

Trajectory Guided Tracking and Recognition of Actions

Rómer Rosales and Stan Sclaroff
Computer Science Department
Boston University
111 Cummington St
Boston, MA 02215
e-mail: {rrosales,sclaroff}@cs.bu.edu

Abstract

A combined 2D, 3D approach is presented that allows for robust tracking of moving people and recognition of actions. It is assumed that the system observes multiple moving objects via a single, uncalibrated video camera. Low-level features are often insufficient for detection, segmentation, and tracking of non-rigid moving objects. Therefore, an improved mechanism is proposed that integrates low-level (image processing), mid-level (recursive 3D trajectory estimation), and high-level (action recognition) processes. A novel extended Kalman filter formulation is used in estimating the relative 3D motion trajectories up to a scale factor. The recursive estimation process provides a prediction and error measure that is exploited in higher-level stages of action recognition. Conversely, higher-level mechanisms provide feedback that allows the system to reliably segment and maintain the tracking of moving objects before, during, and after occlusion. The 3D trajectory, occlusion, and segmentation information are utilized in extracting stabilized views of the moving object that are then used as input to action recognition modules. Trajectory-guided recognition (TGR) is proposed as a new and efficient method for adaptive classification of action. The TGR approach is demonstrated using “motion history images” that are then recognized via a mixture-of-Gaussians classifier. The system was tested in recognizing various dynamic human outdoor activities: running, walking, roller blading, and cycling. Experiments with real and synthetic data sets are used to evaluate stability of the trajectory estimator with respect to noise.

1 Introduction

Tracking multiple objects in cluttered environments and analyzing their perceived motion, are central problems in computer vision. If instead of objects, people are tracked and their actions analyzed, then these problems are also crucial for visual surveillance. Unfortunately, these problems are inherently difficult; only in well controlled situations, normally not useful for common surveillance applications, have researchers been able to obtain satisfactory results. Effective solutions to these problems would lead not only to breakthroughs in video surveillance, but also human motion recognition, ergonomics, motion performance measurement, wireless computer human interfaces, virtual reality, computer animation, and robot navigation.

Low-level image processing methods have been shown to work surprisingly well in restricted domains despite the lack of high-level models [13, 14, 31, 51]. Unfortunately, most of these techniques assume a simplified version of the general problem; *e.g.*, there is only one moving object, objects do not occlude each other, or objects appear at a limited range of scales and orientations. While higher-level, model-based techniques can address some of these problems [10, 19, 27, 28, 37, 40, 44, 52], such methods are highly specific or typically require careful placement of the initial model.

These limitations arise because object tracking, trajectory estimation, and action recognition are treated as separable problems. In fact, these problems are inexorably intertwined. For instance, an object needs to be tracked if its trajectory is to be recovered; while at the same time, tracking can be improved if knowledge of the 3D motion trajectory is given. Similarly, to analyze the internal motion of an object, it is necessary to know what part of the scene it occupies, or how it moves (translates) in its environment; while at the same time, knowledge of the action gives clues to future motion, and can improve robustness of trajectory estimation and tracking. Therefore, our philosophy will be to exploit the interrelated nature of these three problems to gain greater robustness.

The goals of our unified framework are: 1.) to extend low-level techniques with the use of higher level feedback to handle multiple moving objects, 2.) to estimate and predict 3D motion trajectories, 3.) to explicitly model occlusion of multiple moving objects, and 4.) to recognize non-rigid motions of those objects being tracked. An improved feedback mechanism is proposed that combines low-level (image segmentation), mid-level (recursive trajectory estimation), and high-level (action recognition) techniques. The recursive trajectory estimation process provides a prediction and error measure that is exploited in higher-level stages of action recognition. Conversely, higher-level mechanisms provide feedback that allows the system to reliably segment and maintain the tracking of multiple moving objects before, during, and after occlusion. This approach enables accurate extraction of a stabilized coordinate frame for the moving non-rigid objects that is used in action recognition.

Our approach enables tracking and recognition of multiple actions as seen by a single video camera located in the same local area where the activities occur; *e.g.*, in a living room, work area, or on a street corner. This is in contrast to approaches that assume a top or very distant view of the scene. These views provide more substantial information for higher level mechanisms to be developed; *e.g.*, human motion analysis, understanding and recognition. The system has been tested in recognizing various dynamic human outdoor activities; *e.g.*, running, walking, roller blading, and cycling. Also, in order to provide a more objective performance measure, the noise stability properties of 3D trajectory recovery have been evaluated using synthetic data sets, and results are encouraging.

2 Related Work

In this section, a brief overview of related work in tracking, action recognition, and surveillance is given. The relationship of previous work directly related to our approach will be further elaborated later, as the

details of our formulation are introduced.

Our proposed formulation for 3D trajectory recovery utilizes recursive estimation theory, in particular the extended Kalman filter (EKF). The EKF approach has proven to be very useful in recovery of rigid motion and structure from image sequences [1, 11, 9, 39, 43, 46] and non-rigid motion [32, 37]. One of the first important results on recursive structure and motion estimation via an EKF was the work of [11]. The formulation of [1] yields improved stability and accuracy of the estimates. In both methods, image feature tracking and correspondence are assumed. In this paper, we present a method that automatically tracks multiple moving objects. This information is then used to estimate 3D translational trajectories (up to a scale factor).

Recently, a multiple hypothesis approach has been used in condensation tracking [5, 24, 25]. The formulation can account for multimodal state densities, which the Kalman Filter is not designed to handle. While we test our proposed approach in the context of unimodal prior distributions, it would be relatively straightforward to incorporate the condensation algorithm in our system. In our experimental evaluation, we have seen that a unimodal distribution provides an acceptable characterization of the system dynamics in our system.

In order to model trajectories, [9] assumed that the surface on which the motions occur is known, and also that this surface is a plane. Each object was represented as a point moving in the plane, partially avoiding problems related to changes in shape. It is also possible to reduce tracking to a plane, if the projective effect is avoided through the use of a top or distant view [22]. Some simple heuristics about body part relations and motion on the image like [21] can also be used, strongly limiting extensibility. In our work, we do not assume planar motion or detailed knowledge about the object and our formulation can handle some changes in shape that make tracking a harder problem. In our model, we assume locally linear trajectories.

More detailed representations use model-based techniques, generally articulated models comprised of 2D or 3D solid primitives [10, 19, 27, 28, 37, 40, 44, 52]. Such approaches can account for self-occlusion by relying on the object model. However, these models tend to be rather detailed, and therefore require high resolution imagery, multiple cameras, or well controlled environments that are not always available in surveillance applications.

Image contour models have also been used in human tracking and surveillance applications. The most relevant examples include the work by Baumberg *et al.*, [3, 2, 41], where human motion is constrained to walking/standing like configurations. The approach presented in this paper is more general with respect to the appearances that it can handle.

As for motion recognition, it can be divided into two categories: trajectory modeling and appearance change modeling. In trajectory modeling, typically the motion of object centroids is estimated and tracked over time. Trajectory modeling [20, 26, 33, 36] is very useful for surveillance based on trajectory patterns, but does not consider the non-rigid motion or shape changes of the moving objects. Such approaches cannot distinguish between totally different actions if they have similar trajectories (*e.g.*, biking *vs.* walking).

On the other hand, appearance change modeling methods [6, 13, 14, 38] look at the image motion of the object itself instead of static configurations or centroids. In particular, Davis and Bobick [14] used motion history images (MHI) and motion energy images (MEI); MHIs and MEIs are temporal image templates that are matched using a nearest neighbor approach against examples of given motions already learned. Another view-based representation, image skeletonization [17] can be used in a similar way. An important challenge with appearance change modeling methods is the need to normalize for gross changes in scale and 3D orientation and heading. Since the approaches are image-based, an estimate of the mapping between the input image and the appearance prototypes must be reliably estimated.

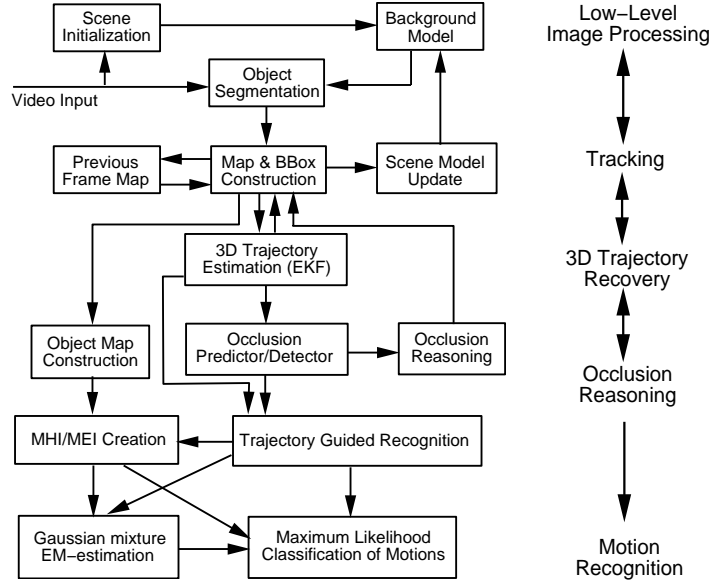


Figure 1: System Diagram.

In general, motion analysis techniques have had the problem that registration of useful, filtered information is a hard labor by itself. Methods for addressing this issue have been proposed previously in the seminal work of [3] and also [8, 14, 17, 35, 38, 53]. Our system provides this information by estimating the 3D trajectories of multiple moving objects, and modeling occlusion. This allows for normalization of incoming appearance with respect to gross changes in scale and 3D orientation and heading. Furthermore, the availability of object 3D trajectory information can be very useful for motion recognition. This results in a system that integrates the advantages of both trajectory modeling and appearance change modeling techniques, while addressing previous weaknesses.

3 Overview of Approach

A diagram of our approach is illustrated in Fig. 1. The system has both feed-forward and feed-back mechanisms. The lowest level of the algorithm is based on the scene modeling methods of [51], where the basic approach extends background subtraction methods. The system is initialized by acquiring statistical measurements of the empty scene over a number of video frames. The statistics acquired are the mean and covariance of image pixels in 3D color space. This is used as a confidence map (of pixel color) to guide segmentation of the foreground from the background.

Once initialized, moving objects in the scene are segmented using a log-likelihood measure between the incoming frames and the current background model. Frames taken from an example input of two people walking across a grass courtyard is shown in the top row of Fig. 2. The input video stream is low-pass filtered to reduce noise effects. The images are then fed into the log-likelihood foreground/background segmentation module. Connected components analysis, and morphological operations are also employed to improve segmentation quality. The resulting foreground segmentation for the example sequence is shown in the second row of Fig. 2.

If there are occlusions, or strong similarities between the empty scene model and the objects, then it is possible that regions are segmented erroneously. This is a known hard problem in computer vision. To alleviate this problem, some sources of temporal and higher level information are used. Three basic feedback criteria are used to gain improved segmentation at this stage: merging by motion differences,

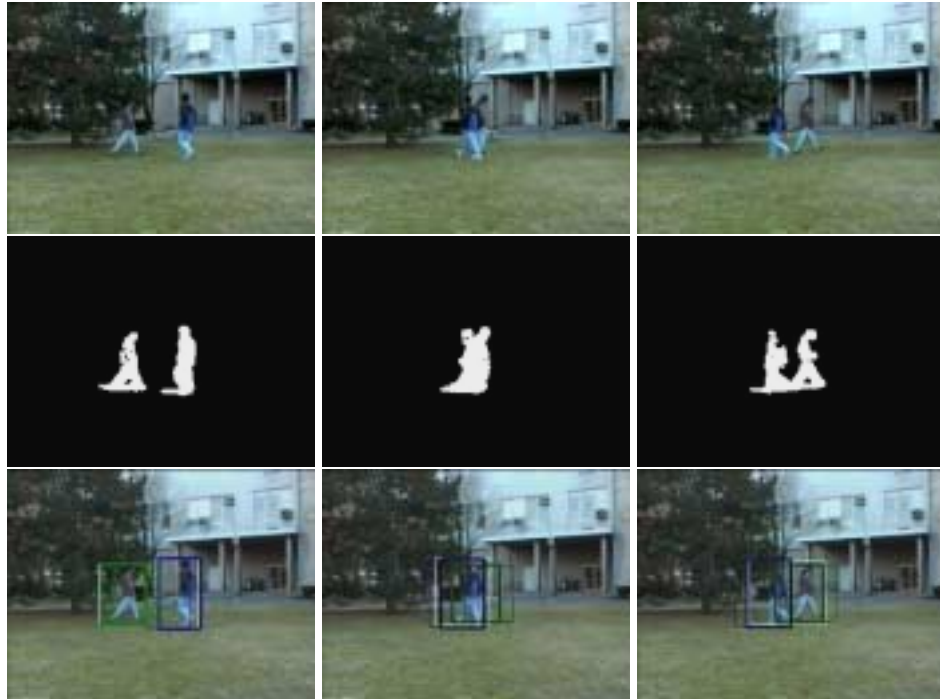


Figure 2: Tracking before, during, and after occlusion. The top row shows frames from the input sequence of two people walking across a grass courtyard. The next row shows the connected components extracted from the background. The bottom row shows the corresponding estimates of the bounding boxes for the moving objects. The bounding boxes are successfully predicted during the occlusion via an EKF.

spatial correlation analysis, and trajectory estimation. Each of these criteria will be described in detail in Sec. 4.2.

In trajectory prediction, an Extended Kalman Filter (EKF) provides an estimate of each object's image bounding box position and velocity. The input to the EKF is a 2D bounding box that encloses the moving object in the image. The extended Kalman filter then estimates the relative 3D motion trajectories for each object, based on a 3D linear trajectory model. In contrast to trajectory prediction based on a 2D model, the 3D approach is explicitly designed to handle nonlinear effects of perspective projection.

Occlusion prediction is performed based on the current EKF estimates. Given that we know the object position and the occupancy map, we can detect occlusions or collisions in the image plane. Our EKF formulation estimates the trajectory parameters for these objects assuming locally linear 3D motion, also the bounding box parameters are estimated. During an occlusion, the EKF can be used to give the maximum likelihood estimate of the current region covered by the object, along with its velocity and position.

Noisy observations and occlusions are commonplace; the EKF provides a natural way to deal with them. For example, if part of one object is not observed in a consecutive number of frames, an approximation of the predicted bounding box of the object is used based on previous tracking information. This is statistically maintained in the EKF state and error covariance matrix.

Occlusion prediction is performed using current 3D position and velocity estimates. During an occlusion or under no measurements, the EKF trajectory prediction can be used to give the maximum likelihood estimate of the current region covered by the object, along with its velocity, position, and



Figure 3: Example pairs of normalized views (support map and real view). The moving object is resampled in a canonical frame throughout the sequence, despite changes in scale and position.

their error covariances. However, during an occlusion, measurements are still used. Their purpose is to limit the output of the filter. Even though single object position and silhouette cannot be observed, their position in the image plane is limited by a shared measurement. The end of an occlusion event can also be detected. When this happens, our confidence in the observation changes our estimation process to the standard state (no occlusions). Every aspect is discussed in more detail in Secs. 5 and 6.

As shown in Fig. 3, a stabilized coordinate frame of the object is easily constructed with the information extracted from the sequence. The moving object is resampled in a canonical frame throughout the tracking sequence, despite changes in scale and position. The estimated bounding box is used to resize and resample the moving blob into a canonical view that can be used as input to motion recognition modules [38, 26, 42, 14, 45, 5], shape modeling [3, 12], or, if resolution allows, more detailed human tracking.

The resulting translation/scale stabilized images of the object are then fed to an action recognition module. Actions are represented in terms of motion energy images (MEI's) and motion history images (MHI's) [8, 14]. An MEI is a cumulative motion image, and an MHI is a function of the recency of the motion at every pixel. By using stabilized input sequences, it is possible to make the MEI/MHI approach invariant to unrestricted 3D translational motion. The stabilized representation is then fed to a moment-based action classifier. The action recognition module employs a mixture of Gaussian classifier, which is learned via the Expectation Maximization (EM).

In theory it is necessary to learn representations of every action for all possible trajectory directions. However, the complexity of such an exhaustive approach would be impractical. We therefore propose a formulation that avoids this complexity without decreasing recognition accuracy. The problem is made tractable via *trajectory-guided recognition* (TGR), and is a direct consequence of our tracking and 3D trajectory estimation mechanisms.

In TGR, we partition the sphere of possible trajectory directions based on the trajectories estimated in the training data. Each partition corresponds to a group of similar trajectory directions. During training and classification, trajectory direction information obtained via the EKF is used to determine the direction-partitioned feature space. This allows automatic learning and adaptation of the direction space to those directions that are commonly observed.

Human body tracking is a particular application of our formulation; however, human body structure and articulation characteristics may not be the best suited example for our assumptions. This is mainly due to its non-planar structure and large number of degrees of freedom. This creates an enormous set of configurations in general. But one advantage of not considering specific details about structure and constraints about human motion is that our approach can be applied to almost any object.

4 Looking for Multiple Moving Objects

In order to detect an object motion or location to track, an accurate registration technique is needed. The use of a fixed camera makes background-foreground segmentation methods well suited for this task [3, 4, 22, 20, 44, 47, 51]. If controlled with a feedback approach, such methods can achieve a

good level of accuracy [51, 46]. More importantly, this approach is efficient enough to allow real-time tracking. Finally, this technique also allows detection of foreground objects even when they are not moving. A moving camera can also be used, by building a background mosaic and estimating camera motion, or detecting independently moving objects [23, 34, 16]; however, such generality at the cost of a considerable increase in computation required.

Another possible approach is to measure the optical flow and use it to guide registration and/or segmentation. Unfortunately, non-rigid body tracking cannot easily benefit from optical flow computation. This mainly because flow estimation relies too heavily on the constant brightness assumption and it often requires solving the segmentation problem *a priori*. In addition, the resolution provided in non-laboratory environments (like that used here) is insufficient for accurate flow estimation. Finally, occlusions due to multiple moving objects further complicate this process.

4.1 Scene Initialization

Our initial low-level step is based on the results of [51], in which a Gaussian distribution for pixel color is assumed. We first build a representation of the background by obtaining the necessary statistics to fully characterize the color distribution of each pixel. This is done by capturing the environment without interesting objects (those we want to track). Practically, for the empty scene to be learned, it is necessary to capture about two seconds (approximately 60 frames) of video. The idea is to obtain a good estimate of color statistics for the background at each image pixel.

In our case, second order statistics are estimated. For a given pixel \mathbf{p} , we estimate its mean $\hat{\mu}_{\mathbf{p}}$ in 3D color space. The covariance for the color distribution of that pixel is then estimated using N frames:

$$\mathbf{K}_{\mathbf{p}} = \frac{1}{N-1} \sum_{t=1}^N (\mathbf{I}(\mathbf{p}, t) - \hat{\mu}_{\mathbf{p}})(\mathbf{I}(\mathbf{p}, t) - \hat{\mu}_{\mathbf{p}})^T, \quad (1)$$

where $\mathbf{I}(\mathbf{p}, t)$ is the image intensity at pixel \mathbf{p} at time t in color space. The initial scene model is given by the pair $(\hat{\mu}(0), K)$, and will be updated as described below.

Using this, we learn the model of our background. The information obtained in this process will make the segmentation more robust to sensor noise, and to changes in small areas of the scene that present some kind of movement. Examples of this are leaves that move due to the action of wind, some specularities, *etc.*

4.2 Segmentation

We will first describe the segmentation procedure used during regular conditions (no occlusions). Objects in the scene are characterized by a relatively large variation of the background statistics. Changes in the learned scene are computed at every frame by comparing current frames with the model. For this a distance measure is computed between the incoming image and the background using the log-likelihood measure:

$$\mathbf{d}_{\mathbf{p}}(t) = -(\mathbf{I}(\mathbf{p}, t) - \hat{\mu}_{\mathbf{p}}(t))^T \mathbf{K}_{\mathbf{p}}^{-1} (\mathbf{I}(\mathbf{p}, t) - \hat{\mu}_{\mathbf{p}}(t)) - \ln |\mathbf{K}_{\mathbf{p}}^{-1}| \quad (2)$$

This provides an estimate of the similarity with the background at every pixel. Then if a region of the image is dissimilar enough to the estimated empty region, the system considers it as an interesting region.

A binary map of the current frame is computed with the interesting regions in it. Then basic image morphological operations are applied (erosion-dilation) in order to rule out small regions and to fill holes in probable regions of interest. The binary map is represented by:

$$\mathbf{b}_i(t) = \bigcup_{\mathbf{p}} [\mathbf{d}_{\mathbf{p}}(t) > \Gamma] \quad (3)$$

The connected region $\mathbf{b}_i(t)$ at time t is defined by the locations where there was a difference from the model greater than a given threshold Γ . For this initial map, connected regions are labeled using a connected component analysis algorithm and their bounding boxes are computed. As opposed to previous approaches, our method includes the application of a function \mathbf{g} that consists of three additional steps or criteria for merging previously separate blobs $\mathbf{b}_i(t)$:

1. Motion difference merging criterion: the motion difference between the current and previous frame is computed for all pixels \mathbf{p} : $\mathbf{I}_d(\mathbf{p}, t) = \mathbf{I}(\mathbf{p}, t) - \mathbf{I}(\mathbf{p}, t - 1)$. Two blobs $\mathbf{b}_i(t)$ and $\mathbf{b}_j(t)$ are merged if there is a connected component from $\mathbf{I}_d(\mathbf{p}, t)$ that intersects both of them. The reason for this is that normally when $\mathbf{d}_p(t)$ is low (poor contrast), object's parts tend to form different blobs $\mathbf{b}_i(t)$. In our experiments we have found that using this criterion generally improves segmentation accuracy without oversegmenting the scene.
2. Spatial analysis criterion: a map of the previous segmented and processed frame is kept. This map is used as a possible approximation of the current connected elements. Connectivity in the current frame is compared with it, and regions are merged if they were part of the same object in previous frames. This can account for some shadow effects, local background/foreground similarities, and brief occlusions causing temporary loss of connectivity in an object. Basically, by assuming that object trajectories are continuous and that their position changes at a rate smaller than their own size in the direction of motion, a temporal matching is done. When occlusions are suspected or detected this process changes as will be described in Sec. 6.
3. 3D trajectory estimation criterion: higher level information obtained from the 3D trajectory estimate is used to predict the location of the object at next frame. A projection on the image plane of the estimated 3D position is used to merge all blobs $\mathbf{b}_i(t)$ falling in the area predicted to be occupied for a given object.

This process produces a labeling of possible objects in the image. Blobs are grouped to represent the different moving objects thought to be present in the scene. One possible drawback of this is the case when objects appear in the scene for the first time being too close to each other, or occluding each other. Because we do not use an explicit or hand-crafted model of the human body shape, these will be labeled as one single object. This situation may occur often, especially in crowded scenes. This is one of the major problems faced by segmentation/tracking algorithms. The problem is slightly different when objects' projections collide after already being present in the scene as separated objects. In this case our occlusion reasoning process will identify and correctly label them.

In conclusion, we denote the final segmentation as $\mathbf{s}_j(t) = \mathbf{g}(\mathbf{b}(t))$, where $\mathbf{g}(\bullet)$ uses all blobs found in our initial step. Our scene map at time t is defined $\mathbf{M}_t = \bigcup_j[\mathbf{s}_j(t)]$.

4.3 Tracker Units

A *tracker* unit T_i , is assigned to every object being tracked. The goal is to associate a given object with the same tracker unit during the whole time the object is in the camera's field of view. Tracker units are the final connection between regions and what the system believes is an object. Recall that we cannot simply associate one region $\mathbf{s}_j(t)$ with one object, nor can we expect that a region will always be tracked. The reason for this is that there are situations where a connected region $\mathbf{s}(t)$ is the projection of more than one object (*i.e.*, for example during occlusion). In other situations a blob may leave the scene definitively or may become occluded by a non-moving object.

For every time step t , a tracker unit contains information about object location, a binary support map $s_j(t)$, bounding box parameters, likelihood of further occlusion, and time it has been continuously tracked. At every time step we check for possible correspondence between every $s_k(t)$ and $s_j(t - 1)$ for $k = 1..N_t, j = 1..N_{t-1}$ (N_t is the total number of regions $s(t)$ at time t). The solution for this correspondence is used to update T_i so that the state information stored by T_i belongs to a unique object. T_i 's information will be used for: 1.) recovering trajectories (Sec. 5), 2.) detecting occlusions (Sec. 6), and 3.) aiding segmentation using $\mathbf{g}(\bullet)$. A summary of this process is given in Fig. 4.

A new tracker unit is assigned to every newly detected object. Special care needs to be taken with new objects in order to produce a stable tracker. This is because the trajectory estimate is not very reliable when the object first appears in the scene. There is a problem when objects appear within the boundary of other objects already present. In such cases the system may misclassify the objects as being the same. But in the general case, new objects can be detected easily by keeping a global state of objects being tracked. These states are maintained by our tracker instances T_i .

4.4 Updating the Model

After a binary map \mathbf{M}_t is obtained with all the possible objects in the scene, the system updates the statistical properties of the scene background. In this implementation only the mean is updated, this is done using a simple adaptive filter:

$$\hat{\mu}(t) = \alpha \mathbf{I}(\mathbf{p}, t - 1) \mathbf{M}_{t-1} + (1 - \alpha) \hat{\mu}(t - 1) \overline{\mathbf{M}}_{t-1}, \quad (4)$$

where $\overline{\mathbf{M}}(t)$ is the complement of $\mathbf{M}(t)$ at time t , α is a gain constant, and multiplication is performed in a pixel by pixel form. This is an exponentially weighted moving average. Note that only regions that are believed to belong to the background are updated. This can account for slow changes in lighting conditions in the scene, and will keep our scene model updated.

5 Estimating 3D Trajectories from 2D Image Motion

An novel extended Kalman filter (EKF) formulation is used to recursively predict the objects' future positions based on their current positions and velocities. The parameterization and selection of the state vector is in part related to the work of [1], who used it in a structure from motion problem. However, they tested their approach with rigid structures with the assumption that a sufficient number of feature points are tracked during the sequence. The problem of feature correspondence and tracking was not addressed directly. Furthermore, their method could not handle the appearance (or disappearance) of new features. Here we face the whole problem of obtaining observations by using them for estimating trajectories, and using trajectories to improve observation quality. Our approach is also related to the ideas of [9] in the use of an EKF to predict and recover object trajectories, but here we use a more general setting.

5.1 Features to Track

To reduce the complexity of the tracking problem, two feature points are selected: two opposite corners in the object bounding box. By using this we avoid the problem of searching for features and establishing correspondence between them in consecutive frames. In general we think that a detailed tracking of features is neither necessary nor easily tenable for non-rigid motion tracking. In fact, it may even be impossible without a detailed model of the object being tracked.

It is therefore assumed that although the object to be tracked is highly non-rigid, the 3D size of the object's bounding box will remain approximately the same, or at least vary smoothly. This assumption might be too strong in some cases; *e.g.*, if the internal motion of the object's parts cannot be roughly self

contained in a bounding box. However, when analyzing basic human locomotion, we believe that these assumptions are a fair approximation.

5.2 Camera Model

For our representation a 3D central projection model similar to [49, 1] is used:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \frac{1}{1 + z\beta}, \quad (5)$$

where (x, y, z) is the real 3D feature location in the camera reference frame, (u, v) is the projection of it to the camera plane, and $\beta = \frac{1}{f}$ is the inverse focal length. The origin of the coordinate system is fixed at the image plane. This model has proven to be useful when estimating focal length and structure in the structure from motion problem [1]. One important property of this model is that it is numerically well defined even in the case of orthographic projection.

5.3 3D Representation of Feature Points

In order to suit the needs of surveillance applications, we must introduce a novel EKF formulation. In particular, we will need a more convenient parameterization of the feature points. Feature point locations will be parameterized in terms of the object's x, y coordinates and its depth z coupled with β . Similarly, we use the vector $(\dot{x}, \dot{y}, z\dot{\beta})$ to represent a feature's 3D velocity relative to the camera reference frame.

We will also assume that the objects are undergoing a locally linear translational motion in 3D; this will allow us to extract relative 3D position of the object (or features) using several frames. This formulation only allows for recovery of depth up to a scale factor. For our tracking application, there is no need to recover the true 3D position of the feature point.

The noise sensitivity of a similar formulation has been analyzed in [1]. Without the effect of registration inaccuracies, the sensitivity in \dot{x} and \dot{y} is directly dependent on the object depth. This is because objects that are farther away from the camera tend to project to fewer image pixels. The sensitivity of \dot{z} depends on camera focal length. As focal length increases, sensitivity decreases. In the orthographic case this sensitivity is zero. In general, the sensitivity to \dot{x} and \dot{y} is higher than to \dot{z} .

5.4 Extended Kalman Filter Formulation

We will use a first order EKF. We also ran some experiments with a Discrete Kalman Filter; however, it became clear that the underlying process is inherently nonlinear and therefore an EKF is better suited. Our state models a planar rectangular box moving along a linear 3D trajectory at constant velocity. Note that this does not imply that the object must move on a line. Because we are considering the objects as being planar, the depth at both feature points should be the same. The reduction in the number of degrees of freedom improves the speed of convergence of the EKF. Our state vector then becomes:

$$\mathbf{x} = (x_0, y_0, x_1, y_1, z\beta, \dot{x}_0, \dot{y}_0, \dot{x}_1, \dot{y}_1, z\dot{\beta})^T, \quad (6)$$

where $(x_0, y_0, z\beta)^T, (x_1, y_1, z\beta)^T$ are the corners of the 3D planar bounding box. The vectors $(\dot{x}_0, \dot{y}_0, z\dot{\beta})^T$ and $(\dot{x}_1, \dot{y}_1, z\dot{\beta})^T$ represent each corner's 3D velocity relative to the camera. Note that the product $z\beta$ is common to the two features, and therefore so is their $z\dot{\beta}$ speed. This was conceived as a way to link the two spatial positions and increase stability by reducing the degrees of freedom of the system.

The 3D trajectory and velocity are recovered up to a scale factor. However, the family of allowable solutions all project to a unique solution on the image plane. We can therefore estimate objects' future positions on the image plane given their motion in $(x, y, z\beta)$ space.

The use of this 3D trajectory model offers significantly improved robustness over methods that employ a 2D image trajectory model. Due to perspective foreshortening effects, even when the trajectories in space are linear, trajectories in the image plane are nonlinear, and 2D models are therefore inaccurate. Note that if motion in space is linear, trajectories in the image plane can be linear if objects move parallel to the image plane, otherwise trajectories in image plane are almost always non-linear.

We now briefly explain the EKF formulation. The general formulation is a standard technique; we will provide the details of our formulation. Our process is guided by the following linear difference equation:

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{w}_k, \quad (7)$$

where \mathbf{x}_k is our state at time k , \mathbf{w}_k is the process noise and \mathbf{A}_k , the system evolution matrix, is based on first order Newtonian dynamics in 3D and assumed time invariant: ($\mathbf{A}_k = \mathbf{A}$). If additional prior information on dynamics is available, then \mathbf{A} can be changed to better describe the system evolution [43].

Our measurement vector is $\mathbf{z}_k = (u_{0k}, v_{0k}, u_{1k}, v_{1k})^T$, where u_{ik}, v_{ik} are the image plane coordinates for the observed feature i at time k . The measurement vector is related to the state vector via the measurement equation: $\mathbf{z}_k = h(\mathbf{x}_k + \mathbf{v}_k)$. Note that $h(\bullet)$ is non-linear. Measurement noise is assumed to be additive in our model. The EKF time update equation becomes:

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{A}_k \hat{\mathbf{x}}_k \quad (8)$$

$$\mathbf{P}_{k+1}^- = \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^T + \mathbf{W} \mathbf{Q}_k \mathbf{W}^T \quad (9)$$

where \mathbf{Q}_k is the process noise covariance. The matrix \mathbf{W} is the Jacobian of the transformation \mathbf{A} with respect to \mathbf{w} .

Because the measurement relationship to the process is nonlinear, at each step, the EKF linearizes around our current estimate using the measurement and state partial derivatives. The measurement update equations become:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{V} \mathbf{R}_k \mathbf{V}^T)^{-1} \quad (10)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-, 0)) \quad (11)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \quad (12)$$

where \mathbf{H}_k is the Jacobian of $h(\bullet)$ with respect to \mathbf{x} :

$$\mathbf{H}_k = \begin{bmatrix} \frac{1}{\lambda} & 0 & 0 & 0 & -\frac{x_0}{\lambda^2} \\ 0 & \frac{1}{\lambda} & 0 & 0 & -\frac{y_0}{\lambda^2} \\ 0 & 0 & \frac{1}{\lambda} & 0 & -\frac{x_1}{\lambda^2} \\ 0 & 0 & 0 & \frac{1}{\lambda} & -\frac{y_1}{\lambda^2} \end{bmatrix}, \quad (13)$$

where $\lambda = 1 + z\beta$. Finally, the matrix \mathbf{V} is the Jacobian of $h(\bullet)$ with respect to \mathbf{v} , and \mathbf{R}_k is the measurement noise covariance at time k . The general assumptions are: \mathbf{w} and \mathbf{v} are Gaussian random vectors with $p(\mathbf{w}_k) \sim N(0, \mathbf{W} \mathbf{Q}_k \mathbf{W}^T)$, and $p(v_k) \sim N(0, \mathbf{V} \mathbf{R}_k \mathbf{V}^T)$. For more detail, see [48, 50].

In general, as more measurements are collected, the error covariance of our estimates \mathbf{P}_k tends to decrease. Experimentally 40 frames were needed for convergence with real data. As will be seen in our experiments, motions that are not linear in 3D can also be tracked, but the estimate at the locations of sudden change in velocity or direction is more prone to instantaneous error. The speed of convergence when a change in trajectory occurs depends on the filter's expected noise. A resetting mechanism is used to detect when the EKF does not represent the true observations. This is done by comparing the current projection of the estimate with the observation. This can be achieved by transforming the state estimate into the observation space using $h(\bullet)$.

6 Occlusion Detection and Reasoning

The majority of tracking systems for non-rigid objects handle only either isolated objects or objects observed from view-points that minimize occlusions [9, 10, 19, 22, 26, 51]. Occlusions present a major problem in visual applications. Researchers have tried to address this problem by using multi-camera tracking, range, stereo, *etc.* When occlusions occur among multiple moving objects, their solution and detection adds complexity to the problem. Partial, temporary occlusions caused by a static scene element can be handled with the formulation of Secs. 4 and 5. In this section we propose a solution for the problem of occlusions among various simultaneously moving objects.

Our occlusion detection routine predicts future locations of objects, based on current estimates of 3D velocity and position. We use the system dynamics formulation of Sec. 5.4.

$$\mathbf{x}_{k+\Delta k} = \mathbf{A}(\Delta k)_k \mathbf{x}_k, \quad (14)$$

and a prediction in the observation is computed:

$$\mathbf{z}_{k+\Delta k} = h(\mathbf{x}_{k+\Delta k}). \quad (15)$$

In this way, it is possible to predict a collision of two or more objects in the image plane by comparing their estimated position and area covered. If a collision of the objects is expected in a given number of frames in the future, we assume that occlusion is probable (see Fig. 4). Our confidence in this probable collision for tracker unit T at time k is determined assuming an exponential distribution in the current estimate of error \mathbf{P}_k given by the EKF:

$$Occ(T, k) = \xi e^{-\xi Trace(\mathbf{P}_k)}, \quad (16)$$

where ξ is just set by experience based on the observed magnitude of values of \mathbf{P}_k . We experimentally found that a exponential distribution provides a good model for occlusion likelihood given our estimates of position, velocity, and error covariance. We use a simple threshold to tell when the *occlusion reasoning* mechanism is used. In this case the tracker units corresponding to the given objects are tagged for this potential event.

A false alarm can occur, so a recovery mechanism is used in which the system checks for reconfirmation of occlusion every time after occlusion was predicted. Occlusion is confirmed by using both the estimated bounding box parameters and the binary maps for the objects being tracked. The effect of an occlusion will be the merging of the blobs (binary maps) for the objects. After this occurs, the system uses the EKF prediction to update the object position and velocity. The blob $s_j(t)$, shared among occluded and occluding objects, provides a good way to constrain the possible location of any of these objects. Thus, $s_j(t)$ is used as observation for the tracker units interacting during the occlusion event.

While an occlusion is in progress, the system expects that merged blobs will eventually separate. Occlusion time can be estimated using the EKF estimates of 3D velocity and position along with the computed area of the blob. Furthermore, the division of merged blobs around the area estimated to be covered by the objects provides strong evidence for the end of an occlusion. Correspondence can be solved using object locations predicted through the 3D trajectory estimates after Δt time steps (the time extent of the occlusion). For clarity, Fig. 4 gives an overview of the tracking algorithm described in Secs. 4- 5.

7 Motion Recognition using TGR

Our tracking approach allows the construction of an object-centered representation. The resulting translation/scale stabilized images of the object are then fed to an action recognition module. Actions are

Parameters:

- Γ : segmentation threshold.
- Γ_{Occ} : occlusion probability threshold.
- Δt : time ahead (max) used to predict occlusions.

Iterate

Construct a statistical representation of the empty scene: $\mathcal{B}_0 = (\mu_0, \mathbf{K})$, using Eq. 1. We will represent the estimated empty scene at time t as \mathcal{B}_t . Then repeat for every time step t :

1. Use ML estimator of Eq. 2 to compute a log-likelihood map $\mathbf{d}_p(t) = (\mathcal{B}_t, \mathbf{I}(\mathbf{p}, t))$.
2. Segment moving objects using $\mathbf{d}_p(t) > \Gamma$, where Γ is the segmentation threshold. Represent the set of all connected component units as $\mathbf{b}(t)$.
3. Compute final segmentation: $\mathbf{s}_j(t) = \mathbf{g}(\mathbf{b}(t))$, $j = 1..N$ where N is the total number of objects in the scene, $\mathbf{g}(\bullet)$ uses all blobs found at $t - 1$. Our scene map at time t is $\mathbf{M}_t = \bigcup_j [\mathbf{s}_j(t)]$.
4. For all segmented objects (for all j), use $\mathbf{s}_j(t)$ to compute their bounding boxes, and obtain observations $\mathbf{z}_j(t) = (u_{0t}, v_{0t}, u_{1t}, v_{1t})$ taken to be two opposite corners.
5. For every segmented object $\mathbf{s}_j(t)$, and every tracker unit T_i .
 - (a) If $Occ(T_i, t) < \Gamma_{Occ}$ (no occlusion predicted)
 - i. Project state \mathbf{x} of object to the image plane using Eq. 7.
 - ii. Find closest $\mathbf{s}_j(t)$, using a L_2 norm on the pair of corners of bounding boxes, and use it as observation \mathbf{z}_t .
 - iii. Use Eqs: 8-12 to obtain $\hat{\mathbf{x}}_{t+1}^-$, $\hat{\mathbf{P}}_t$, and $\hat{\mathbf{x}}_t$, and update global EKF state.
 - iv. Determine probability of occlusion of object i , ($Occ(T_j, t+1)$) at time $t+1$ using $\hat{\mathbf{x}}_{t+\Delta t}$, and $\hat{\mathbf{x}}_{t+\Delta t}$ with chosen parameter Δt .
 - (b) else ($Occ(T_i, t) > \Gamma_{Occ}$)
 - i. Find closest connected element using Eq.7. Allow for possibly shared connected elements (*i.e.*, blobs that belong to two different objects).
 - ii. Use Eqs: 8-12, with possibly shared observed bounding box as observation \mathbf{z}_t , and $\mathbf{R} = \lambda \mathbf{R}$ to obtain $\hat{\mathbf{x}}_{t+1}^-$, $\hat{\mathbf{P}}_t$, and $\hat{\mathbf{x}}_t$, update global EKF state, where λ is a scalar that increases the error covariance \mathbf{R} at the time of occlusions.
6. Update scene representation: $\hat{\mu}(t) = \alpha \mathbf{I}(\mathbf{p}, t-1) \mathbf{M}_{t-1} + (1 - \alpha) \hat{\mu}(t-1) \overline{\mathbf{M}}_{t-1}$.

Figure 4: Tracking Algorithm.

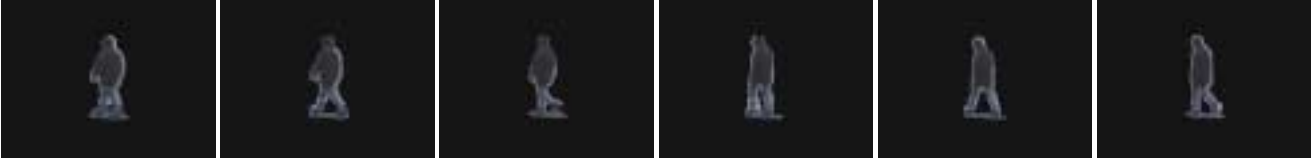


Figure 5: Example frames of output of the new processing step applied to one subject in one of our reconstructed walking sequences.

represented in terms of motion energy images (MEI's) and motion history images (MHI's) [8, 14]. An MEI is a cumulative motion image, and an MHI is a function of the recency of the motion at every pixel. By using stabilized input sequences, it is possible to make the MEI/MHI approach invariant to unrestricted 3D translational motion.

7.1 Motion Segmentation

Motion segmentation is required to center attention on the regions of interest only (*i.e.*, on each moving object). Let T_i and particularly $s_j(t)$, as defined in Sec. 4 (with $i = 1..N_t$ objects segmented at time t), and $\mathbf{I}(\mathbf{p}, t)$ the color input frame at time step t (for all pixels \mathbf{p}). Recall that T_i 's region of support $s(t)$ indicates the spatial location of the object i in frame $\mathbf{I}(\mathbf{p}, t)$. Using this information, we can construct an image sequence where the given object is the only one present in it, and all background pixels have been removed:

$$\mathbf{I}_i(\mathbf{p}, t) = s(t)\mathbf{I}(\mathbf{p}, t), \quad (17)$$

for each T_i . We use $\mathbf{I}_i(\mathbf{p}, t)$ as the new input (coming from object i at time t) to construct the view-based representations of motion. An example of frames from a sequence reconstructed by this basic process is shown in Fig. 5.

7.2 Motion Parameterization

A way to parameterize the high dimensional representation provided by each instance of the pairs of MHI's and MEI's is needed. The seven Hu moment invariants are computed for both the MHI and MEI [8, 14]. We denote them \mathbf{x}_{MHI} and \mathbf{x}_{MEI} respectively. The resulting features are combined in a 14-dimensional vector: $\mathbf{x}_f = (\mathbf{x}_{\text{MEI}}, \mathbf{x}_{\text{MHI}})$. The dimension of \mathbf{x}_f is reduced via principal components analysis (PCA) [18]. The main motivation for this reduction is that this creates a more compact representation that improves distribution estimation accuracy, and statistically reduces empty-space problems [45]. The reduction is achieved by solving the well known eigenvalue decomposition problem:

$$\Lambda = \Phi^T \Sigma \Phi, \quad (18)$$

where Φ is the eigenvector matrix of the covariance of the data (Σ) and Λ is the corresponding diagonal matrix of eigenvalues. Only M eigenvectors are kept corresponding to the M largest eigenvalues, obtaining the matrix Φ_R . Now we define our new feature vector

$$\phi = \Phi_R^t \mathbf{x}_f. \quad (19)$$

In our experiments, the PCA allowed a dimensionality reduction of 64% ($M=5$), while capturing 90% of the variance of our training data, and more importantly, the accuracy of the estimated data distributions improved considerably (Sec. 7.3). The reduced feature vector ϕ is then fed into a maximum likelihood classifier that is based on a mixture of Gaussians model.

Training is performed using $\mathbf{I}_i(\mathbf{p}, t)$ and indicating the beginning and end of each action (*e.g.*, one full walking repetition). One MHI and one MEI of the given image motion produced by the object is computed in this fashion and then labeled. This process is repeated for all object centered sequences \mathbf{I}_i in the training set. For recognition, $\mathbf{I}_i(\mathbf{p}, t)$ is used to compute the MHI and MEI at time t using a fixed time window. Note that multiple time windows may allow a search for a matching motion at many temporal scales [14], but for simplicity, in our experiments we use a fixed time window.

7.3 Probabilistic Learning

Each action class i is represented by a set of mixture model parameters, Θ_i . We have performed experiments where different models were tested, and the reasons for choosing each of them is analyzed in [45]. The model parameters and prior distribution, $P(\phi|\Theta_i)$ for each action class are estimated during a training phase using the expectation maximization (EM) algorithm [15].

For a mixture of Gaussians, class parameters are $\Theta = (\mu, \Sigma, \Lambda)$. $\mu = (\mu_1 \dots \mu_m)$ is the set of mean vectors, $\Sigma = (\Sigma_1 \dots \Sigma_m)$ is the set of covariance matrices, in our implementation, we use a diagonal $\Sigma_k = (\sigma_{ij})_k$ with $i, j = 1 \dots M$ (M was chosen to be 5 from PCA results), $\Lambda = (\lambda_1, \dots, \lambda_m)^T$ is the vector of weights of the Gaussian modes.

In estimating the parameters for a Gaussian mixture distribution, we could think of the mixture modes as being unobserved. The EM algorithm is based on increasing the expected likelihood of the complete data given the observed data. In our case this can be represented as follows (for each class):

$$Q(\Theta|\Theta^{(P)}) = \sum_{i=1}^n \sum_{k=1}^m z_{ik} (\log \lambda_k - \log 2\pi - \log(\prod_{l=1}^d \sigma_{ll}) - \frac{1}{2} \sum_{l=1}^d (y_{il} - \mu_{ki})^2 \sigma_{ll}^2) \quad (20)$$

where $z_{ik} = p(Z_i = k|y_i, \Theta^{(P)})$ and Z_i is an indicator of the mixture model. In the EM equations, n and m are the number of observations and the number of Gaussians used in the mixture respectively.

The vector of parameters that define the given distribution is represented with Θ . For the E-step we find the expected likelihood of the complete data as a function of Θ . It basically reduces to finding z_{ik} on the E-step [15]:

$$z_{ik} = \frac{P(\phi_i|Z_i = k, \Theta^{(P)})P(Z_i = k|\Theta^{(P)})}{\sum_{j=1}^m P(Z_i = j|\Theta^{(P)})P(\phi_i|Z_i = j, \Theta^{(P)})}. \quad (21)$$

The M-step re-estimates the parameters such that $Q(\Theta|\Theta^{(P)})$ is maximized: $\Theta^{(P+1)} = \arg \max_{\Theta} Q(\Theta|\Theta^{(P)})$. Due to space limitations, we omit the detailed derivation and give the final solution for our re-estimation step (M-Step):

$$\mu_k^{(P+1)} = \frac{\sum_{i=1}^n z_{ik} \phi_i}{\sum_{i=1}^n z_{ik}} \quad (22)$$

$$\sigma_{kll}^{2(P+1)} = \frac{\sum_{i=1}^n z_{ik} (\phi_{il} - \mu_{kl})^2}{\sum_{i=1}^n z_{ik}} \quad (23)$$

$$\lambda_k^{(P+1)} = \frac{\sum_{i=1}^n z_{ik}}{\sum_{l=1}^m \sum_{i=1}^n z_{il}} \quad (24)$$

The motivation for using a mixture model is that there may be a number of variations (or body configurations) for motions within the same action class. If necessary, a mixture model is able to associate a mode with each of those motions that are labeled the same but are performed relatively differently. The model should adequately span the space of standard configurations for a given action. However, we are

only interested in finding a representative set of these modes. We therefore use an information theoretic technique to select the *best* number of parameters to be used via MDL principle:

$$MDL : \arg \max_{\Theta_{i,k}} (\log P(\phi|\Theta_i) - \frac{k}{2} \log n), \quad (25)$$

where k is the number of parameters in the model (*i.e.*, length of each Θ_i in our case), and n is the number of samples in the training data. Further details about model estimation and reasons for the choice of the given underlying distribution can be found in [45].

7.4 Trajectory-Guided Recognition

We are now faced with this problem: given an instance of ϕ , how do we classify it as one of the learned actions Θ_j ? A theoretical solution for this is: given $P(\phi|\Theta_i)$, calculate $P(\Theta_i|\phi)$ using Bayes rule, and then find the most likely class among the set of all classes, given the measurement ϕ :

$$j = \arg \max_i P(\Theta_i|\phi). \quad (26)$$

An important problem in action recognition using view-based representations is that in order for an action to be recognized, the action has to occur in the same view-angle as the actions used for learning. In general, even non view-based methods cannot cope with the problem of recognition without a prior estimation of the 3D alignment of the action. This is generally time-consuming and its complexity increases dramatically with the number of degrees of freedom used to represent the motion.

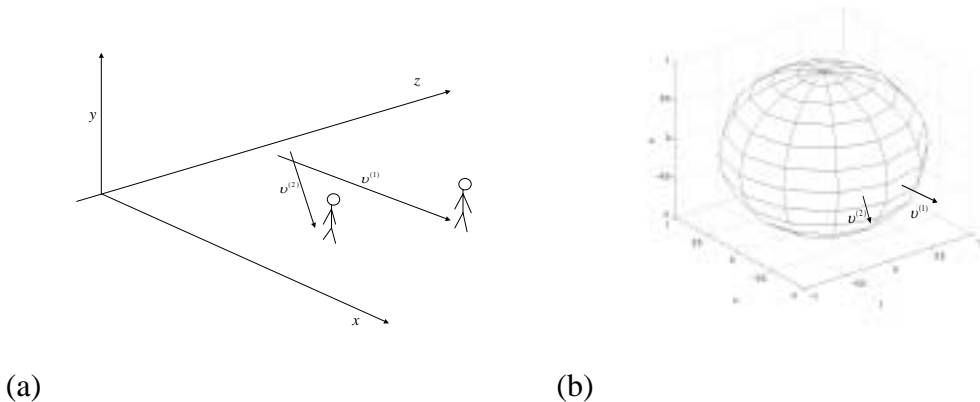


Figure 6: Example tessellation and trajectory matching as one step in TGR. Action being performed by each subject is mapped to the respective regions in the sphere of action PDF's.

We propose to solve this view dependence problem by using a trajectory estimation approach. In theory it is necessary to learn representations of every action from all possible directions. Let \mathcal{P}_j to be the set of PDF's used to represent m actions performed by the object while moving under a given direction j . Each action i has its own PDF, $P(\Theta_i^{(j)}|\phi) \in \mathcal{P}_j$ ($i = 1..m$). Building such a representation would require incredible amounts of training data acquired from multiple viewpoints. For motion classification, an exhaustive search through the whole space of views to find the best match would be needed. We propose a formulation that avoids this complexity without decreasing recognition accuracy.

The problem can be made tractable via a new approach: *trajectory-guided recognition* (TGR). TGR is made possible as a direct consequence of our tracking and 3D trajectory estimation mechanisms. In

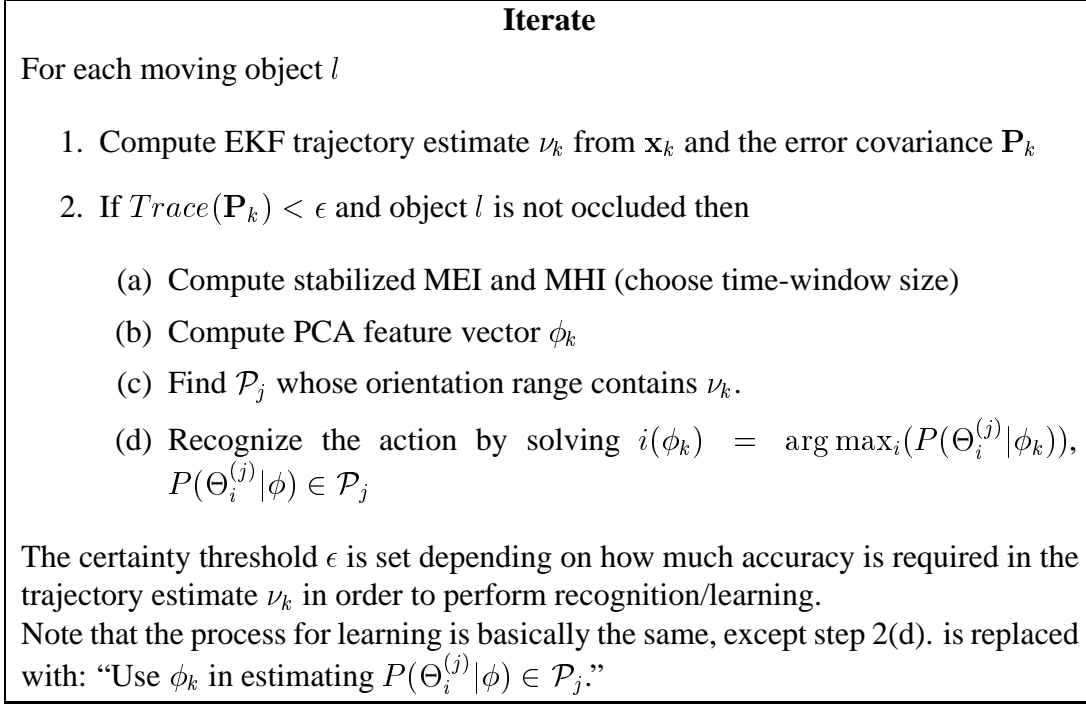


Figure 7: Overview of the trajectory-guided recognition algorithm.

TGR, we partition the direction sphere based on the trajectories estimated in the training data. Recall that trajectories are estimated by tracking the object of interest. Each partition j in the tessellated sphere corresponds to a group of similar trajectory directions. The PDF's can then be automatically estimated for each bin in the trajectory direction space.

Fig. 6 graphically presents this idea. At a given instant subject 1, whose state vector is $\mathbf{x}^{(1)}$, is moving in direction $\nu^{(1)}$. In general the error covariance of the two bounding box corners is similar, therefore we can then simply use $\nu^{(1)} = \frac{1}{2}(\dot{x}_0 + \dot{x}_1, \dot{y}_0 + \dot{y}_1, 2z\dot{\beta})$. Similarly, subject 2 has a state $\mathbf{x}^{(2)}$ and average direction $\nu^{(2)}$. These estimates are obtained during tracking. If the system wants to learn the action being performed by subject 1 at the given instant, then this action will be automatically mapped to the set \mathcal{P}_j , where $\nu^{(1)}$ is in the range of trajectory directions defined by j . This is shown in the tessellated sphere in 6(b), where every region symbolizes a range of trajectory directions. Note that the partition of the sphere surface does not necessarily have to be done as illustrated on the sphere, for example an adaptive mesh could be used [30].

Therefore, we only need to use a single camera viewpoint and sample the space based on the estimates of trajectories. In this way we can recognize the motion given its trajectory.

Statistically, the more examples of an action we observe in a given trajectory direction, the better that action class will be described for that direction. In a specific surveillance environment, by observing the scene and indicating the action that is occurring, the system can automatically estimate what direction is associated with the indicated action, and therefore use the right set of PDF's \mathcal{P}_j .

As a consequence, trajectory bins that are rarely observed (or not observed at all) will not be likely to have a statistically reliable representation. The same argument is applicable to actions not observed under the given directions. If we assume that the training data is representative of the environment characteristics, then observing only a few samples in a given region during training may indicate that in the chosen surveillance environment, those trajectories rarely occur. A summary of the TGR algorithm

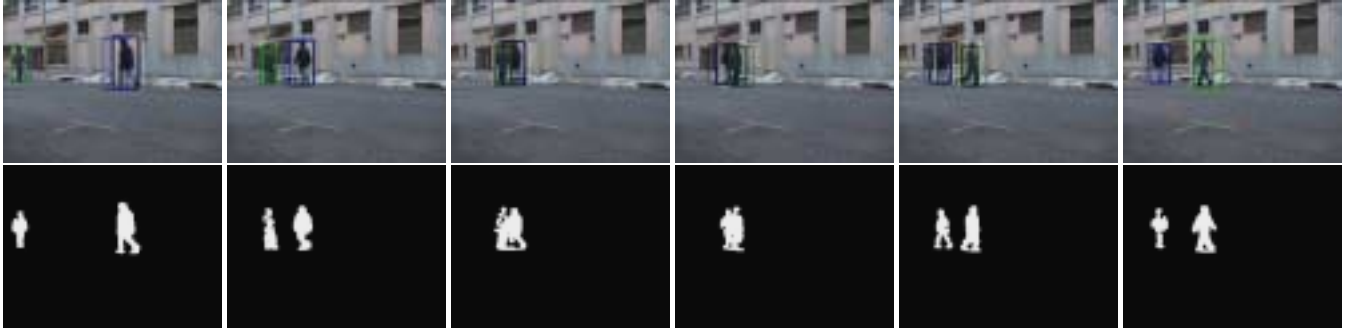


Figure 8: Tracking example: input sequence, and blob segmentation results.

is provided in Fig. 7.

This approach has important advantages. In many practical situations, not all trajectories are observed, so the 3D space can be divided into fewer areas (*e.g.*, see Sec.8.2). Learning can be accomplished without the need for examples of the same actions collected from many different camera orientations. During action recognition, an object's estimated 3D object trajectory can be used to find directly the appropriate PDF, instead of exhaustively searching over all possible directions and all possible actions. Also *a priori* distributions of actions from given directions can be learned. An action in a given direction may be more likely than another action in the same direction. Finally, it is possible to adapt the partitioning of the direction space based on availability of data. A finer or coarser tessellation of regions in the view-sphere can be constructed adaptively based on the distribution of the data (*e.g.*, [30]).

An extension allowed by TGR, currently being examined, is the possibility to infer how actions would look if the trajectory were slightly different. In other words it may be possible that given $P(\Theta_i^{(j)}|\phi) \in \mathcal{P}_j$, we can infer $P(\Theta_i^{(l)}|\phi) \in \mathcal{P}_l$, for adjacent sphere bins j and l . This could be accomplished by approximating the 3D projective effect on the motion representation under the slightly different views. This could be very useful in reducing the amount of data needed for every trajectory direction, or even to generate action representations under completely unobserved views.

8 Experimental Results

The system was implemented and experiments were conducted on a SGI O2 R5K workstation. For all experiments we used either a consumer hand-held video camera, or the standard SGI O2 uncalibrated camera recording at 30 frames/sec (320x240 pixels, color). Sequences undergo a loss in resolution due to MJPEG encoding.

8.1 Using 3D Trajectory Estimation to Track

In order to test the 3D trajectory estimation approach with real data, and its use in tracking multiple people, we first collected twenty sequences of people walking and occluding each other in outdoor environments. We show some representative examples. As a way to emphasize the robustness to parameter setting, all experiments use exactly the same system parameter settings. The scene learning phase took 1.5 sec in each example (45 frames).

Our first example, in Fig. 8, consists of a ten second multiple body walking sequence. It shows the standard behavior of the system when motions are roughly on a linear path. Note that trajectories are not parallel to the camera plane. This forms non-linear trajectories in the image. There are two color-coded boxes and one white box surrounding every tracked object. The box with thicker lines corresponds to the object estimated position. The box with thinner lines gives an estimate of the position $\frac{1}{3}$ sec. ahead

based on the current estimate of the velocity, using a first order model. The white box is the measurement obtained from the low-level processes. During the whole sequence, people were tracked and segmented accurately. Occlusion was predicted, detected and handled properly by estimating positions and velocities followed by projecting these estimates to the image plane.

Fig. 9 shows the normalized coordinate frames extracted. The moving object is resampled in a canonical frame throughout the tracking sequence, despite changes in scale and position. The estimated bounding box is used to resample the moving blob to a canonical view that can be used as input to motion recognition modules.

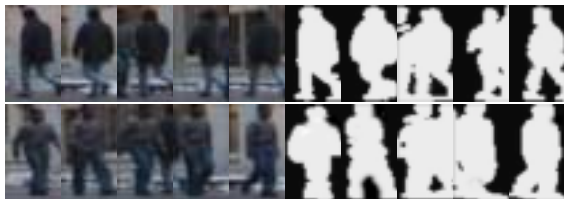


Figure 9: Normalized views of the 2 bodies in the sequence, one row per body. 6 views with their respective regions of support.

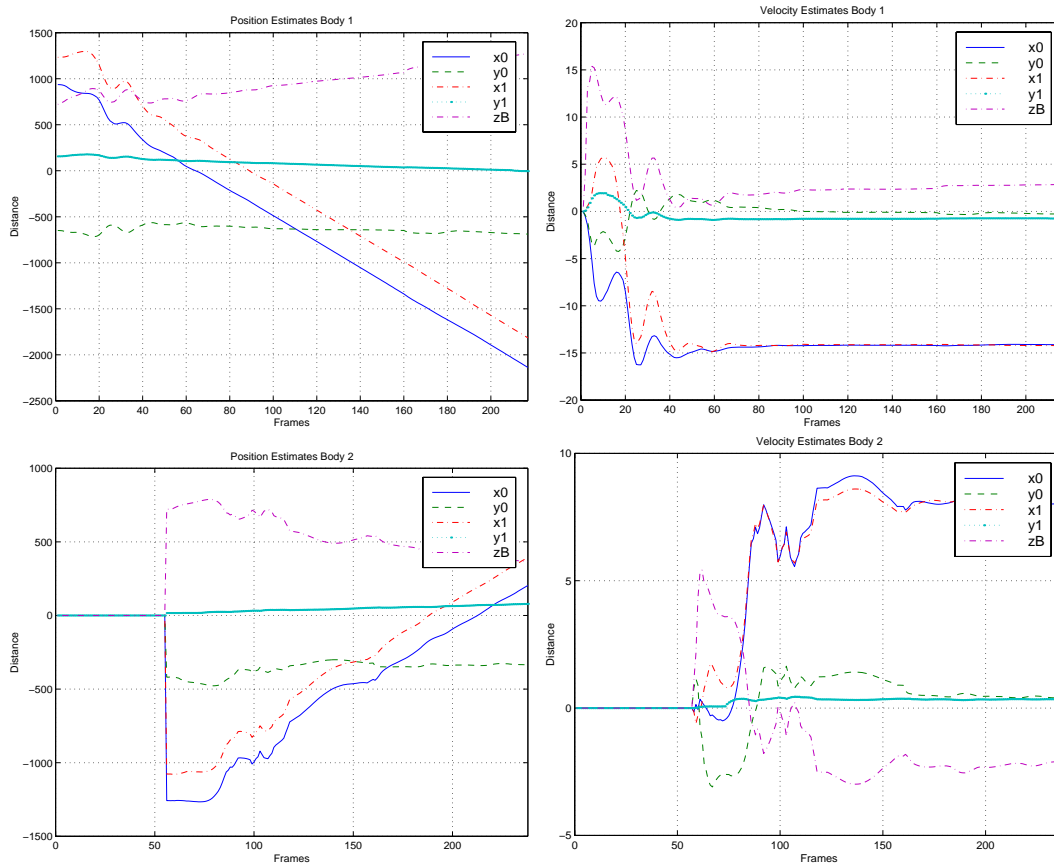


Figure 10: Estimated positions and velocities in 3D given by the EKF. The top row shows position and velocity predictions for body 1. The bottom row shows predictions for body 2. Note that body 2 appears in scene about 55 frames after body 1.

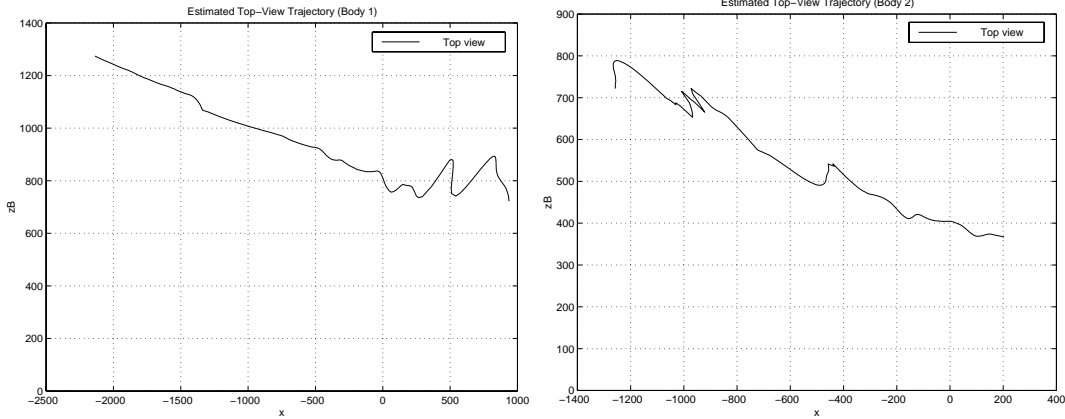


Figure 11: Recovered (top view) motion. Note that motion of body 1 is from right to left. Body 2 goes in the opposite direction. Estimates at the beginning of the sequence are not very accurate, because error covariance in the EKF is higher. Uncertainty is reduced as enough samples are given.

Fig. 10 shows the evolution in the estimates of 3D velocity and position computed via our EKF. Body 2 entered the scene about 55 frames after body 1. The estimates visually agree with the motions shown in the sequence. Note that we are estimating $z\beta$ not z . Scale in depth estimates is not necessarily the same as in x, y estimates.

Finally, given the information recovered, it is possible to construct a top-view map, showing the trajectories of the bodies Fig. 11. The first body moves from right to left, while the second body moves in the opposite direction. While the early position estimates are noisy, after 20-40 frames the EKF converges to a stable estimate of the objects' trajectories.

Fig. 12, shows frames and tracking results from another outdoor sequence. Here resolution of bodies is smaller and the difference in trajectory, depth and velocity is more noticeable. Again the system can track and recover positions and velocities. Fig. 13 shows the warped normalized views, and Fig. 14 shows the estimates of 3D trajectories.



Figure 12: Example tracking results, during occlusion the system gives a good estimate of position of the bodies, and solves for correspondence.

In this first set of experiments, the system was tested on the 20 collected sequences, with motions varying in depth, direction, and velocity. In the experiments, the system successfully tracked and recovered positions and velocities despite motion towards the camera, and/or occlusions. In general, the model works well, as long as there is not a sudden change in direction during the time an occlusion is taking place. The next example illustrates this behavior.

Our next example shows a sequence where, during occlusion, one of the subjects suddenly changes his direction. If there were no occlusion, the EKF would adapt its estimate to the new direction. But because the change in direction occurs during occlusion, the EKF estimate will be in error and could obviously lose the track, as seen in the estimates shown during and after occlusion in Fig. 15. Recall



Figure 13: Normalized views of the 2 bodies in the sequence, one row per body. 6 views with their respective regions of support.

that the prediction of the position $\frac{1}{3}$ seconds ahead is represented with the thin box. In the fourth and fifth image from Fig. 15, we can see the wrong prediction for the subject moving from left to right. Our model assumed that the object continued its observed trajectory.

Speed of re-convergence depends on the current error covariance, the expected measurement noise, and the model noise. In this example even though there is a sudden change in direction, the tracker can in some cases recover by solving for correspondence directly. This is a result of the characteristics of the situation, but it is useful to illustrate how in some cases it is possible to recover from inaccuracies in the estimates during occlusion.

In cases where trajectories change considerably during occlusions, the system will fail if the correspondence function is not correct. A simple situation is when during occlusion, the subjects decide to go back (in the opposite direction initially taken). We are interested in the use of trajectory information to track. To solve this problem, trajectory information is not enough. Object color or shape could be used to disambiguate these instances. However, our interests are in the use of trajectory information for tracking.

8.2 Learning and Recognition of Actions

In order to test the full action recognition system, we collected sequences (three hours total) of random people performing different actions on a pedestrian road in an outdoor environment (Charles River sequences). We trained the system to recognize four different actions (walking, running, roller blading, biking) gathered from two different camera viewpoints. The camera was located 45° and -45° angle with respect to the road. Note that, as explained above, only one view is needed. Data from different view points was initially taken to perform two different tests. However, this data was combined as explained next.

Video sequences showing 56 examples of each action were extracted from this data set. The duration of each example ranged from one to five seconds (30-150 frames). Because of the geometric properties of the view angles used, the 56 examples were obtained by merging the samples taken from both views. The data obtained from the 45° view angle was kept the same, but the data obtained from the -45° view angle was reflected.

Example frames from the data set are shown in Fig. 16. As before, the estimated bounding boxes for each moving object are shown overlaid on the input video image. Note that a simplifying issue in these experiments is the existence of a foot path. This reduces the number of total examples needed to test our approach. Recall that each class needs a minimum number of examples to be statistically characterized. By concentrating the data into fewer trajectory directions, we reduce the total number of examples needed.

Fig. 17 shows the distribution of observed trajectory directions for the walking action. Fig. 17(a) shows the frontal hemisphere with respect to the camera view point, Fig. 17(b) shows the back hemisphere, Fig. 17(c) shows a top view of the sphere. Only trajectories with error covariance smaller than a

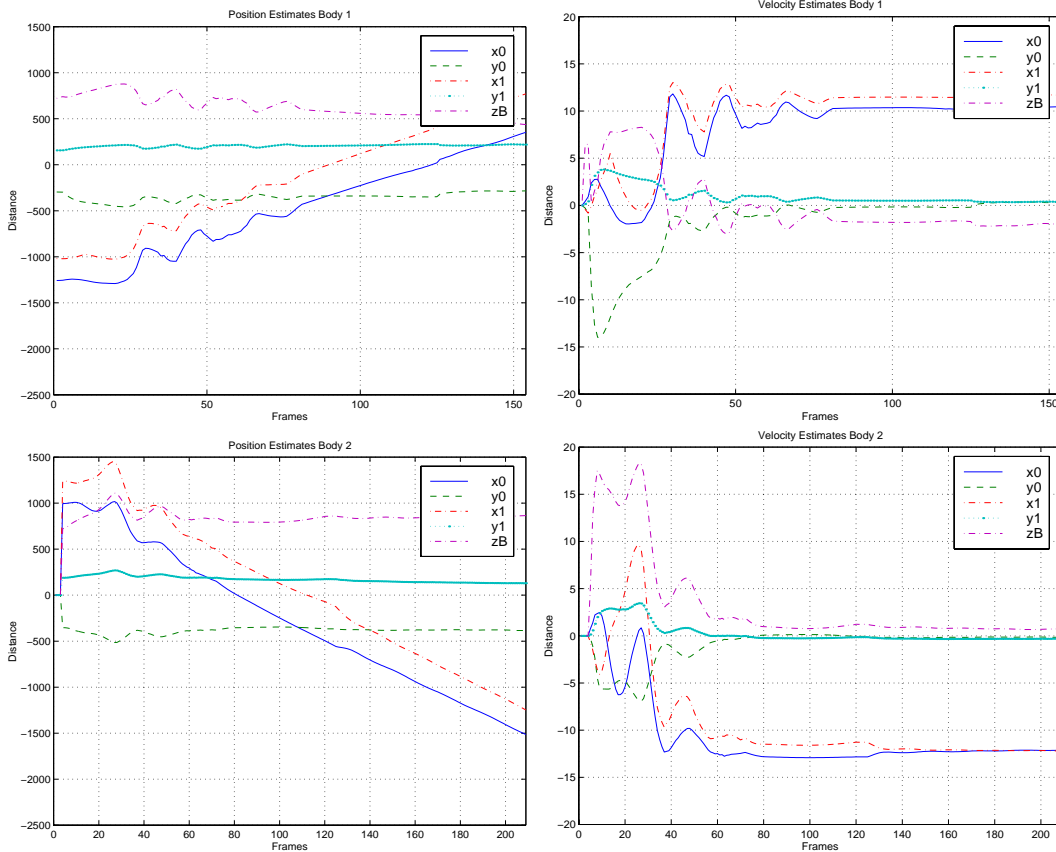


Figure 14: Example 2, estimated positions and velocities in 3D. Estimates visually agree with the trajectories seen in the scene.

given threshold are shown. From these, only bins whose amount of data allowed class estimation were considered. The data points in these bins are represented with blue circles, while data points in other bins are represented with red cross-marks.

Our approach indicated that only two trajectories were mainly observed: either direction along the foot path, on an horizontal plane. Therefore, the system learned just two sets of $m = 4$ PDF's: $(\mathcal{P})_1$ and $(\mathcal{P})_2$ (recall that there are four classes and two major trajectory directions).

The recognition experiment was conducted as follows. For each trial, a subset of 40 training examples per action was selected at random from the full example set. The remaining examples per action (excluded from the training set) were then classified.

Classification performance is summarized in the confusion matrix shown in Tab. 1. Each confusion matrix cell A_{ij} represents the probability of error in classification. The entry at A_{ij} is the probability of

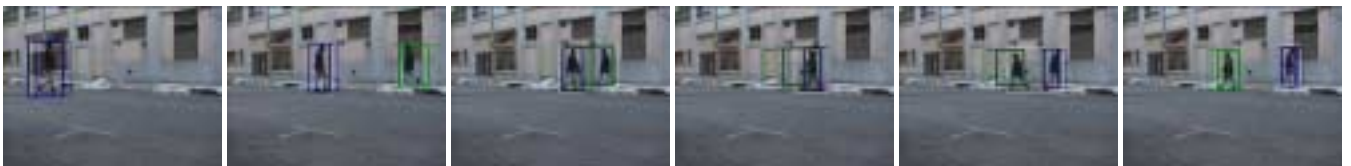


Figure 15: Tracking results, non-linear trajectories. Bodies are tracked, but prediction is less accurate because of sudden change in direction during occlusion.

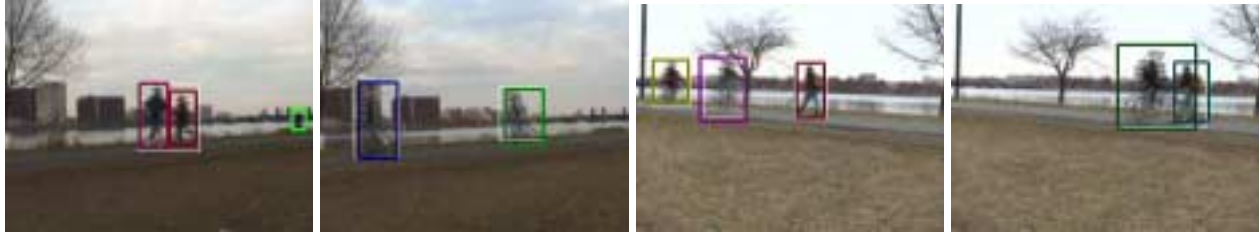


Figure 16: Example frames taken from the river sequences.

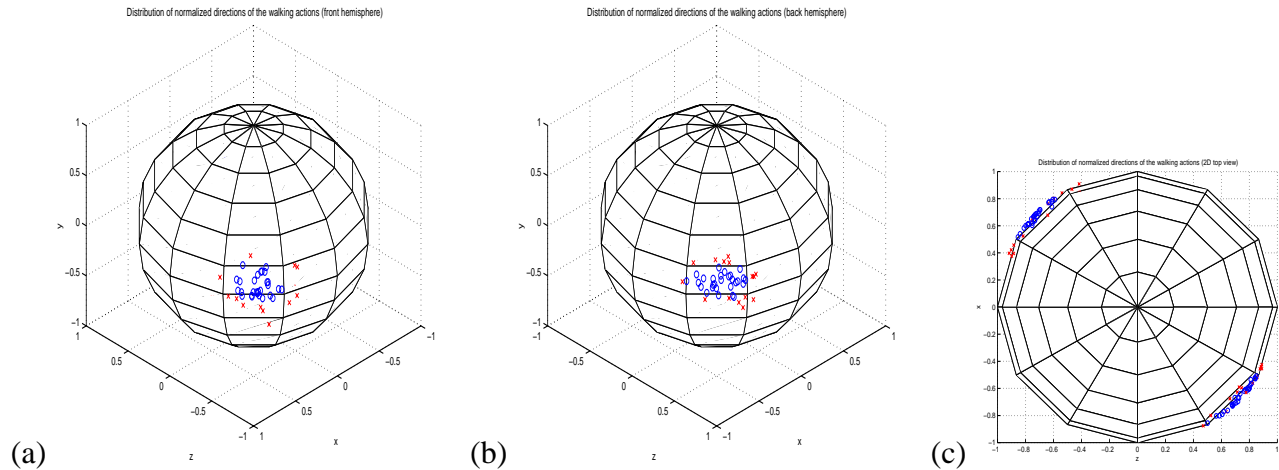


Figure 17: Distribution of normalized trajectory directions for the walking actions. Frontal view of the trajectory direction sphere (a), back view (b), and top view (c). Blue circles indicate data points falling on tessellations where the class could be statistically characterized, red cross-marks indicate data falling in other tessellations.

action i being misclassified as action j . The last column is the total probability of a given action being misclassified as another action. The last row represents the probability of misclassification to action class i . In the experimental trials, the total probability of an error in classification $P(e) = 0.227$ (chance = 0.75).

8.3 Sensitivity Experiments

In order to provide a more comprehensive analysis of the 3D trajectory estimation technique, we tested its sensitivity with synthesized data sets. We conducted Monte Carlo experiments to evaluate the stability of trajectory estimation and prediction. In our experiments, two types of noise effects were tested: noise in the 2D image measurements, and noise in the 3D motion model.

Test sequences were generated using a synthetic planar bounding box moving along varying direc-

Actions	Walking	Running	R. blading	Biking	Totals
Walking	-	0.21	0.12	0.02	0.35
Running	0.19	-	0.08	0.01	0.28
R. blading	0.11	0.12	-	0.00	0.23
Biking	0.02	0.02	0.01	-	0.05
Totals	0.32	0.35	0.21	0.03	0.227

Table 1: Confusion matrix for classifying four action classes: walking, running, roller blading, and biking. In the experimental trials, the total probability of an error in classification $P(e) = 0.227$ (chance = 0.75).

tions in 3D from a given starting position. The 3D box was then projected onto the image plane using our camera model (Eq.:5) with unit focal length. Each synthetic image sequence was 100 frames long. The set of directions was sampled by the azimuth θ and elevation γ using $\Delta\theta = \Delta\gamma = \pi/24$. In total, there were $12 \times 48 = 576$ different directions. For each experiment, each of the 576 possible trajectory directions was tested using 15 sequences with randomly perturbed inputs. An azimuth $\theta = 0$ and elevation $\gamma = 0$ indicates a trajectory in a direction parallel to the camera view vector.

All synthetic video sequences were sampled at a pixel resolution of 512×512 . This was mapped to a physical view-port size of 2×2 world units. Therefore one pixel width is equivalent to 0.0039 in world units. The depths of the object from the image plane ranged in a scale from 0 to 20. This resulted in a projected bounding box that occupied approximately 3% of the image on average.

For all of our experiments we define the error in our estimate at a given time k to be measured in the image plane. This is formulated in the standard mean-square sense:

$$\mathcal{E}_k = \frac{1}{N} \sum_{i=1}^N (\hat{\mathbf{x}}_k^{(i)} - \mathbf{x}_k^{(i)})^2, \quad (27)$$

where $\hat{\mathbf{x}}_k^{(i)}$ is our estimate of the position of feature i on the image plane at time k , $\mathbf{x}_k^{(i)}$ is the actual position of the feature on the object, and N is the total number of point comparisons or features needed to fully describe the error in the estimate. In our case our object representation has four degrees of freedom on the image plane, so $N = 2$.

The mean squared error (MSE) in estimating the bounding box corner positions was computed over the 100 frames within each test sequence. To better understand the effect of error due to differences in the projected size of the object from frame to frame, a second error measure, *normalized MSE* was also computed. In this measure, the error at each frame was normalized by length of the projected bounding box diagonal.

To test the sensitivity of the system to noise in the measurements, white noise with variance σ_M^2 was added to the measured bounding box corner positions at each frame. This was meant to simulate sensor noise and variations in bounding box due to non-rigid deformations of the object. To test the sensitivity of the model formulation to perturbations in the actual 3D object trajectory, white noise with variance σ_S^2 was added to perturb the 3D position of the object at each frame. The resulting trajectories were therefore not purely linear.

Our experiments have consistently shown that the mean-square error depends exclusively on the azimuth angle of the trajectory, θ . We found that mean-square error is relatively invariant to elevation (γ). Due to this result, our graphs drop the elevation parameter by averaging over it, so that the complexity of visualization is reduced.

Fig. 18 shows results of experiments designed to test the effect of increasing the measurement noise. We set $\sigma_S^2 = 0.01$, relatively low with respect to the real 3D dimensions, and varied σ_M^2 . As expected, the error is lower at lower noise levels. The first graph shows the normalized mean-square error in the state estimate over various trajectory directions. The second graph shows the normalized mean-square error in the state predicted for the future frame $k + 10$ (ten frames ahead). This error is due to the linearization step in the EKF. This error reduces the accuracy of “look ahead” needed to foresee an occlusion, and to predict the state during occlusions.

The graph in Fig. 19 shows the results of experiments designed to test the effect of increasing noise in the 3D motion trajectory. This corresponds to noise added to the model. Here, σ_S^2 is varied as shown and $\sigma_M^2 = 0.0001$ is kept relatively small. Note that the normalized mean-square error in the estimates is relatively constant over all values of θ and with different levels of noise. The main reason for this is

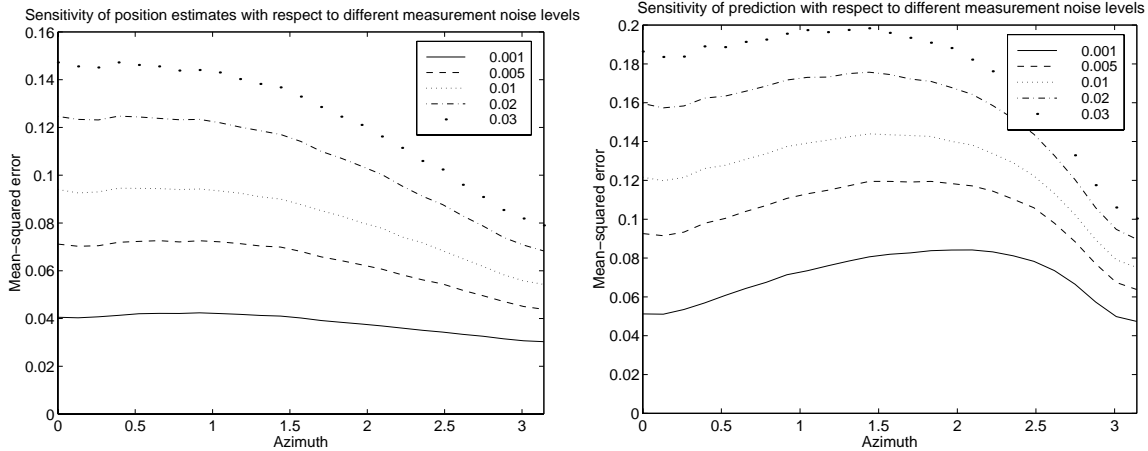


Figure 18: Graphs showing the sensitivity with respect to varying levels of measurement noise. The first graph shows the normalized mean-square error in the state estimate over various trajectory directions. The second graph shows the normalized mean-square error in the state predicted for the future frame $k + 10$.

that perturbation is consistent with the underlying motion model employed in the EKF. The prediction error in general increases with σ_S^2 and θ , showing the higher error in linearizing among the current state space point. The error decreases after a given value for θ due to the normalization effect and the smaller effect that θ close to π has with respect to changes on the image plane.

In a final set of experiments, we tested the accuracy of the EKF's trajectory prediction, at varying Δk frames into the future. Results are shown in Fig.20. Notice that as expected, the 3D trajectory prediction is more accurate in short time windows. The uncertainty increases with the number of frames in the future we want to make our prediction. This is mainly due to the fact that the EKF computes an approximation linearizing around the current point in state space, and as we pointed out the underlying process is non-linear. The prediction error in general increases with θ , showing the higher error as consequence of a linearized state.

9 Conclusions

We have shown how to integrate low-level and mid-level mechanisms to achieve more robust and general tracking. 3D trajectories can be successfully estimated, given some constraints on non-rigid objects. Prediction and estimation based on a 3D model gives improved performance over 2D image plane based prediction. This trajectory estimation can be used to predict and correct for occlusion.

In our system, it is assumed that the object trajectory is locally linear while the object can be independently segmented. While a limitation, this is not particularly limiting constraint in most situations. However, if the moving object undergoes a sudden change in direction during occlusion, the linear prediction will be inaccurate. We note that in some cases, correct correspondence after occlusion can still be achieved despite a noticeable change in the object trajectory during occlusion. This is due to the bootstrapping mechanism employed.

The system can accurately track multiple subjects in most situations and with no user interaction required, mainly due to the power of the detection mechanism and the motion estimation techniques. Handling occlusions using motion trajectories is inherently difficult if the expected motion model is violated during critical estimation stages as was shown in Fig. 15. In our experiments prediction estimates were not very reliable if the object had been tracked for (roughly speaking) fewer than five frames before the occlusion. This unreliability is indicated statistically in the state error estimate. The use of

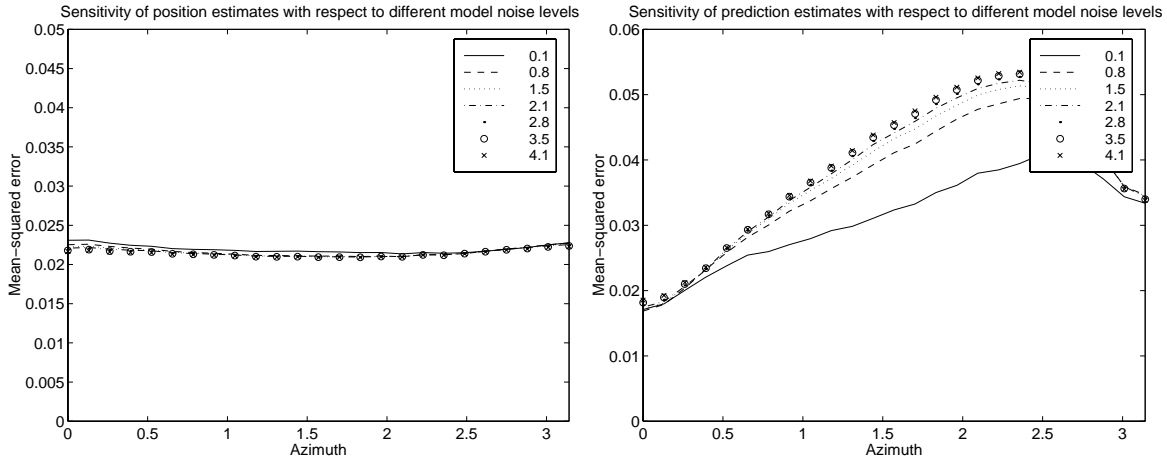


Figure 19: Sensitivity with respect to varying levels of noise in the 3D motion trajectory. The first graph shows the normalized mean-square error in the state estimate over various trajectory directions. The second graph shows the normalized mean-square error in the state predicted $k + 10$.

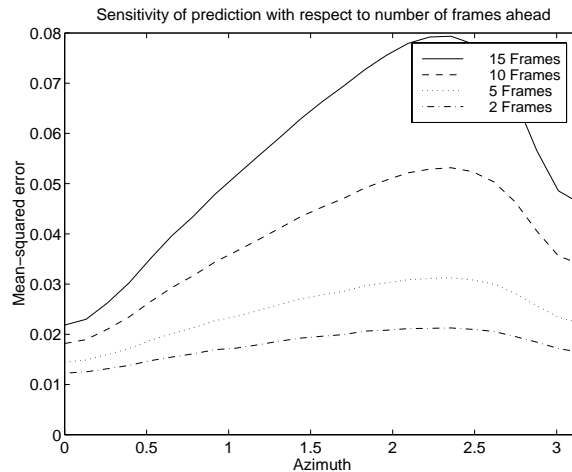


Figure 20: Sensitivity with number of frames ahead used to calculate the prediction. Normalized mean-square error over $\theta = [0, \pi]$

other features, like color or shape, would be a natural extension to provide a more robust mechanism.

Recovery of trajectories of non-rigid objects has been normally approached assuming no structure (*e.g.*, tracking subjects seen as points) or using domain constraints (*e.g.*, ground plane known or collapsing one dimension by using a top-view of the scene). For motion analysis, enough information about the shape of the objects and how they evolve is needed. Many previous approaches cannot provide this, even though they make the recovery of trajectories a lot simpler.

We utilized 3D trajectory estimates in a new framework: Trajectory-Guided Recognition (TGR). This general method significantly reduces the complexity of action classification, and could be used with other techniques (*e.g.*, [6, 38]). Our tracking approach allows the construction of an object centered representation. The resulting translation/scale stabilized images of the object are then fed to the TGR action recognition module that selects the appropriate classifier based on trajectory direction.

The system was tested in classifying four basic actions in a large number of video sequences collected in unconstrained, outdoor scenes. The noise stability properties of the trajectory estimation sub-

system were also tested using synthetic data sequences. The results of the experiments are encouraging. Classification performance was quite good, considering the complexity of the task.

For convergence, the EKF needed about 40 frames on average in our experiments. The performance and generality of the system can be improved by using an Interacting Multiple Model approach (IMM) [9, 7]. In this approach, n EKF's with different dynamics properties are run together and the system determines which one better describes the observations. This could allow for the estimation of positions when we want to consider different model dynamics.

References

- [1] A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal length. *PAMI*, 17(6), 1995.
- [2] A. Baumberg and D. Hogg. An efficient method for contour tracking using active shape models. In IEEE Computer Society Press, editor, *IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 194–199, November 1994.
- [3] A. Baumberg and D. Hogg. Learning flexible models from image sequences. In *European Conference on Computer Vision*, volume 1, pages 299–308, May 1994.
- [4] M. Bichsel. Segmenting simply connected moving objects in a static scene. *PAMI*, 16 (11):1138-1142, 1994.
- [5] M. Black and A. Jepson. A probabilistic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *ECCV*, 1998.
- [6] M. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motion using local parametric models of image motion. In *ICCV*, 1995.
- [7] H. Blom. An efficient filter for abruptly changing systems. In *Proc. Conf. on Decision Control*, 1984.
- [8] A. Bobick and J. Davis. An appearance-based representation of action. In *ICPR*, 1996.
- [9] K. Bradshaw, I. Reid, and D. Murray. The active recovery of 3d motion trajectories and their use in prediction. *PAMI*, 19(3), 1997.
- [10] C. Bregler. Tracking people with twists and exponential maps. In *CVPR98*, 1998.
- [11] T. Broida and R. Chellappa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *PAMI*, 13(6):497-513, 1991.
- [12] T. Cootes, D. Cooper, C. Taylor, and J. Graham. Trainable method of parametric shape description. *Image and Vision Computing*, 10(5):289-294, 1992.
- [13] T. Darrell and A. Pentland. Classifying hand gestures with a view-based distributed representation. In *NIPS*, 1994.
- [14] J. Davis and A. F. Bobick. The representation and recognition of human movement using temporal templates. In *CVPR*, 1997.
- [15] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data. *Journal of the Royal Statistical Society (B)*, 39(1), 1977.
- [16] S. Fejes and L. Davis. What can projections of flow fields tell us about the visual motion. In *ICCV*, 1998.
- [17] H. Fujiyoshi and A. Lipton. Real-time human motion analysis by image skeletonization. In *WACV*, 1998.
- [18] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1972.
- [19] D. Gavrilu and L. Davis. Tracking of humans in action: a 3-d model-based approach. In *Proc. ARPA Image Understanding Workshop, Palm Springs*, 1996.
- [20] W.E.L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *CVPR*, 1998.
- [21] L. Davis I. Haritaoglu, D. Harwood. W4s: A realtime system for detecting and tracking people in 2.5d. In *ECCV*, 1998.
- [22] S. Intille and A. F. Bobick. Real time close world tracking. In *CVPR*, 1997.
- [23] M. Irani and P. Anandan. A unified approach to moving object detection in 2d and 3d scenes. *PAMI*, 20(6), 1998.
- [24] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *IJCV*, 29(1): 5-28, 1998.
- [25] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *ECCV*, 1998.
- [26] N. Johnson and D. C. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14:609–615, 1996.
- [27] S. Ju, M. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Proc. Gesture Recognition*, 1996.

- [28] I. Kakadiaris, D. Metaxas, and R. Bajcsy. Active part-decomposition, shape and motion estimation of articulated objects: A physics-based approach. In *CVPR*, 1994.
- [29] T. Kanade and et. al. Advances in cooperative multi-sensor video surveillance. IFD PI Overview Paper. Darpa 98.
- [30] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59-69, 1982.
- [31] M.W. Krueger. *Virtual Reality II*. Addison-Wesley, 1990.
- [32] D. Metaxas and D. Terzopoulos. Shape and non-rigid motion estimation through physics-based synthesis. *PAMI*, 15(6):580-591, 1993.
- [33] D. Morris and J. Rehg. Singularity analysis for articulated object tracking. In *CVPR98*, 1998.
- [34] R. Nelson. Qualitative detection of motion by a moving observer. *International Journal of Computer Vision*, 7(1), 1991.
- [35] S. Niyogi and E. H. Adelson. Analyzing and recognizing walking figures in xyt. In *CVPR*, 1994.
- [36] N. Oliver, B. Rosario, and A. Pentland. Statistical modeling of human interactions. In *IEEE CVPR Workshop on the Interpretation of Visual Motion*, 1998.
- [37] A. Pentland and B. Horowitz. Recovery of non-rigid motion and structure. *PAMI*, 13(7):730-742, 1991.
- [38] R. Polana and R. Nelson. Low level recognition of human motion. In *Proc. IEEE Workshop on Nonrigid and Articulate Motion*, 1994.
- [39] B. Rao, H. Durrant-Whyte, , and J. Sheen. A fully decentralized multi-sensor system for tracking and surveillance. *Int. J. of Robotics Research*, 12(1), 1993.
- [40] J. M. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *ICCV*, 1995.
- [41] P. Remagnino, A. Baumberg, T. Grove, D. Hogg, T. Tan, A. Worrall, and K. Baker. An integrated traffic and pedestrian model-based vision system. In *British Machine Vision Conference*, 1994.
- [42] P. Remagnino, T. Tan, and K. Baker. Agent orientated annotation in model based visual surveillance. In *ICCV*, 1998.
- [43] D. Reynard, A. Wildenberg, A. Blake, and J. Marchant. Learning dynamics of complex motions from image sequences. In *ECCV*, 1996.
- [44] K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP:IU*, 59(1):94-115, 1994.
- [45] R. Rosales. Recognition of human action using moment-based features. Technical Report BU 98-020, Boston University, 1998.
- [46] R. Rosales and S. Sclaroff. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. In *IEEE CVPR Workshop on the Interpretation of Visual Motion*, 1998.
- [47] R. Rosales and S. Sclaroff. 3d trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *CVPR*, 1999.
- [48] H.W. Sorenson. Least-squares estimation: From gauss to kalman. *IEEE Spectrum*, Vol. 7, pp. 63-68, 1970.
- [49] R. Szeliski and S. Bing Kang. Recovering 3d shape and motion from image streams using non-linear least squares. In *CVPR*, 1993.
- [50] G. Welch and G. Bishop. An introduction to the kalman filter,. Technical Report TR 95-041, Computer Science, UNC Chapel Hill, 1995.
- [51] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfunder: Real time tracking of the human body. Technical Report TR 353, MIT Media Lab, 1996.
- [52] M. Yamamoto and et. al. Incremental tracking of human actions from multiple views. In *CVPR98*, 1998.
- [53] J. Yamato, J. Ohya, and K. Isii. Recognizing human action in time sequential images using hidden markov models. In *CVPR*, 1993.