a single background model and classify each extracted feature as either a background or a foreground feature using a motion tracker to differentiate motion consistent foreground points from background points with random or repetitive motion. Since feature point extraction and descriptor generation are computationally expensive, this algorithm may not be suitable for real-time applications if number of feature points are excessive.

3 Object Modeling

To keep track of location and other properties of the detected objects, one must have an internal representation of an object suitable for matching its descriptor to image features. In this section, we will describe the various object model representations, descriptors, and features commonly employed for tracking (Figure ??). In general, there is a strong relationship between the object representations, descriptions, their low-level features and applications.

3.1 Model Representations

Object representations are usually chosen according to the application domain. 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 case if a geometric shape representation, like ellipse, is used for the object then parametric motion models like affine or projective transformations are appropriate. These representations can approximate the motion of rigid objects in the scene. For a non-rigid object, silhouette or contour is the most descriptive representation and both parametric and non-parametric models can be used to specify their motion. We explain some of these representations (as illustrated in Figure ??) in more detail below.

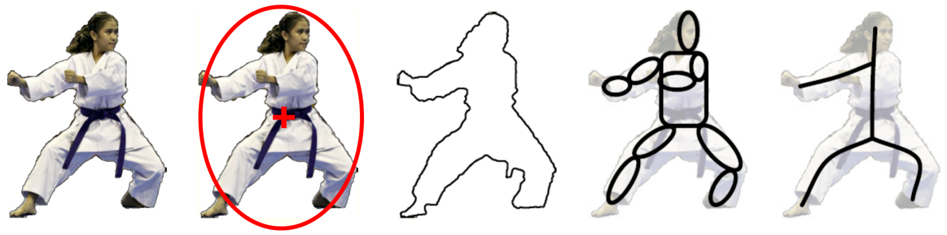

Fig. 7 From left to right: object region, elliptical region, silhouette (contour), part-based, skeletal representations.

3.1.1 Point & Region

Many methods represent an object by a predefined shape around its centroid [?] or a set of points [?]. Object shape can be defined as a rectangle, ellipse, etc. on the imaging plane or on the physical ground plane. Often such shapes are normalized into a fixed size to simplify feature extraction. In general, motion for such representations is modeled by simple translation, similarity, affine or homography transformations. Though primitive shapes are more suitable for representing simple rigid objects, they are also used for tracking non-rigid objects.

3.1.2 Silhouette

Silhouette is the region inside the contouring boundary of object. The most common silhouette representation is in the form of a binary indicator function, which marks the object region by ones and the non-object regions by zeros. For contour based methods, the silhouette is represented either explicitly or implicitly. Explicit representation defines the boundary of the silhouette by a set of control points. Implicitly representation defines the silhouette by means of a function defined on a grid. The most common implicit contour representation is the level sets representation.

A traditional explicit contour structure is composed of a set of points (tuples) on the contour along with a set of spline equations, which are real functions fit to the control points generated from the tuples. A natural choice is cubic spline defined using piecewise cubic polynomials, between the tuples. The piecewise analytical contour form provides the ability to estimate differential descriptors with ease. The analytical form, however, does not let changes in the contour topology (merging and splitting), which is necessary during contour evolution.

Alternatively, contour in an image can be implicitly defined using the level set [?]. In this formalism, the position of a contour is embedded as the zero level set in a two dimensional function. The value at a grid position (e.g. local object coordinates) is commonly set to its distance from the closest contour location and is computed by applying a distance transform. The level set representation has attracted much attention due to its ability to adapt to topology changes, direct computation of differential contour features (e.g. contour curvature), and extendibility to higher dimensions with no change in formulation.

3.1.3 Connected Parts

Articulated objects are composed of parts that are held together with joints. Depending on the granularity of choice, parts of a human body can be grouped into head, torso, arms and legs, which can be defined as geometric primitives, such as rectangles, cylinders and ellipses [?]. The relationship between the parts are governed by kinematic motion models, e.g. joint angle. An important issue that needs explicit handling is the occlusion problem that arises when one part is behind the others making it invisible. Occluded parts constitute missing observations and can be dealt with by applying heuristics or by enforcing learned part-arrangements. In addition, the degree of articulation increases the complexity of the models.

3.1.4 Graph & Skeletal

Another spatial representation is the skeletal models which are used to animate characters and humans in graphics. Skeleton is an articulated structure with a set of curve segments and joints connecting them [?]. Object skeleton can be extracted by applying medial axis transform, which takes the object silhouette and iteratively computes the set of points lying on its topological skeleton such that each point has more than one closest points to the bounding contour. Alternatively, it is termed as the loci of centers of bi-tangent circles that fit within the object silhouette.

Learned skeletal representations have many possible uses. For example, manually constructed skeletal models are often a key component in full-body tracking algorithms. The ability to learn skeletal structure could help to automate the process, potentially producing models more exible and accurate than those constructed manually.

The motion of an articulated object can be described as a collection of rigid motions, one per part, with the added constraint that the motions of connected parts must be spatially coherent. This constraint causes the motion subspaces of two connected objects to intersect, making them linearly dependent. In particular, for each pair of connected parts, the motion subspaces share one dimension (translation) if they are joined at a point and two dimensions (translation and one angle of rotation) if they are joined at an axis of rotation.

3.1.5 Spatiotemporal

While spatial representations lacks motion indicators, there are specific representations that are defined in the spatiotemporal space and inherently convey the motion information. Spatiotemporal representations can be extracted by local analysis or by looking at the space-time cube globally. Local representations are composed of a set of points that present characteristic motion and appearance content. The point set is commonly treated as a bag of features without temporal order. Space-time cube [?], which is generated by stacking video frames, can be considered a volumetric 3D image. An important observation about the space-time cube is that aside from providing temporal information, it also carries a unique view geometric information when we have two cameras observing a common scene. Alternative to using temporally sparse point sets, one can consider progression of the point set in time by extracting their trajectories [?]. Trajectory representation is constituted of a temporally ordered point series, which represent the position of a point starting from its initial observation at until it disappears from the scene.

3.2 Model Descriptors

Descriptors are the mathematical embodiments of object regions. The size of the region, dynamic range, imaging noise and artifacts play a significant role in achieving discriminative descriptors. Generally speaking, the larger the object region, the more discriminative the descriptor will be.

A major concern for most descriptors is the lack of a competent similarity criterion that captures both statistical and spatial properties, i.e., most approaches either depend only on the color distributions or structural models. Many different representations, from aggregated statistics to appearance models, have been used for describing objects. Color histograms are popular representations of non-parametric density, but they disregard the spatial arrangement of the feature values. Moreover, they do not scale to higher dimensions due to exponential size and sparsity. Appearance models map the

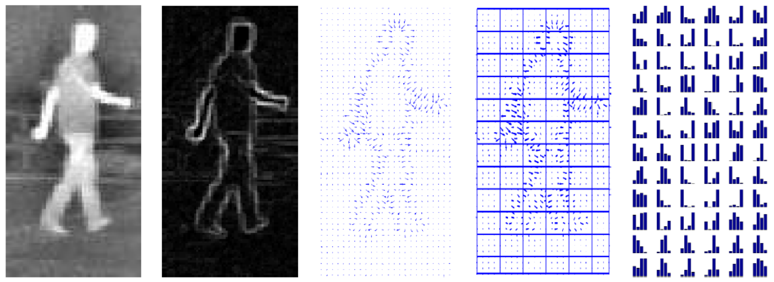

Fig. 8 HOG concatenates the bins of the local gradient histograms into a vector form (Courtesy of E. Benhaim).

image features onto a fixed size window. Since the dimensionality is a polynomial in the number of features and the window size, only a relatively small number of features can be used. Appearance models are highly sensitive to the pose, scale and shape variations.

3.2.1 Template

Templates are the most intuitive and commonly adopted descriptors and often formed from geometric shapes or silhouettes. A unique property of a template is that it is an ordered list of appearance observations inside its region. This property naturally provides the template descriptor with the capability to carry both spatial and appearance information. They can be 2D (spatial) or 3D (spatiotemporal) depending on their use, and commonly have a shape in the form of a geometric primitive, such as a rectangle, square, ellipse, circle or their 3D versions. They are often centered around a point and defined by weighted spatial kernels that may have varying scalars at different pixels within the kernel. The kernel function represents a convolution of the geometric primitive with the template. The motion of the kernel from one frame to the next follows a parametric model including translation, conformal and affine transformations.

Templates, however, only encode the object appearance generated from a single view. Thus, they are more suitable for problems where the viewing angle of the camera and the object pose remains constant or changes very slowly.

3.2.2 Histogram, HOG, SIFT

Histogram is a distribution based descriptor that estimates the probability distribution from the observations within a spatial or spatiotemporal region defined by a template, silhouette or a volume. The observations considered can be raw color values, derivative information or texture measures. Color distributions are generally estimated non-parametrically by a histogram, or parametrically by mixture models.

The histogram, which is a common choice, can be generated by first defining the number of bins (quantization levels) and counting the number of observations that fall into respective bins. While a histogram can be generated using raw color or intensity values, they may need to be processed, such as mean color adjustment, prior to estimating the histogram to remove the illumination and shadowing effects. This adjustment can be achieved by subtracting the mean color computed in the neighborhood of the region of interest.

Alternative to using color values, image gradient is adapted for generating a distribution based descriptor. Two closely related approaches are the Scale Invariant Feature Transform (SIFT) descriptors [?] and the Histogram of Oriented Gradients (HOG) [?, ?]. Both of these approaches compute the gradient of intensity and construct a histogram of gradient orientations weighted by the gradient magnitude (Figure ??). Shared steps between the two approaches can be listed as follows:

```
**Input**: Image and regions of interest
**Output**: Histograms
foreach region $R$ do
    foreach $(x, y) \in R$ do
        compute the gradient:
        $\nabla I(x,y) = (I_x, I_y) = \big(I(x-1,y) - I(x+1,y), I(x,y-1) - I(x,y+1)\big)$;
        compute gradient direction: $\theta(\nabla I(x,y) = \arctan(I_x, I_y)$;
        compute gradient magnitude: $|\nabla I(x,y)| = (I_x * I_x + I_y * I_y)^{1/2}$;
        if $|\nabla I(x,y)| \geq \tau$ then
            Increment histogram bin for $\theta(\nabla I(x,y))$;
        end
    end
    smooth histogram;
end
```

Aside from the common steps outlined above, SIFT approach computes the major orientation from the resulting histogram and subtracts it from computed orientations to achieve rotation invariance. HOG, on the other hand does not perform orientation normalization. SIFT generates more 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 [?] that SIFT is more resilient to image deformations. More recently the color based SIFT method is introduced and is widely adopted [?].

Another possibility is to generate the histogram defining the shape of the object. Shape histograms model the spatial relation between the pixels lying on the contour. The spatial relation between a reference point on the contour with respect to other points is modeled by a histogram, and a set of such histograms, which are generated by taking all or randomly chosen contour points individually as a reference, provides a distribution based descriptor. The spatial relation between two points is measured by computing the angle and magnitude of the vector joining them. Similarly, shape context uses a concentric circular template centered on a reference contour point, which provide the bins of the histogram in the polar coordinates [?].

In all cases, a commonly used approach to compare histograms is Bhattacharya distance.

3.2.3 Region Covariance

The region covariance matrix proposes a natural way of fusing multiple features. Its diagonal entries represent the variance of each feature and the nondiagonal entries represent the correlations. The noise corrupting individual samples are largely filtered out with an average filter during covariance computation.

For a given region R, let $F(x, y, i)$ be the pixel feature values. Features $i = 1, .., d$ can be any mapping including pixel's Cartesian coordinates, intensity, color, gradient, texture, filter responses, etc. For instance, $F(., ., 1)$ can be assigned to the x-gradient: $F(x, y, 3) = I_x(x, y)$. Region covariance descriptor C represents the region R with the $d \times d$ covariance matrix of the pixel-wise features

$$C = \begin{bmatrix} c_{11} & \cdots & c_{1d} \\ \vdots & \ddots & \vdots \\ c_{d1} & \cdots & c_{dd} \end{bmatrix} \quad \text{and} \quad c_{ij} = \frac{1}{N_R - 1} \sum_{n=1}^{N_R} \left(F(x_n, y_n, i) - \mu_i\right)\left(F(x_n, y_n, j) - \mu_j\right) \qquad (1)$$

where μ is the mean of the corresonding feature for all N_R pixels in the region.

There are several advantages of using covariance matrices as region descriptors. It is a natural way of fusing multiple features. A single covariance matrix extracted from a region is usually enough to match the region in different views and poses. The covariance of a distribution is enough to discriminate it from other distributions. The covariance matrices are low-dimensional compared to other region descriptors and due to symmetry it has only $(d^2 + d)/2$ different values. This provides significant robustness to object variations while keeping the descriptor efficiently discriminative. Whereas if the same region is described with joint histograms a total of B^d values, where B is the number of histogram bins, are needed, which renders such joint histograms very fragile.

Distance Calculation: The covariance matrices do not lie on Euclidean space. For example, the space is not closed under multiplication with negative scalers. Most of the common machine learning methods work on Euclidean spaces and therefore they are not suitable for our features. The nearest neighbor algorithm which will be used in the following sections, only requires a way of computing distances between feature points. A distance measure is proposed in [?] to measure the dissimilarity of two covariance matrices

$$\rho^2(C_1, C_2) = \sum_{i=1}^{d} \ln^2 \lambda_i(C_1, C_2) \tag{2}$$

where $\{\lambda_i(C_1, C_2)\}$ are the generalized eigenvalues of C_1 and C_2. The distance measure follows from the Lie group structure of positive definite matrices and an equivalent form can be derived from the Lie algebra of positive definite matrices. We refer the readers to [?] for a detailed discussion on the distance metric.

3.2.4 Ensembles & Eigenspaces

Ensemble descriptors keep a combination of weak or partial descriptors. More specifically, the ensemble tracking [?] works by constantly updating a collection of weak classifiers to separate the object from its background. The weak classifiers can be added or removed at any time to reflect changes in the object appearance or incorporate new information about the background. Hence, object is not represented explicitly, instead an ensemble of classifiers is used to determine if a pixel belongs to the object or not. Each weak classifier is trained on positive and negative examples where, the examples coming from the object are positive examples and examples coming from the background are negative examples. The strong classifier is then used to classify the pixels in the next frame, producing a confidence map of the pixels, where the classification margin is used as the confidence measure. The peak of the map is assumed to be the location of the object in the current frame. Once the detection for the current frame is completed, a new weak classifier is trained on the new frame, added to the ensemble, and the process is repeated all over again.

Given a set of images, eigenspace approaches construct a small set of basis images that characterize the majority of the variation in the training set and can be used to approximate any of the training images. For each image in a training set of p images a 1D column vector is constructed by scanning the image in the standard lexicographic order. Each of these 1D vectors becomes a column in a data matrix. The number of training images is assumed to be less than the number of pixels, and Singular Value Decomposition (SVD) is used to decompose the data matrix into 1) an orthogonal matrix of the same size as the data matrix representing the principal component directions in the training set, 2) a diagonal matrix with singular values sorted in decreasing order along the diagonal, and 3) an orthogonal matrix that encodes the coefficients to be used in expanding each column of the data matrix in terms of the principal component directions.

It is possible to approximate some new image vector in terms of the orthogonal matrix columns that have comparably higher singular values by taking the dot product of the image vector and the columns of the orthogonal matrix. This amounts to a projection of the input image onto the subspace defined by the largest basis vectors. The eigenspace can be thought of as a compact view-based object representation that is learned from a set of input images. Previously observed views of an object can be approximated by a linear combination of the basis vectors. This can be thought of as matching between the eigenspace and the image.

3.2.5 Appearance Models

Active appearance models are generated by simultaneously modeling the object shape and appearance [?]. 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.

3.3 Model Features

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 off-line 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. 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 acquisition, the RGB (red, green, blue) color space is usually used to represent color. However, the RGB space is not a perceptually uniform, that is, the difference between the colors in the RGB space does not correspond to the color differences perceived by the humans [?]. Instead, YUV and LAB are perceptually uniform, while HSV (Hue, Saturation, Value) is an approximately uniform color space. However, these color spaces are sensitive to noise. 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.
- **Gradient:** Object boundaries usually generate strong changes in image intensities. Edge gradient identifies these changes. An important property of edges is that they are less sensitive to illumination changes as 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 [?].
- **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 a brightness constraint, which assumes "brightness constancy" of corresponding pixels in consecutive frames and is usually computed using the image derivatives [?, ?]. Optical flow is commonly used as a feature in motion-based segmentation and tracking applications. A comparison of the popular optical flow techniques can be find in [?].
- **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 including gray level co-occurrence matrices, Law's texture measures (twenty five 2D filters generated from five 1D filters corresponding to level, edge, spot, wave and ripple), wavelets, Gabor filters, and steerable pyramids. A detailed anaysis of texture features can be found in [?]. Similar to edge features, the texture features are less sensitive to illumination changes as compared to color.
- **Corner Points:** Corner points are one of the earliest and most commonly used features due to its low computational complexity and ease of implementation. Harris corner detector, like many others, defines texture-content by conjecturing that the change in the color content of pixels in

the locality of a candidate interest point should be high:

$$E(x,y) = \sum_u \sum_v \Big(I(x+u, y+v) - I(x,y) \Big)^2. \tag{3}$$

The Taylor series approximation of this equation around (x, y) results in

$$E(u,v) = [u\, v] \underbrace{\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}}_{\mathtt{M}} \begin{bmatrix} u \\ v \end{bmatrix}. \tag{4}$$

This equation contains the commonly termed structure tensor $\mathtt{M}$, which is a second moment computed from the template around the candidate. This matrix defines an ellipse with minor and major axes denoted by its eigenvectors and their extent by respective eigenvalues. The eigenvalues, λ_i of $\mathtt{M}$ are computed from its characteristic equation: $\lambda^2 + \det(\mathtt{M}) - \lambda \cdot \mathrm{trace}(\mathtt{M}) = 0$, which suggests that using determinant and trace of $\mathtt{M}$ should suffice in marking interest points as stated in [**?**]. Therefore, the traditional texture content measure $\min(\lambda_1, \lambda_2)$ can be approximated by $\det(\mathtt{M}) - c \cdot \mathrm{trace}(\mathtt{M})^2$ for constant c. The texture content measure is computed for all pixels and it is subjected to nonmaximal suppression which removes weak interest point candidates and eliminates multiple candidates in small neighborhoods.
Harris detector, when applied in scale space, such as by convolving the image with a set of different scaled Gaussian filters, provides feature points at multiple scales. The interest points coexisting at different scales can be combined to provide scale-invariant interest points. Considering that the shape tensor is invariant to rotations, Harris feature becomes invariant to similarity transform. The spatial point detection scheme outlined for Harris detector is later extended to spatiotemporal coordinates by introducing the time as an additional dimension to the formulation [**?**]. Limitations of the Harris feature include its inability to locate interest points at subpixel level, and difficulty setting the number of interest points it detects.

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 communities. Automatic feature selection methods can be divided into filter methods and wrapper methods [**?**]. The filter methods try to select the features based on a general criteria, e.g. the features should be uncorrelated. The wrapper methods select the features based on the usefulness of the features in a specific problem domain, e.g. 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 number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called the principal components.

A common wrapper method of selecting the discriminatory features is boosting. More specifically, AdaBoost finds 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. Adaboosts discover a weighted combination of classifiers (representing features) that maximize the classification performance of the algorithm. These weights indicates the discriminative power of the features.

4 Object Tracking

Given the detected object, it is then the tracker's task to perform find its correspondence in the following frames while constructing object's trajectory. It is an essential component of several vision applications as well as the video analytics for business applications. Object tracker may also provide the complete region in the image that is occupied by the object.

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 updating object location and region information obtained from previous frames.

Robust and accurate tracking of a deforming, non-rigid and fast moving object without getting restricted to particular model assumptions presents a major challenge. One can simplify tracking by imposing constraints on motion and appearance of objects. For example, almost all tracking algorithms assume the object motion to be 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 motion, appearance and shape of the object be modeled? The answers to these questions depend on the context/environment in which the tracking is being performed, and the end use for which the tracking information is being sought.

For business applications, they should perform mostly partitioned indoors spaces where sudden illumination changes, for instance due to on-and-off of a light switch, can occur. They are expected to handle significant object size changes due to oblique views and severe occlusions due to usually lower camera heights. They need to resolve multiple-object tracking even often the descriptors are insufficient as people tend to dress in for instance dark clothes in business environments.

We give a brief description of most common tracking techniques in the following.

4.1 Template Matching

The most common approach in this category is template (or blob) matching. Template matching is a brute-force method of searching the image for a region similar to the object template defined in the previous frame. The position of the template in the current image is computed by a similarity measure, e.g. cross correlation. Usually image intensity or color features are used to form the templates. Since image intensity is very sensitive to illumination changes, image gradients can also be used as features. Note that instead of region templates, other object descriptors for instance, histograms or mixture models can also be used for matching, with the same spirit of cross correlation based exhaustive search.

A limitation of template matching is high computation cost due to the brute-force search. To reduce the computational cost, search window is usually limited to the vicinity of its previous position [**?**].

4.2 Density Estimation: Mean-Shift

Mean-shift [**?**] is a nonparametric density gradient estimator to find the image window that is most similar to the object's color histogram in the current frame. It iteratively carries out a kernel based search starting at the previous location of the object as shown in Figure **??**.

The mean-shift tracker maximizes the appearance similarity iteratively by comparing the histogram of the object h_o and the histogram of the window around the candidate object location $h_\star$. Histogram distance is defined in terms of the Bhattacharya distance. Minimizing the Bhattacharya distance or alternatively maximizing the Bhattacharya coefficient $\rho(h_o, h_\star) = \sum_{b=1}^{B} [h_o(b) h_\star(b)]^{1/2}$

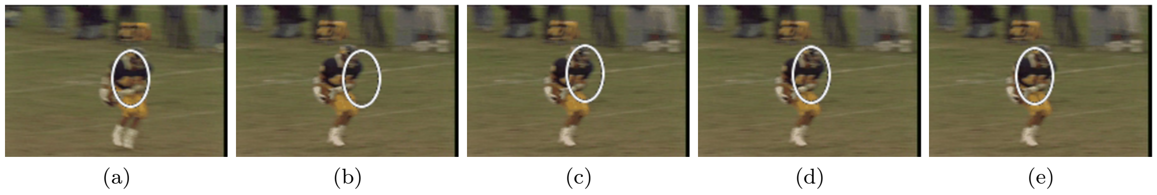

Fig. 9 Mean-shift tracking iterations: estimated object location at time $t-1$, (b) Frame at time t with initial location estimate using the previous object position, (c), (d) location update using mean shift iterations, (e) final object position at time t.

and expanding it to Taylor series suggests that the new location of the object can be iteratively computed [?] by estimating the likelihood ratio between the model h_o and candidate histograms $h_\star$.

At each iteration the candidate window is shifted towards the direction that the Bhattacharya coefficient is maximum. The iterative process is repeated until the increase in the Bhattacharya coefficient becomes insignificant:

- Compute the candidate histogram $h_\star(\mathbf{m}_{t-1})$,
- Calculate $\rho(h_o, h_\star(\mathbf{m}_{t-1}))$ and the new weights $w_n = \sum_b^B \delta[I(x_n, y_n)_B - b]\sqrt{h_o/h_\star(\mathbf{m}_{t-1})}$,
- Find new location $\mathbf{m}_t$ by,

$$\mathbf{m}_t = \frac{\sum_n^{N_R} w_n \dot{K}(||\mathbf{x}_n - \mathbf{m}_{t-1}||)\mathbf{x}_n}{\sum_n^{N_R} w_n \dot{K}(||\mathbf{x}_n - \mathbf{m}_{t-1}||)}, \tag{5}$$

- Stop if $\rho(h_o, h_\star(\mathbf{m}_t)) < \rho(h_o, h_\star(\mathbf{m}_{t-1}))$ or $||\mathbf{m}_t - \mathbf{m}_{t-1}|| < \epsilon$, else $\mathbf{m}_t \leftarrow \mathbf{m}_{t-1}$ and iterate.

Above $\mathbf{x}_n = [x_n, y_n]^T$, $\mathbf{m}_t = [m_x, m_y]^T$ is the estimated location of the object, t is the iteration index, $I(x_n, y_n)_B$ is the bin of the pixel value $I(x, y)$, and $\dot{K}$ is the derivative of the kernel function K, which can be a 2D Gaussian. The kernel state is defined by the centroid of the object in spatial coordinates, such that the estimation process results in the new object position. Alternatively, the state can be defined by the scale, orientation and position of the object [?].

In other words, 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, a weighting scheme is defined by a spatial kernel which gives higher weights to the pixels closer to the object center [?] extended the mean shift tracking approach used a joint spatial-color histogram instead of just color histogram.

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, the mean-shift tracker requires that a portion of the object is inside the previous object region upon initialization. Even though there are variants [?] to improve its localization by using additional modalities, the success of the mean-shift strongly depends on the discriminating power of the histograms. To overcome this shortcoming, a covariance matrix representation version is proposed in [?], and a multiple-kernel version in [?].

4.3 Regression

Regression refers to understand the relationship between multiple variables. Linear regression assumes the relationship depends linearly on a model in which the conditional mean of a scalar variable given the other variables is an affine function of those variables. Numerous procedures have been developed for parameter estimation and inference in linear regression. Here a least squares estimator is described for object tracking, details can be found in [?].

Suppose (α_i, X_i) are the pairs of observed data $\alpha \in \mathbb{R}^d$ in vector space and the corresponding points on the manifold $X \in \mathcal{M}$. The regression function φ maps the vector space data onto the

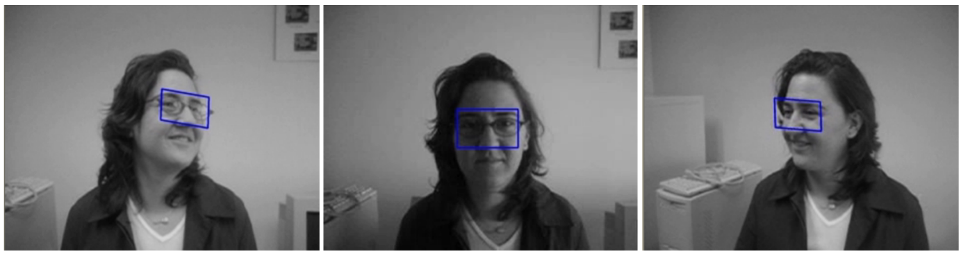

Fig. 10 Regression tracking on manifold for a given region. Note that the tracking is still valid even the region undergoes out-of-plane rotations.

manifold $\varphi : \mathbb{R}^d \mapsto \mathcal{M}$. An objective function is defined as the sum of the squared geodesic distances between the estimations $\varphi(\alpha_i)$ and the points X_i

$$J = \sum_i \Delta^2 \left[\varphi(\alpha_i), X_i\right]. \tag{6}$$

Assuming a Lie algebra on the manifold can be defined, the objective function can be approximated as

$$J = \sum_i \left\|\log\left[\varphi^{-1}(\alpha_i)X_i\right]\right\|^2 \approx \sum_i \left\|\log\left[\varphi(\alpha_i)\right] - \log\left[X_i\right]\right\|^2 \tag{7}$$

up to the first order terms. The regression function φ can be written as

$$\varphi(\alpha_i) = \exp\left(\alpha_i^T \Omega\right) \tag{8}$$

to learn the function $\Omega : \mathbb{R}^d \mapsto \mathbb{R}^r$ which estimates the tangent vectors $\log(X_i)$ on the Lie algebra where Ω is the $d \times r$ matrix of regression coefficients. Thus, the objective function (**??**) becomes

$$J = \sum_i \left\|\alpha_i^T \Omega - \log\left[X_i\right]\right\|^2 \tag{9}$$

Let $\mathbf{X}$ be the $k \times d$ matrix of initial observations and $\mathbf{Y}$ be the $k \times r$ matrix of mappings to the Lie algebra

$$\mathbf{X} = \begin{bmatrix} [\alpha_1]^T \\ \vdots \\ [\alpha_k]^T \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} [\log(X_1)]^T \\ \vdots \\ [\log(X_k)]^T \end{bmatrix} \tag{10}$$

Substituting (**??**) into (**??**), one can obtain

$$J = tr[(\mathbf{X}\Omega - \mathbf{Y})^T(\mathbf{X}\Omega - \mathbf{Y})] \tag{11}$$

where the trace replaces the summation in (**??**). Differentiating the objective function J with respect to Ω, the minimum is achieved at $\Omega = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$. To avoid overfitting, additional constraints on the size of the regression coefficients can be introduced

$$J = tr[(\mathbf{X}\Omega - \mathbf{Y})^T(\mathbf{X}\Omega - \mathbf{Y})] + \beta\|\Omega\|^2 \tag{12}$$

which is called the ridge regression. The minimizer of the cost function J is given by $\Omega = (\mathbf{X}^T\mathbf{X} + \beta\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$ where $\mathbf{I}$ is an $d \times d$ identity matrix. The regularization coefficient β determines the degree of shrinkage on the regression coefficients.

At the initialization of the object, the affine motion tracker estimates a regression function that maps the region feature vectors to the hypothesized affine motion vectors by first hypothesizing a set of random motion vectors within the given bounds, determining the transformed regions for

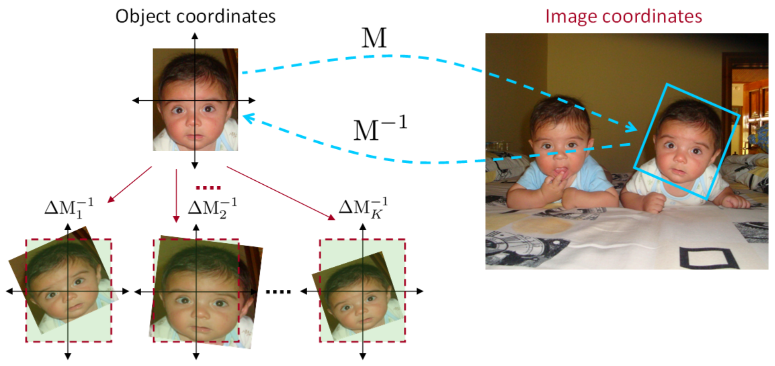

Fig. 11 Random transformations are applied in object coordinates to generate the training features for regression function estimation.

these motions, and then computing the corresponding features within each warped region. In the tracking time, it extracts the feature vector only for the previous object region location and applies the learned regression function. Sample affine tracking results are shown in Figure **??**.

Let M transforms a unit square at the origin to the affine region enclosing the target object $[x\ \ y\ \ 1]_I^T = M[x\ \ y\ \ 1]_O^T$ where the subscripts indicate the image and object coordinates respectively. The inverse M^{-1} is an affine motion matrix and transforms the image coordinates to the object coordinates. The aim of tracking is to estimate the transformation matrix M_t, given the previous images and the initial transformation M_0. The transformations are modeled incrementally

$$M_t = M_{t-1}.\Delta M_t \tag{13}$$

and estimate the increments ΔM_t at each time. The transformation ΔM_t corresponds to motion of target from time $t-1$ to t in the object coordinates.

Suppose the target region is represented with orientation histograms computed at a regular grid inside the unit square in object coordinates, i.e with $\alpha(I(M_t^{-1})) \in \mathbb{R}^d$ where d is the dimension of the descriptor. Given the previous location of the object M_{t-1} and the current observation I_t, the new transformation ΔM_t by the regression function is estimated as

$$\Delta M_t = \varphi(\alpha(M_{t-1}^{-1})). \tag{14}$$

The problem reduces to learning and updating the regression function φ. During the learning step, a training set of k random affine transformation matrices $\{\Delta M_j\}_{j=1...k}$ are generated around the identity matrix to learn the regression function Ω as illustrated in Figure **??**. The approximation is good enough since the transformations are in a small neighborhood of the identity. The training set consists of samples $\{\alpha_j, \Delta M_j\}_{j=1...k}$. Since number of samples is smaller than the dimension of the feature space, $k < d$, the system is underdetermined. To relieve this, the ridge regression is applied to estimate the regression coefficients.

Since objects can undergo appearance changes in time, it is necessary to adapt to these variations. The model update achieves reestimating the regression function. During tracking, a set of random observations are generated at each frame with the same method described above. The observations stored for most recent frames constitute the update training set. More details and an importance sampling based adaptation can be found in [**?**].

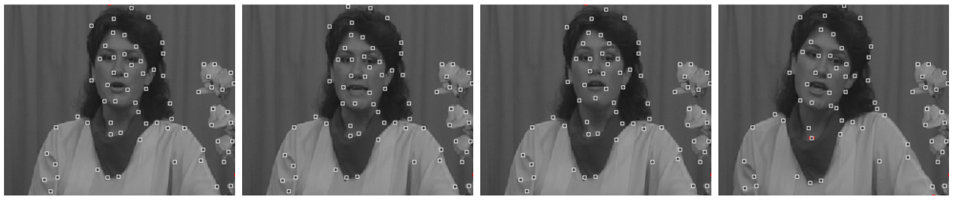

Fig. 12 Tracking features using the KLT tracker.

4.4 Motion Estimation

Optical flow methods are used for generating dense flow fields by computing the flow vector of each pixel under the brightness constancy constraint, $I(x,y,t) - I(x+dx, y+dy, t+dt) = 0$. This computation is always carried out in the neighborhood of the pixel either algebraically or geometrically. Extending optical flow methods to compute the translation of a rectangular region is trivial. In [**?**], the KLT tracker, which iteratively computes the translation (du, dv) of a region (e.g. 25×25 patch) centered on an interest point, is proposed as:

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

This equation is similar in construction to the optical flow [**?**]. 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 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 results obtained by the KLT tracker are shown in Figure **??**.

In [**?**] an object tracker that tracks an object as a three component mixture, consisting of the stable appearance features, transient features and noise process is proposed. The stable component identifies the most reliable appearance for motion estimation, i.e. the regions of 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, which 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, rotation and scale parameters. 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.

4.5 Kalman Filtering

Let the location of a moving object is defined by a sequence of states X_t. The change in state over time is governed by the linear system,

$$X_t = A_t X_{t-1} + \eta, \tag{15}$$

where η is white noise with covariance Σ^{η}. The relationship between the measurement and the state is specified by the measurement equation $Y_t = D_t X_t + \xi$, where D_t is the measurement matrix and ξ is the white noise with covariance Σ^{ξ} independent of η. 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 $p(X_t|Y_{1,..,t})$.

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 probability distribution of the current state. Then, the correction step employs the likelihood function of the current measurement to compute the posterior probability distribution.

Kalman filter is used to estimate the state of a linear system where the state is assumed to be distributed by a Gaussian. The prediction step of the Kalman filter uses the state model to predict the new state of the variables:

$$\begin{aligned} X_t^p &= A X_{t-1} \\ \Sigma_t^p &= A \Sigma_{t-1} A^T + \Sigma_t^{\eta}, \end{aligned}$$

where X_t^p and Σ_t^p are the state and the covariance predictions at time t. A_t is the state transition matrix which defines the relation between the state variables at time t and $t-1$. Similarly, the correction step uses the current observations Y_t to update the object's state:

$$\begin{aligned} X_t &= X_t^p + G_t[Y_t - D_t X_t^p], && (16) \\ G_t &= \Sigma_t^p D_t^T [D_t \Sigma_t^p D_t^T + \Sigma_t^{\xi}]^{-1}, && (17) \\ \Sigma_t &= \Sigma_t^p - G_t D_t \Sigma_t^p, \end{aligned}$$

where G_t is the Kalman gain, which is used for propagation of the state models. Note that the updated state, X_t, is still distributed by a Gaussian. In case system is nonlinear, it can be linearized using the Taylor series expansion to obtain the Extended Kalman Filter. Similar to Kalman filter, Extended Kalman filter assumes that the state is distributed by a Gaussian.

Kalman filter has been extensively used in early vision community for tracking, e.g. to track points in noisy images [**?**] and to estimate 3D trajectory from 2D motion [**?**]. When tracking multiple objects using particle and Kalman filters, one needs to deterministically associate the most likely measurement for a particular object to that object's state, i.e. 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.

4.6 Particle Filtering

One limitation of the Kalman filter is the assumption that the state variables are Gaussian. Thus, Kalman filter will give a poor estimations of state variables that do not follow Gaussian distribution. This limitation can be overcome by using particle filtering [**?**, **?**].

In particle filtering, the conditional state density $p(X_t|Y_t)$ at time t is represented by a set of N_s samples $\{s_{t,n}\}$ where $n = 1,..,N_S$ with weights $\pi_{t,n}$ corresponding to sampling probability. These samples are called as particles. The weights define the importance of a sample, i.e. its observation frequency [**?**]. To decrease computational complexity, for each tuple (s_n, π_n) a cumulative weight c_n is also stored, where $\sum c_n = 1$. The new samples at time t are drawn from $S_{t-1} = \{(s_{t-1,n}, \pi_{t-1,n}, c_{t-1,n})\}$ at the previous time $t-1$ step based on different sampling schemes. The most common sampling scheme is the "importance sampling" which can be stated as follows:

- Selection: Select N_S random samples $\hat{s}_{t,n}$ from 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 $\hat{s}_{t,n} = s_{t-1,j}$.
- 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 $f(s) = s$.
- Update: Weights $\pi_{t,n}$ corresponding to the new samples $s_{t,n}$ are computed using the measurements Y_t by $\pi_{t,n} = p(Y_t|X_t = s_{t,n})$, where the probability can be modeled as a Gaussian density.

Using the new samples S_t one can estimate the new object position by $\sum_{n=1}^{N_S} \pi_{t,n} f(s_{t,n}, W)$. Particle filter based trackers can be initialized by either using the first measurements, $s_{0,n} \sim X_0$, with weight $\pi_{0,n} = \frac{1}{N_S}$ 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 filtering suffers from sample degeneracy and impoverishment, especially for higher dimensional representations due to the importance sampling.

Note that the particle filter described above assume a single measurement at each time instant, i.e. the state of single object is estimated. Tracking multiple objects requires a joint solution of data association and state estimation problems. There exist several statistical data association techniques to tackle this problem. A detailed review of these techniques can be found in [?]. Joint Probability Data Association Filtering (JPDAF) and Multiple Hypothesis Tracking (MHT) are two widely used techniques for data association explained in the following.

4.6.1 Joint Probability Data Association Filter

Suppose we have k tracks and at time t and $\{Y_{1t}, .., Y_{mt}\}$ are the m measurements. We need to assign these measurements to the existing tracks. Let Γ be a set of assignments. It is assumed that the number of tracks will remain constant over time. Let γ_{ij} be the innovation associated with the track j due to the measurements. The JPDAF associates all measurements with each track [?]. The combined weighted innovation is given by

$$\gamma_j = \sum_{i=1}^{m} z_{ij} \gamma_{ij}, \tag{18}$$

where z_{ij} is the posterior probability that the measurement i originated from the object associated with track j and is given as:

$$z_{ij} = \sum_{\Gamma} p(\Gamma_j | Y_{1t}, .., Y_{mt}) L_{i,j}(\Gamma), \tag{19}$$

where L_{ij} is the indicator variable, $i = 1, .., m$ and $j = 1, .., k$. It is equal to one if the measurement Y_{it} is associated with track j, otherwise it is zero. The weighted innovation given in (**??**) can be plugged in the Kalman filter update equations (**??**) for each track j.

4.6.2 Multiple Hypothesis Tracking

The major limitation of the JPDAF algorithm is its inability to handle new objects entering the scene or already tracked objects exiting the scene. Since the JPDAF algorithm performs data association of a fixed number of objects being tracked over two frames, serious errors can arise if there is a change in the number of objects. The MHT algorithm does not have this shortcoming.

As 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. The final track of the object is the most likely set of correspondences over the time period of its observation. The MHT algorithm has the ability to handle occlusions, i.e. continuation of a track even if some of the measurements from an object are missing.

An MHT 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 scene, 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 scene, 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.

Since the MHT makes associations in a deterministic sense and exhaustively enumerates all possible associations it is computationally exponential both in memory and time. To reduce the computational load, a Probabilistic Multiple Hypotheses Tracker (PMHT) is proposed by [?] in which the associations are considered to be statistically independent random variables. Thus, there is no requirement for exhaustive enumeration of associations. Alternatively, [?] use Murty's algorithm [?] to determine k-best hypotheses in polynomial time for tracking points. Particle filters to handle multiple measurements to track multiple objects have also been proposed, e.g. [?] where data association is handled in a similar way as in PMHT, however the state estimation is achieved through particle filters.

4.7 Silhouette Tracking

The goal of a silhouette based object tracker is to find the object region by means of an object model generated using the previous frames. Silhouette trackers can be categorized into two categories: matching and contour evolution. Shape matching approaches search for the object silhouette in the current frame. Contour evolution approaches, on the other hand, track an initial contour to its new position in the current frame by either using the state space models or direct minimization of some energy functional.

Important factors to distinguish different silhouette trackers are: What features are used? How occlusion is handled? and If the training is required or not? Moreover some algorithms only use information about the silhouette boundary for tracking while other use the complete region inside the silhouette. Generally the region based approaches are more resilient to noise.

The most important advantage of tracking silhouettes is their flexibility to handle a large variety of object shapes. Occlusion handling is another issue 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. Another important aspect related to silhouette trackers is their capability of dealing with object split and merge. For instance, while tracking a silhouette of 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.

4.7.1 Shape Matching

Shape matching often assumes similarity transform from the current frame to the next, therefore nonrigid object motion is not explicitly handled. It is usually carried out by background subtraction.

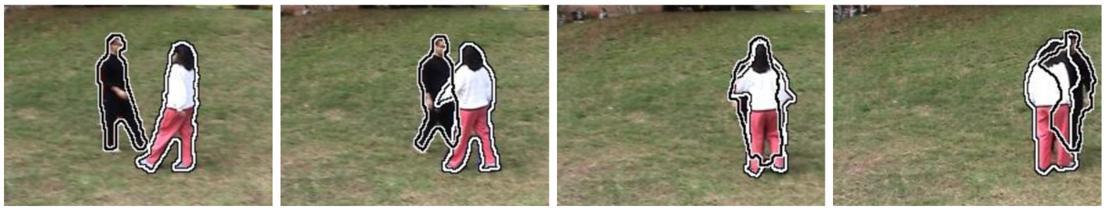

Fig. 13 Contour tracking results in presence of occlusion using the method proposed in [?](©[2004] IEEE).

Once the object silhouettes are extracted, matching is performed by computing some distance between the object models associated with each silhouette. 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.

[?] performs shape matching using an edge based representation. Hausdorff distance is used to construct a correlation surface, from which the minimum is selected as the new object position. In the context of matching using an edge based model, Hausdorff distance measures the most mismatched edges. Due to this, this 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 changes, such that, removing the edges corresponding to arms and legs will improve the tracking performance.

In contrast to looking for possible matches in consecutive frames, shape matching can be performed by computing the flow vectors for each pixel inside the silhouette, such that the flow, which is dominant over the entire silhouette, is used to generate the silhouette trajectory.

4.7.2 Contour Evolution

Contour evolution requires 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 for contour shape and motion. However, explicit representations do not allow topology changes such as split or merge. On the other hand, the second approach directly evolves the contour by minimizing the contour energy using direct minimization techniques, such as gradient descent.

The state spaces are updated at each frame 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 [?] defines the state space 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 Kalman filter. The correction step uses the image observations which are defined in terms of the image gradients.

In [?], the state space is defined in terms of spline shape parameters and affine motion parameters. The measurements consist of image edges computed in the normal direction to the contour. The state is updated using a particle filter. [?], extends the particle filter to track multiple objects by including the "exclusion principle" for handling occlusion. [?] proposes 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.

Contour evolution minimizes an energy functional either by greedy methods or by gradient descent. The contour energy is defined in terms of temporal gradient [?], or appearance statistics generated from, for instance a band around the object boundary and the background regions [?]. The width of the band serves as a means to combine region and boundary based contour methods contour tracking methods into a single framework. The object shape and its changes are modeled using level sets to resolve the object occlusions during the course of tracking. Sample results of the appearance statistics based approach are given in Figure **??**.

5 Final Observations

Tracking approaches that employ a stable model can only accommodate small changes in the object appearance but do not explicitly handle severe occlusions or continuous appearance changes.

Occlusion, either partial or full, can be classified into self occlusion, inter-object occlusion and occlusion by the background scene structure. Self occlusion occurs when one part of the object occludes another, especially for articulated objects. Inter-object occlusion occurs when two objects being tracked occlude each other, which is the common case in surveillance video. Similarly, occlusion by the background occurs when a structure in the background e.g. a column, a divider, etc., occludes the tracked objects. Generally, for inter-object occlusion, the multi-object trackers can exploit the knowledge of position and appearance of the occluded and occluding objects 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 full occlusions during tracking is to assume motion consistency, and in case an occlusion is detected, to keep on predicting the object location till the object reappears. Among such predictors, Kalman filter can be given as an example. Occlusion can also be implicitly resolved during generation of object tracks. The chance of occlusion can be reduced by an appropriate selection of camera positions. For instance, if the cameras are mounted on for birds eye view of the scene, most occlusions can be eliminated. Multiple cameras viewing the same scene can also be used to resolve object occlusions during tracking.

Multi-camera tracking methods have demonstrated superior tracking results as compared to single camera trackers in case of persistent occlusion between the objects. In many situations it is not possible to have overlapping camera views due to limited resources or large areas of interest. Non-overlapping multi-camera tracking has to to deal with sparse object observations. Therefore additional assumptions have to be made about the object speed and the path in order to obtain the correspondences across cameras. Methods that establish object correspondence 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

Object appearance changes can be included in the model update by introducing noise and transient models. Despite explicit modeling of noise and transient features, trackers often perform poorly, or even lose tracking, in cases when the performer suddenly turns around during an action and reveals a completely different appearance, which has not been learned before.

A potential approach to overcome this limitation is to learn different views of the object and later use them during tracking. In addition, a tracker that takes advantage of contextual information to incorporate general constraints on shape and motion of objects will usually perform better than the one that does not exploit this information. The capability to learn object models online may greatly increase the applicability of a tracker. Unsupervised learning of object models for multiple non-rigid moving objects from a single camera remains an unsolved problem. One interesting direction, that has largely been unexplored, is the use of semi-supervised learning techniques for modeling objects. These techniques (co-training, transductive SVMs, constrained graph cuts) do not require prohibitive amount of training data.

Overall, additional sources of information, in particular prior and contextual information, should be exploited whenever possible to attune the tracker to the particular scenario. A principled approach to integrate these disparate sources of information will result in a general tracker that can be employed with success in business intelligence applications.

References

1. Davis, J. and Bobick, A., "The representation and recognition of action using temporal templates," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Juan, Puerto Rico (1997).
2. Karman, K. and von Brandt, A., "Moving object recognition using an adaptive background memory," in [*Time-varying Image Processing and Moving Object Recognition*], Capellini, ed., **II**, 297–307, Elsevier, Amsterdam, The Netherlands (1990).
3. Toyama, K., Krumm, J., Brumitt, B., and Meyers, B., "Wallflower: principles and practice of background maintenance," *Proc. 7th Intl. Conf. on Computer Vision,* Kerkyra, Greece (1999).
4. Wren, C., Azarbayejani, A., Darell, T., and Pentland, A., "Pfinder: real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19** (1997).
5. Gao, X., Boult, T., Coetzee, F., and Ramesh, V., "Error analysis of background adaption," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Hilton Head, SC (2000).
6. Stauffer, C. and Grimson, E., "Adaptive background mixture models for real-time tracking," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Fort Collins, CO (1999).
7. Tuzel, O., Porikli, F., and Meer, P., "A bayesian approach to background modeling," *IEEE Workshop on Machine Vision for Intelligent Vehicles (MVIV) in conjunction with CVPR* (2005).
8. Elgammal, A., Harwood, D., and Davis, L., "Non-parametric model for background subtraction," *Proc. European Conf. on Computer Vision,* Dublin, Ireland (2000).
9. Porikli, F. and Wren, C., "Change detection by frequency decomposition: wave-back," *Proc. of Workshop on Image Analysis for Multimedia Interactive Services,* Montreux (2005).
10. Stenger, B., Ramesh, V., Paragios, N., Coetzee, F., and Buhmann, J., "Topology free Hidden Markov Models: application to background modeling," *Proc. 8th Intl. Conf. on Computer Vision,* Vancouver, Canada (2001).
11. Wixson, L., "Detecting salient motion by accumulating directionary consistent flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22** (2000).
12. Mittal, A. and Paragios, N., "Motion-based background subtraction using adaptive kernel density estimation," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Washington, DC (2004).
13. Oliver, N., Rosario, B., and Pentland, A., "A Bayesian computer vision system for modeling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22** (2000).
14. Porikli, F., "Multiplicative background-foreground estimation under uncontrolled illumination using intrinsic images," (2005).
15. Zhao, C., Wang, X., and Cham, W.-K., "Background subtraction via robust dictionary learning," *EURASIP Journal on Image and Video Processing* (2011).
16. Freund, Y. and Schapire, R., "A decision-theoretic generalization of on-line learning and an application to boosting," *Annual Conference on Computational Learning Theory* (1995).
17. Boser, B., Guyon, I., and Vapnik, V., "A training algorithm for optimal margin classifiers," *Annual Conference on Computational Learning Theory* (1995).
18. Papageorgiou, C. and Poggio, T., "A trainable system for object detection," *International Journal of Computer Vision* **38** (2000).
19. Mohan, A., Papageorgiou, C., and Poggio, T., "Example-based object detection in images by components," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2001).
20. Dalal, N. and Triggs, B., "Histograms of oriented gradients for human detection," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA (2005).
21. Zhu, Q., Avidan, S., Ye, M., and Cheng, K.-T., "Fast human detection using a cascade of histograms of oriented gradients," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY (2006).
22. Porikli, F., "Integral Histogram: a fast way to extract histograms in Cartesian spaces," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA (2005).
23. Viola, P., Jones, M., and Snow, D., "Detecting pedestrians using patterns of motion and appearance," *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France (2003).
24. Tuzel, O., Porikli, F., and Meer, P., "Region Covariance: a fast descriptor for detection and classification," *Proc. European Conf. on Computer Vision,* Graz, Austria (2006).
25. Tuzel, O., Porikli, F., and Meer, P., "Human detection via classification on riemannian manifolds," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2008).
26. Felzenszwalb, P. and Huttenlocher, D., "Pictorial structures for object recognition," *International Journal of Computer Vision* **61** (2005).

27. Mikolajczyk, K., Leibe, B., and Schiele, B., "Multiple object class detection with a generative model," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY (2006).
28. Mikolajczyk, K., Schmid, C., and Zisserman, A., "Human detection based on a probabilistic assembly of robust part detectors," *Proc. European Conf. on Computer Vision,* Prague, Czech Republic (2004).
29. Leibe, B., Seemann, E., and Schiele, B., "Pedestrian detection in crowded scenes," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA (2005).
30. Gavrila, D. and Philomin, V., "Real-time object detection for smart vehicles," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Fort Collins, CO (1999).
31. Opelt, A., Pinz, A., and Zisserman, A., "Incremental learning of object detectors using a visual shape alphabet," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY (2006).
32. Blum, A. and Mitchell, T., "Combining labeled and unlabeled data with co-training," *Annual Conference on Computational Learning Theory* (1998).
33. Levin, A., Viola, P., and Freund, Y., "Unsupervised improvement of visual detectors using co-training," *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France (2003).
34. Kockelkorn, M., Luneburg, A., and Scheffer, T., "Using transduction and multi-view learning to answer emails," *European Conf. on Principle and Practice of Knowledge Discovery in Databases* (2003).
35. Zhu, Q., Avidan, S., and Cheng, K., "Learning a sparse, corner-based representation for time-varying background modeling," *Proc. 10th Intl. Conf. on Computer Vision,* Beijing, China (2005).
36. Comaniciu, D. and Meer, P., "Mean-Shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24** (2002).
37. Serby, D., Koller-Meier, S., and Gool, L. V., "Probabilistic object tracking using multiple features," *Proc. 17th Int'l Conf. on Pattern Recognition,* Cambridge, UK (2004).
38. Sethian, J., [*Level set methods: evolving interfaces in geometry, fluid mechanics computer vision and material sciences*], Cambridge University Press (1999).
39. Andriluka, M., Roth, R., and Schiele, B., "Pictorial structures revisited: people petection and articulated pose estimation," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Miami, F (2009).
40. Ross, D., Tarlow, D., and Zemel, R., "Learning articulated structure and motion," *International Journal of Computer Vision* (2010).
41. Zelnik-Manor, L. and Irani, M., "Event-based video analysis," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, HI (2001).
42. Rao, R., Yilmaz, A., and Shah, M., "View invariant representation and recognition of actions," *International Journal of Computer Vision* **50** (2002).
43. Lowe, D., "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision* **60** (2004).
44. Freeman, W. and Roth, M., "Orientation histograms for hand gesture recognition," *Intl. Workshop on Automatic Face and Gesture Recognition, Zurich, Switzerland* (1995).
45. Mikolajczyk, K. and Schmid, C., "A performance evaluation of local descriptors," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, WI (2003).
46. Sande, K., Gevers, T., and Snoek, C., "Evaluating color descriptors for object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32** (2010).
47. Belongie, S., Malik, J., and Puzicha, J., "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24** (2002).
48. Förstner, W. and Moonen, B., "A metric for covariance matrices," *Dept. of Geodesy and Geoinformatics, Stuttgart University* (1999).
49. Avidan, S., "Ensemble tracking," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Diego, CA (2005).
50. Edwards, G., Taylor, C., and Cootes, T., "Interpreting face images using active appearance models," *International Conference on Face and Gesture Recognition* (1998).
51. Paschos, G., "Perceptually uniform color spaces for color texture analysis: an empirical evaluation," *IEEE Trans. on Image Processing* **10** (2001).
52. Canny, J., "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8** (1986).
53. Horn, B. and Schunk, B., "Determining optical flow," *Artificial Intelligence* **17** (1981).
54. Lucas, B. and Kanade., T., "An iterative image registration technique with an application to stereo vision," *Intl. Joint Conf. on Artificial Intelligence* (1981).
55. Sun, D., Roth, S., and Black, M., "Secrets of optical flow estimation and their principles," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* San Francisco, CA (2010).
56. Mirmehdi, M., Xie, X., and Suri, J., eds., [*Handbook of Texture Analysis*], Imperial College Press (2008).
57. Harris, C. and Stephens, M., "A combined corner and edge detector," *Proc. Alvey Vision Conf.* (1988).
58. Laptev, I., "On space-time interest points," *International Journal of Computer Vision* **64** (2005).
59. Blum, A. and Langley, P., "Selection of relevant features and examples in machine learning," *Artificial Intelligence* (1997).

60. Schweitzer, H., Bell, J., and Wu, F., "Very fast template matching," *Proc. European Conf.. on Computer Vision,* Copehagen, Denmark (2002).
61. Comaniciu, D., Ramesh, V., and Meer, P., "Real-time tracking of non-rigid objects using Mean-Shift," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Hilton Head, SC (2000).
62. Comaniciu, D., Ramesh, V., and Meer, P., "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25** (2003).
63. Yilmaz, A., "Kernel based object tracking using asymmetric kernels with adaptive scale and orientation selection," *Machine Vision and Applications* **22** (2011).
64. Porikli, F. and Tuzel, O., "Multi-kernel object tracking," *Proceedings of IEEE Int'l. Conference on Multimedia and Expo,* Amsterdam, Netherlands (2005).
65. Porikli, F., Tuzel, O., and Meer, P., "Covariance tracking using model update based on Lie algebra," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* New York, NY (2006).
66. Porikli, F. and Tuzel, O., "Object tracking in low-frame-rate video," *Proc. of PIE/EI - Image and Video Communication and Processing,* San Jose, CA (2005).
67. Tuzel, O., Porikli, F., and Meer, P., "Learning on Lie groups for invariant detection and tracking," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Anchorage, AK (2008).
68. Porikli, F. and Pan, P., "Regressed importance sampling on manifolds for efficient object tracking," *6th IEEE Advanced Video and Signal based Surveillance Conference* (2009).
69. Shi, J. and Tomasi, C., "Good features to track," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Seattle, WA (1994).
70. Jepson, A., Fleet, D., and ElMaraghi, T., "Robust online appearance models for visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25** (2003).
71. Broida, T. and Chellappa, R., "Estimation of object motion parameters from noisy images," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8** (1986).
72. Rosales, R. and Sclarroff, S., "A framework for heading-guided recognition of human activity," *Computer Vision and Image Understanding* **91** (2003).
73. Tanizaki, H., "Non-Gaussian state-space modeling of nonstationary time series," *Journal of the American Statistical Association* (1987).
74. Bouaynaya, N., Qu, W., and Schonfeld, D., "An online motion-based particle filter for head tracking applications," *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* Philadelphia (2005).
75. Isard, M. and Blake, I., "Condensation: conditional density propagation for visual tracking," *International Journal of Computer Vision* **29** (1998).
76. Bar-Shalom, Y. and Foreman, T., [*Tracking and Data Association*], Academic Press Inc. (1988).
77. Rasmussen, C. and Hager, G., "Probabilistic data association methods for tracking complex visual objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23** (2001).
78. Streit, R. and Luginbuhl, T., "Maximum likelihood method for probabilistic multi-hypothesis tracking," *In Proceedings of SPIE* (1994).
79. Cox, I. and Hingorani, S., "An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **18** (1996).
80. Murty, K., "An algorithm for ranking all the assignments in order of increasing cost," *Operations Research* **16** (1968).
81. Hue, C., Cadre, J., and Perez, P., "Sequential Monte Carlo methods for multiple target tracking and data fusion," *IEEE Trans. on Signal Processing* **50** (2002).
82. Huttenlocher, D., Noh, J., and Rucklidge, W., "Tracking non-rigid objects in complex scenes," *Proc. 4th Intl. Conf. on Computer Vision,* Berlin, Germany (1993).
83. Terzopoulos, D. and Szeliski, R., "Tracking with kalman snakes," in [*Active Vision*], Blake, A. and Yuille, A., eds., MIT Press (1992).
84. MacCormick, J. and Blake, A., "Probabilistic exclusion and partitioned sampling for multiple object tracking," *International Journal of Computer Vision* **39** (2000).
85. Chen, Y., Rui, Y., and Huang, T., "JPDAF based HMM for real-time contour tracking," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, HI (2001).
86. Mansouri, A., "Region tracking via level set PDEs without motion computation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24** (2002).
87. Yilmaz, A., Li, X., and Shah, M., "Contour based object tracking with occlusion handling in video acquired using mobile cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26** (2004).