A Framework for Heading-Guided Recognition of Human Activity

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

Version of September 11, 2001

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. Heading-guided recognition (HGR) is proposed as an efficient method for adaptive classification of activity. The HGR approach is demonstrated using "motion history images" that are then recognized via a mixture-of-Gaussians classifier. The system is tested in recognizing various dynamic human outdoor activities: running, walking, roller blading, and cycling. In addition, experiments with real and synthetic data sets are used to evaluate stability of the trajectory estimator with respect to noise.

Keywords: motion tracking, recursive estimation, motion recognition, video surveillance, temporal templates, computer vision.

1 Introduction

Tracking multiple humans in cluttered environments, and analyzing their perceived motion are challenging problems in computer vision. Only in well controlled situations, normally not useful for common applications, have researchers been able to obtain satisfactory results. Effective solutions to these problems would lead to breakthroughs in computer human interfaces and video surveillance, as well as in ergonomics, motion-based performance measurement, virtual reality systems, computer animation, and robot navigation.

Low-level image processing methods have been shown to work surprisingly well in restricted domains despite the lack of explicit high-level models [15, 9, 40, 29, 31, 27]. However, 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 [17, 47, 12, 25, 36, 37, 51, 54, 69], such methods are highly specific, typically requiring careful placement of the initial model, multiple cameras, etc.

Many of the 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.

We aim to employ a theoretically well-founded model, while at the same time providing computational efficiency. This balance is in general hard to achieve. 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. Furthermore, direction of heading estimates obtained during tracking are used to reduce the complexity of the nonrigid motion recognition task.

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 many previous approaches that assume a top or very distant view of the scene [45, 30]. As will be demonstrated in our framework, a less restrictive camera viewpoint can provide more substantial information for use in higher-level mechanisms; *e.g.,* human motion analysis, understanding and recognition. The framework has been tested in recognizing various dynamic human outdoor activities; *e.g.,* running, walking, roller blading, and cycling. Furthermore, 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 human motion tracking and action recognition is given. The relationship of previous work directly related to our approach will be further elaborated as the details of our formulation are introduced.

Our formulation for 3D motion trajectory recovery utilizes recursive estimation theory, in particular the extended Kalman filter (EKF). The EKF approach has proven to be useful in recovery of rigid motion and structure from image sequences [1, 13, 10, 49, 53, 58] and non-rigid motion [41, 46]. One of the first important results on recursive structure and motion estimation via an EKF was the work of [13]. The formulation of [1] yields improved stability and accuracy of the estimates. In both methods, image feature tracking and correspondence are assumed. In contrast with previous approaches, we propose a framework that automatically tracks multiple moving objects, without the need for feature tracking or feature correspondence. The resulting tracking information is then used to estimate 3D trajectories up to a scale factor.

More recently, a particle filtering formulation to tracking has become popular [6, 33, 34]. Such a formulation can account for multimodal state densities, common when edge maps are used as observations, which the Kalman Filter is not designed to handle. While we test our proposed approach in the context of unimodal state distributions, it would be relatively straightforward to incorporate sampling algorithms 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, [10] 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 [31]. Some simple heuristics about body part relations

and motion in the image plane can also be used [29, 27]; however, this strongly limits the extensibility of the system. In our work, we do not assume planar motion nor do we employ detailed knowledge about the object. Instead, we assume locally linear trajectories. More recent approaches for tracking include the use of tensor voting [39] and mean shift algorithms [14]. These approaches have not yet unified tracking and action recognition in the same framework.

More detailed representations generally use 2D and 3D articulated human body models [12, 17, 47, 25, 36, 37, 46, 51, 54, 69]. The most recent work at the level of joint/part location body tracking has concentrated on sampling methods [19, 62, 34] (given a learned model for the probability distribution over configurations and motions), and 3D reconstruction from 2D markers [2, 28]. However, estimating 2D joint location is still a very hard problem by itself [60]. At this level of detailed modeling, we can also find [11, 63, 59]. Such approaches can account for self-occlusion by relying on the object model or learned probability distributions; e.g., Hidden Markov Models [11], triangulated-graph Markov random fields [63], or non-linear mixture models [59]. However, these models tend to be rather detailed, and typically require one or more of the following: high resolution imagery, manual interaction/initialization of models, multiple cameras, intrusive devices, or well-controlled environments. In this paper we do not focus on pose estimation or recovery of other detailed representations (see [42, 24] for excellent reviews); instead we focus on 3D trajectory estimation, tracking in the presence of multi-object occlusion, and action recognition.

Image contour models [4, 3, 52, 34], because they are less detailed, avoid the problems associated with 2D and 3D articulated human body models. In these systems, human motion generally is constrained to walking/standing like configurations. The approach presented in this paper is more general with respect to the appearances that it can be handle.

Another aspect of our system is motion recognition. Previous approaches can be divided roughly into two categories: trajectory modeling and appearance change modeling approaches. In trajectory modeling, typically the motion of each object's centroid is estimated and tracked over time. Trajectory modeling [50, 26, 35, 43, 45, 61] 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 [16, 7, 23, 61, 15, 9, 48] look at the image motion of the object itself instead of static configurations or too compact representations such as centroids, etc. In particular, [9] used motion history images (MHI) and motion energy images (MEI). MHIs and MEIs are temporal image templates that are matched against examples of given motions already learned. Recently, [16] extended MHI to hierarchical MHI's. Of course, other view-based representations can be used

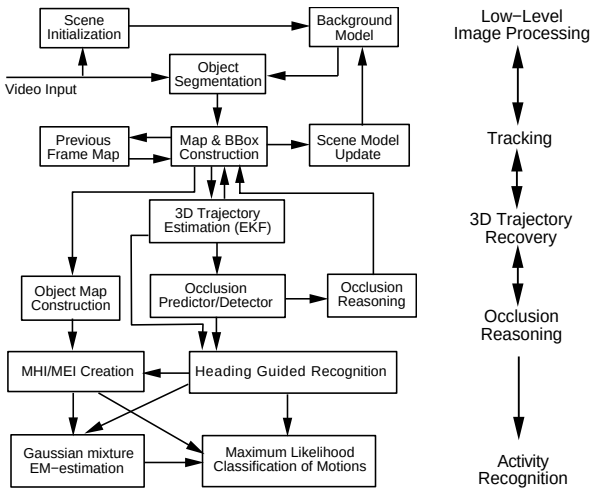


Figure 1: System Diagram.

in a similar way, *e.g.,*image skeletonization [21]. An important challenge with appearance change modeling is the need to normalize for gross changes in scale, 3D orientation, translation, etc. Since the approaches are image-based, an estimate of the mapping between the input image and the appearance prototypes must be reliably estimated. 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.

It should also be noted that the estimated 3D trajectories can be used directly in trajectory-based recognition of human activity; e.g., [45, 30]. However, a trajectory-based recognition system is not developed in this paper. Instead we exploit a 3D motion model and occlusion modeling to gain more reliable object correspondence, as well as to aid in view-based action 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 shown in Fig. 1. The system has both feed-forward and feed-back mechanisms. In our framework, the low-level image processing is based on the scene modeling methods of [68]. Here, the color distribution of each pixel is modeled using a Gaussian distribution. Once the empty scene is modeled, moving objects in the scene are segmented using a log-likelihood measure between the incoming frames and the current background model. Connected components analysis and morphological operations are also employed to improve segmentation quality. Example input frames and the resulting foreground

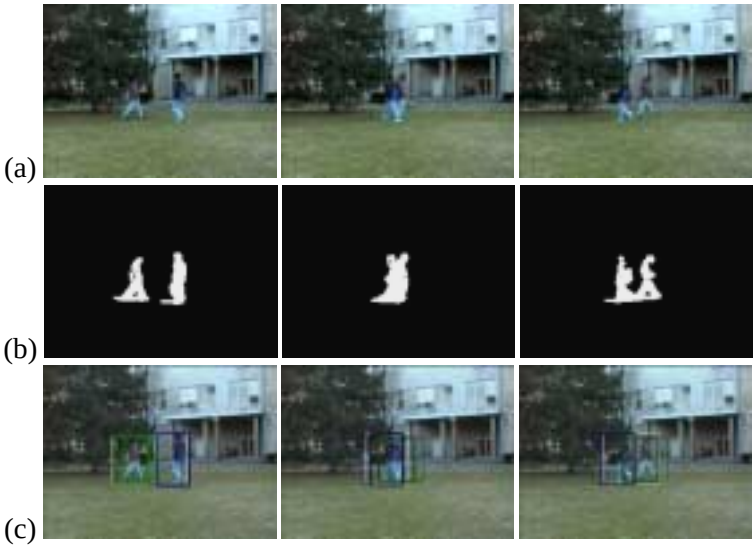


Figure 2: Tracking before, during, and after occlusion: (a) example frames from the input sequence, (b) connected components extracted from the background, (c) estimates of the bounding boxes for the moving objects. The bounding boxes are successfully predicted during the occlusion via an extended Kalman filter.

segmentation for a video sequence are shown in Fig. 2(a,b).

If there are occlusions, or strong similarities between the empty scene model and the objects, then it is possible that regions are segmented erroneously. 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, spatial correlation analysis, and trajectory estimation/prediction via an extended Kalman filter (EKF). The input to the EKF is a 2D bounding box that encloses the moving object in the image. The EKF then predicts the relative 3D motion trajectories for each object, based on a 3D trajectory model. The EKF also provides an estimate of each object's image bounding box position and velocity. An example of estimated bounding boxes for tracked objects is shown in the bottom for of Fig. 2 (c). In contrast to trajectory prediction based on a 2D model, our 3D formulation is explicitly designed to handle nonlinear effects of perspective projection.

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, then the object's bounding box can be predicted via the EKF and previous tracking information. This previous tracking information is statistically maintained in the EKF state and error covariance matrix. During an occlusion, the EKF can be used to give the maximum likelihood estimate of the current bounding box for the object, along with its predicted velocity and position.

As shown in Fig. 3, each moving object is resampled in a object-centered frame throughout the tracking sequence, despite changes in scale and position due to locomotion. The resulting object-centered motion

(a)  (b)  (c) 

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

sequence can be used as input to motion recognition modules. Actions are represented in terms of motion energy images (MEIs) and motion history images (MHIs) [9]. An MEI is a cumulative motion image, and an MHI is a function of the recency of the motion at every pixel. By using object-centered input sequences, it is possible to make the MEI/MHI approach invariant to locomotion. It allows the recognition of movements performed while a person is locomoting. The object-centered representation is then fed into a moment-based action classifier. The action recognition module employs a mixture of Gaussian classifier, which is learned via the Expectation Maximization (EM) algorithm. However, given that we would like view-point independent recognition, we must incorporate some key representational elements.

The MEI/MHI is a view-based approach to action recognition. In theory it is necessary to learn representations of every action for all possible viewing 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 *heading-guided recognition* (HGR). HGR is a direct consequence of our tracking and 3D trajectory estimation mechanisms. In HGR, we partition the sphere of possible motion heading directions based on the trajectories estimated in the training data. Each partition corresponds to a group of similar heading directions. During training and classification, heading 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 approximately those headings that are commonly observed.

4 Looking for Multiple Moving Objects

In order to detect object motion, the use of a fixed camera makes background-foreground segmentation methods well suited for this task [4, 5, 31, 26, 54, 68]. This approach is computationally efficient enough to allow real-time tracking. A moving camera can also be used, by building a background mosaic and estimating camera motion, and/or detecting independently moving objects [32, 44, 20]; however, such generality comes 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 segmen-

tation. Unfortunately, non-rigid body tracking cannot easily benefit from optical flow computation. This is 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 those used here) is insufficient for accurate flow estimation. Moreover, occlusions due to multiple moving objects further complicate this process.

4.1 Scene Model Initialization

Our initial low-level step is based on the results of [68], in which a Gaussian distribution for pixel color is assumed [1]. Using this model, for a given pixel $\mathbf{p}$, it is only necessary to estimate its sufficient statistics, its mean $\hat{\mu}_{\mathbf{p}}$ and covariance $\mathbf{K}_{\mathbf{p}}$ in 3D color space, over a collection of N training frames. This background model is simple, and sufficient to demonstrate our heading-guided activity recognition framework. However, adaptive statistical models have been proposed recently [26, 65]. Such adaptive models provide improved robustness to large illumination changes, as well as sudden motion (e.g., leaf motion due to wind, etc.), and can easily be used in our framework.

4.2 Segmentation

We will first describe the segmentation procedure used during regular conditions (no occlusions). Following [68], objects in the scene are characterized by a relatively large variation from the background statistics. For each incoming image scene changes are detected using the log-likelihood measure at every pixel, which is the optimal decision function under our pixel color model:

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

where $\mathbf{I}(\mathbf{p},k)$ is the image intensity at pixel $\mathbf{p}$ at time k in color space. A binary map of the current frame is then computed: $\mathbf{b}_i(k) = \bigcup_{\mathbf{p}}[\mathbf{d}_{\mathbf{p}}(k) < \Gamma]$. The connected region $\mathbf{b}_i(k)$ at time k is defined by the locations where there was a difference from the model was greater than a given threshold Γ. Image morphological operations are applied (erosion-dilation) in order to rule out small regions and to fill holes in probable regions of interest. Finally, to improve reliability of the segmentation method of [68], we define a merging function $\mathbf{g}$ that applies three additional criteria for merging previously separate blobs $\mathbf{b}_i(k)$:

1. **Frame difference merging criterion.** The difference between the current and previous frame is computed and thresholded for all pixels $\mathbf{p}$, forming the binary image: $\mathbf{I}_d(\mathbf{p},k) = (\mathbf{I}(\mathbf{p},k) - \mathbf{I}(\mathbf{p},k-$

[1] Moreover pixel colors are assumed independent across pixels

$1) > \Gamma'$), with threshold Γ'. Two blobs $\mathbf{b}_i(k)$ and $\mathbf{b}_j(k)$ are merged if there is a connected component from $\mathbf{I}_d(\mathbf{p}, k)$ that intersects both of them. The reason for this is that often when $\mathbf{d}_\mathbf{p}(k)$ is low (poor contrast), objects' parts tend to form different blobs $\mathbf{b}_i(k)$. In our experiments we have found that use of this criterion generally improves segmentation accuracy without oversegmenting the scene.

2. **Spatial analysis criterion.** Each object's region of support (or blob) is stored and used as a rough approximation of the region in the subsequent frame. If two regions are not connected in the subsequent frame (due to errors in initial segmentation using Eq. 1), but they were connected in the previous frame, then they are merged. This can account for some shadow effects, local background/foreground similarities, and brief occlusions causing temporary loss of connectivity in an object. This temporal matching assumes that object trajectories are continuous and that object positions change at a rate smaller than the object size in the direction of motion. When occlusions are detected this process changes as described in Sec. 6.

3. **3D trajectory estimation criterion.** The higher-level information obtained from the 3D trajectory estimate is used to predict the location of the object in the subsequent frame. A projection on the image plane of the estimated 3D position is used to merge all blobs $\mathbf{b}_i(k)$ falling in the area predicted to be occupied for a given object.

This simple approach may fail when objects first enter the scene overlapping 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 they have already appeared in the scene as separate objects. In this case our occlusion reasoning process will predict this, and properly identify and label the objects.

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

4.3 Updating the Scene Model

After a binary scene map $\mathbf{M}(k)$ is obtained with all the possible objects in the scene, the system updates the statistical properties of the scene background. In our implementation only the mean is updated via a simple adaptive filter:

$$\hat{\mu}(k) = \alpha\hat{\mu}(k-1) + (1-\alpha)[\mathbf{I}(\mathbf{P}, k-1)\overline{\mathbf{M}}(k-1) + \hat{\mu}(k-1)\mathbf{M}(k-1)], \tag{2}$$

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

5 Estimating 3D Trajectories from 2D Image Motion

A 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. Our approach is also related to the ideas of [10] in the use of an EKF to predict and recover object trajectories. However, unlike [10] no planar motion assumption is necessary. This allows use of our formulation in a more general setting.

5.1 Camera Model

For our representation a 3D central projection model similar to [1, 66] is used:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \frac{1}{1+z\beta}, \tag{3}$$

where (x, y, z) is the real 3D point location in the camera reference frame, (u, v) is the point's projection onto 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. One property of this model is that it is numerically well defined even in the case of orthographic projection.

5.2 3D Feature Representation

To reduce the complexity of the tracking problem, two feature points are selected: two opposite corners in the object bounding box. These feature point locations will be parameterized in terms of the object's bounding box coordinates x, y, in addition to the depth coupled with the inverse focal length $z\beta$. Similarly, we use the vector $(\dot{x}, \dot{y}, \dot{z\beta})$ to represent a feature's 3D velocity relative to the camera reference frame. 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.

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 limiting in some cases; *e.g.*, if the internal motion of the object's parts cannot be roughly contained in a bounding box. It is important to mention that the choice of features to track is more appropriate for rigid objects, *e.g.*, car, bikes, airplanes, etc. However, when analyzing basic human locomotion, we believe that these assumptions provide a fair approximation that balances simplicity and utility. Arm and leg motion affect the rigidity of the bounding box, but if our observation noise model is accurate enough, then the general EKF formulation allows this effect to be modeled as noise.

5.3 Extended Kalman Filter Formulation

We will use a first order extended Kalman filter (EKF). Our state models a planar rectangular box moving along a linear 3D trajectory at constant velocity. Changes in direction and velocity are modeled with a Gaussian noise model. Note that this does not imply that the object must move on a line. Our EKF state vector is as follows:

$$\mathbf{x} = (x_0, y_0, x_1, y_1, z\beta, \dot{x}_0, \dot{y}_0, \dot{x}_1, \dot{y}_1, \dot{z\beta})^T, \tag{4}$$

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 $(\dot{x}_0, \dot{y}_0, \dot{z\beta})^T$ and $(\dot{x}_1, \dot{y}_1, \dot{z\beta})^T$ 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 $z\beta$ speed. This was conceived as a way to link the two spatial positions and increase stability by reducing the degrees of freedom in the system.

Our EKF process is then guided by the standard linear difference equation [64, 67]:

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{w}_k, \tag{5}$$

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 [53].

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 is non-linear, since it includes perspective projection. Measurement noise is assumed to be additive in our model. The EKF time update and measurement update equations are shown in Fig. 4.

Because the measurement relationship to the state is nonlinear, at each step, the EKF approach requires a linearization of h around our current estimate, yielding $\mathbf{H}_k$ the Jacobian of h with respect to $\mathbf{x}_k$. Therefore, we consider a linear Taylor approximation of the system around the current estimate. This approach provides a simple and efficient way to calculate the Kalman gain under a nonlinear model. Without this

Iterate

Time Update Equations

$$\hat{\mathbf{x}}_{k+1}^{-} = \mathbf{A}_k \hat{\mathbf{x}}_k \qquad (6)$$

$$\mathbf{P}_{k+1}^{-} = \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^T + \mathbf{Q}_k \qquad (7)$$

Measurement Update Equations

$$\mathbf{K}_k = \mathbf{P}_k^{-} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^{-} \mathbf{H}_k^T + \mathbf{V}\mathbf{R}_k \mathbf{V}^T)^{-1} \qquad (8)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k (\mathbf{z}_k - h(\hat{\mathbf{x}}_k^{-}, 0)) \qquad (9)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^{-}, \qquad (10)$$

$\mathbf{Q}_k$ is the process noise covariance. $\mathbf{H_k}$ is the Jacobian of the observation function h at the current state estimate $\mathbf{x}_k$ (*i.e.,*we linearize around $\mathbf{x}_k$). Using our state and observation definitions, $\mathbf{H}_k$ can be found as follows:

$$\mathbf{H}_k = \begin{bmatrix} \frac{1}{\lambda} & 0 & 0 & 0 & -\frac{x_0}{\lambda^2} \\ 0 & \frac{1}{\lambda} & 0 & 0 & -\frac{y_0}{\lambda^2} \\ 0 & 0 & \frac{1}{\lambda} & 0 & -\frac{x_1}{\lambda^2} \\ 0 & 0 & 0 & \frac{1}{\lambda} & -\frac{y_1}{\lambda^2} \end{bmatrix}, \qquad (11)$$

where $\lambda = 1 + z\beta$. $\mathbf{V}$ is the Jacobian of h 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 $p(\mathbf{w}_k) \sim N(0, \mathbf{Q}_k)$, and $p(\mathbf{v}_k) \sim N(0, \mathbf{V}\mathbf{R}_k\mathbf{V}^T)$. For more detail, see [64, 67].

Figure 4: EKF summary and update equations. In the case where there are no occlusions, the algorithm iterates between the time update and measurement update equations.

approximation, the Kalman gain cannot be calculated in closed form (in general). As pointed out by [70], linearization of a nonlinear model leads to small errors in the estimates, which in general can be neglected. However, if the current estimate $\mathbf{x}_k$ is far from the true one, then this first-order approximation can lead to bigger errors. For a thorough discussion of this approximation, readers are referred to [70].

5.4 Tracker Units

Given the EFK formulation above, a *tracker unit*, T_i is assigned to every tracked object. At any time step k, a tracker unit contains information about the object location, a binary image support map $\mathbf{s}_j(k)$, bounding box parameters, likelihood of occlusion, and time the object has been continuously 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. Recall that we cannot simply associate one region $\mathbf{s}_j(k)$ with one object, nor can we expect that a region will always be tracked. For instance, a blob may leave the scene indefinitely or may become occluded by a non-moving object. In addition, there are situations where a connected region $\mathbf{s}(k)$ is the projection of more than one object (*i.e.,*, for example during occlusion). A mechanism for predicting, detecting, and reasoning about occlusions is presented in the next section.

6 Occlusion Prediction, 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 [10, 12, 25, 31, 35, 68]. Occlusions present a major problem in tracking applications. Researchers have tried to address this problem by using multi-camera tracking, range, stereo, *etc*. 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 handling occlusions among various simultaneously moving objects.

Our occlusion detection module predicts future locations of objects, based on current estimates of 3D velocity and position. We use the system dynamics formulation of Sec. 5.3 and a prediction in the observation as follows:

$$\mathbf{x}_{k+\Delta k} = \mathbf{A}(\Delta k)_k \mathbf{x}_k \tag{12}$$

$$\mathbf{z}_{k+\Delta k} = h(\mathbf{x}_{k+\Delta k}). \tag{13}$$

In this way, it is possible to predict a collision of two or more objects in the image plane by comparing their predicted position and image area covered. If a collision of the objects is expected at a given number of frames in the future, we assume that occlusion is probable (see Fig. 5). Our confidence in this collision prediction for two particular tracker units T_i, T_j at time k is determined assuming an exponential distribution:

$$Occ(T_i, T_j, k) = \xi e^{-\xi Trace(\mathbf{P}_k^{(i)} + \mathbf{P}_k^{(j)})}, \tag{14}$$

where $\mathbf{P}_k^{(i)}$ is the EKF's current estimate of error covariance for tracked object i,given by Eq.10. This quantity incorporates, in a probabilistic form, our notion of confidence in the current estimate. This can be understood as the probability of occlusion given that collision is predicted. In our experiments $\xi = 0.8$. Ideally, ξ should be estimated (learned) from training data, which is a simple one-dimensional parameter estimation task. The training data would consist of examples where collision is predicted and information about whether occlusion occured or not. We experimentally found that this 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 described next is used. In this case the tracker units corresponding to the given objects are tagged for this potential event.

False alarms can occur, so the system checks for reconfirmation of occlusion at every time step 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 $\mathbf{s}_j(k)$, shared among occluded and occluding objects, provides a constraint on the possible location of any of these objects. Thus, $\mathbf{s}_j(k)$ is used as the observation for the tracker units that are interacting during an 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). Fig. 5 outlines the complete tracking algorithm described in Secs. 4- 6. It also provides details of the update equations used during occlusion.

7 Heading-Guided Recognition of Human Activity

The aforementioned framework enables tracking and segmentation of multiple, independently moving objects despite momentary occlusions. The framework yields an object-centered representation of each object's motion, as well as an estimate of each moving object's 3D motion trajectory and direction of heading. With this information, our attention turns to the action recognition task. For the sake of demonstrating our framework, we will represent actions in terms of motion energy images (MEIs) and motion history images (MHIs) [9, 16]. By using object-centered input sequences, it is possible to use the MEI/MHI approach in classifying movements performed while a person is locomoting. Furthermore, knowledge of a moving object's direction of heading can significantly reduce the complexity of recognizing actions from all possible viewing angles.

7.1 Motion Representation

Temporal and spatial characteristics of motion over a short series of frames can be summarized by an MHI and MEI (see [9, 16] for a complete explanation). This appearance-based approach can result in a very high-dimensional representation. A reduced-dimensional representation is required for use in motion classification problems. As proposed in [9], the seven Hu moment invariants are computed for each MHI and MEI. We denote them (at time k) $\mathbf{x}_k^{\mathbf{MHI}}$ and $\mathbf{x}_k^{\mathbf{MEI}}$ respectively. For each motion, the resulting features are combined in a 14-dimensional vector: $\mathbf{x}_k^{\mathbf{f}} = (\mathbf{x}_k^{\mathbf{MHI}}, \mathbf{x}_k^{\mathbf{MEI}})$. The dimension of $\mathbf{x}_k^{\mathbf{f}}$ is further reduced via principal components analysis (PCA) [22]. The PCA yields the reduced feature vector $\not{\phi}_k$. In our experi-

Parameters:

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

Iterate

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

1. Use ML estimator of Eq. 1 to compute a log-likelihood map $\mathbf{d_p}(k) = (\mathcal{B}_k, \mathbf{I}(\mathbf{p}, k))$.
2. Segment moving objects using $\mathbf{d_p}(k) > \Gamma$, where Γ is the segmentation threshold. Represent the set of all connected component units as $\mathbf{b}(k)$.
3. Compute final segmentation: $\mathbf{s}_j(k) = \mathbf{g}(\mathbf{b}(k))$, $j = 1 \ldots N$ where N is the total number of objects in the scene, $\mathbf{g}(\bullet)$ uses all blobs found at $k-1$. Our scene map at time k is $\mathbf{M}(k) = \bigcup_j [\mathbf{s}_j(k)]$.
4. For all segmented objects (for all j), use $\mathbf{s}_j(k)$ to compute their bounding boxes, and obtain observations $\mathbf{z_j}(k) = (u_{0k}, v_{0k}, u_{1k}, v_{1k})$ taken to be two opposite corners.
5. For every segmented object $\mathbf{s}_j(k)$, and every tracker unit T_i.
   (a) If $Occ(T_i, T_l, k) < \Gamma_{Occ}, \forall l$ (no occlusion predicted involving T_i)
      i. Project state $\mathbf{x}$ of object to the image plane using Eq. 5.
      ii. Find closest $\mathbf{s}_j(k)$, using a L_2 norm on the pair of corners of bounding boxes, and use it as observation $\mathbf{z}_k$.
      iii. Use Eqs: 6-10 to obtain $\hat{\mathbf{x}}^-_{\mathbf{k+1}}$, $\hat{\mathbf{P}}_{\mathbf{k}}$, and $\hat{\mathbf{x}}_{\mathbf{k}}$, and update global EKF state.
   (b) else ($Occ(T_i, T_l, k) > \Gamma_{Occ}$ for some l)
      i. Find closest connected element using Eq.5. Allow for possibly shared connected elements (*i.e.*,blobs that belong to two different objects).
      ii. Use Eqs: 6-10, with possibly shared observed bounding box as observation $\mathbf{z}_k$, and $\mathbf{R} = \lambda\mathbf{R}$ to obtain $\hat{\mathbf{x}}^-_{k+1}$ and $\hat{\mathbf{x}}_k$. The scalar parameter λ is set to a large value during occlusion (intuitively the goal is to make EKF pay little attention to observations during occlusions).
      iii. The error covariance remains unchanged during occlusion,$\hat{\mathbf{P}}_k = \hat{\mathbf{P}}_{k-1}$.
6. Update scene representation: $\hat{\mu}(k) = \alpha\mathbf{I}(\mathbf{P}, k-1)\mathbf{M}_{k-1} + (1-\alpha)\hat{\mu}(k-1)\overline{\mathbf{M}}_{\mathbf{k-1}}$.

Figure 5: Summary of tracking algorithm.

ments, 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.

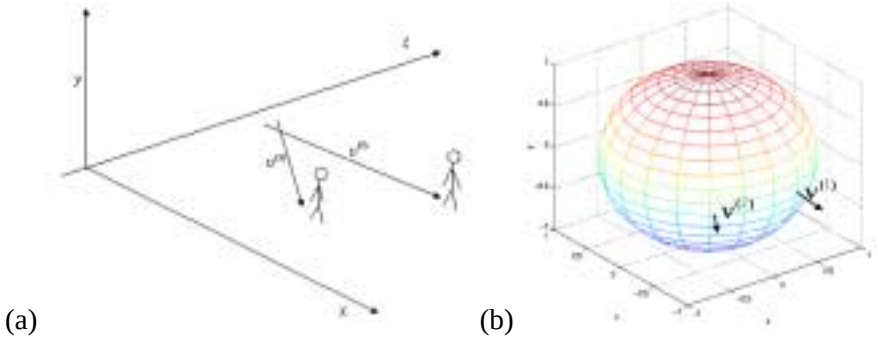
The reduced feature vector ϕ_k is then fed into a maximum likelihood classifier that is based on a mixture of Gaussians model. Each action class ω_i (in reduced Hu-moment space) is represented by a set of mixture model parameters, Θ_i. The conditional distribution $p(\omega_i|\phi)$ for each action class is estimated by maximizing

$p(\phi_i|\Theta_i)$ during a training phase using the expectation maximization (EM) algorithm [18], with ϕ the examples from the given class ω_i. Implementation details are provided in the Appendix.

7.2 Heading-Guided Recognition

In theory it is necessary to learn MEI/MHI representations of every action from all possible viewing directions. Let $\mathcal{P}_j$ be the set of PDF's used to represent m actions classes $\omega_1^{(j)}, ..., \omega_m^{(j)}$ performed by the object while moving under a given (quantized) direction j. Each action class $\omega_i^{(j)}$ has therefore its own PDF: $P(\omega_i^{(j)}|\phi) \in \mathcal{P}_j$ ($i = 1 \ldots m$). 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.

The problem can be made tractable via a new approach: *heading-guided recognition* (HGR). HGR is made possible as a direct consequence of our tracking and 3D trajectory estimation mechanisms. In HGR, we partition the direction sphere based on the heading directions estimated in the training data. Recall that trajectories are estimated by tracking the object of interest, and direction of heading at a given time is defined as the instantaneous estimate of the normalized velocity. Each partition j in the tessellated sphere corresponds to a group of similar instantaneous directions. The PDF's can then be automatically estimated for each bin in the heading direction space.



(a) (b)

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

Fig. 6 graphically presents this idea. At a given instant k subject 1, whose state vector is $\mathbf{x}_k^{(1)}$, 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 $\nu_k^{(1)} = \frac{1}{2}(\dot{x}_0 + \dot{x}_1, \dot{y}_0 + \dot{y}_1, 2\dot{z\beta})$. Similarly, subject 2 has a state $\mathbf{x}_k^{(2)}$ and average direction $\nu_k^{(2)}$. 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 associated with the set $\mathcal{P}_j$, where $\nu^{(1)}$ is in the range of trajectory directions defined by this set. This set contains one PDF per action.

Iterate

For each moving object l, at time k

1. Compute EKF trajectory estimate $\nu_k = \frac{1}{2}(\dot{x}_0 + \dot{x}_1, \dot{y}_0 + \dot{y}_1, 2\dot{z\beta})$ and the error covariance $\mathbf{P}_k$
2. If $Trace(\mathbf{P}_k) < \epsilon$ and object l is not occluded then
   (a) Compute stabilized MEI and MHI (choose time-window size)
   (b) Compute PCA feature vector ϕ_k
   (c) Find $\mathcal{P}_j$ whose orientation range contains ν_k.
   (d) Recognize the action by solving $i(\phi_k) = \arg\max_i(p(\omega_i^{(j)}|\phi_k))$, $p(\omega_i^{(j)}|\phi_k) \in \mathcal{P}_j$

The certainty threshold ϵ is set depending on how much accuracy is required in the trajectory estimate ν_k in order to perform recognition/learning. In our experiments $\epsilon = 1/\xi$.

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

A summary of the HGR algorithm is provided in Fig. 7.

Use of the HGR has a number of benefits. First, in many applications not all heading directions will be observed due to environmental constraints (e.g., motion along sidewalks). Thus, the PDFs need only be estimated for sphere partitions that correspond to observed heading directions. Second, during action recognition, an object's estimated 3D object trajectory can be used to lookup the appropriate PDFs, instead of exhaustively searching over all possible directions. Third, *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.,*[38]).

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 $\mathcal{P}_j$. Conversely, direction bins that are rarely observed (or not observed at all) will not contain a statistically reliable representation. Confidence measures estimating during learning of the mixture models will indicate this. We assume that the training data is representative of the environmental characteristics and accurately spans the range of possible observable motions.

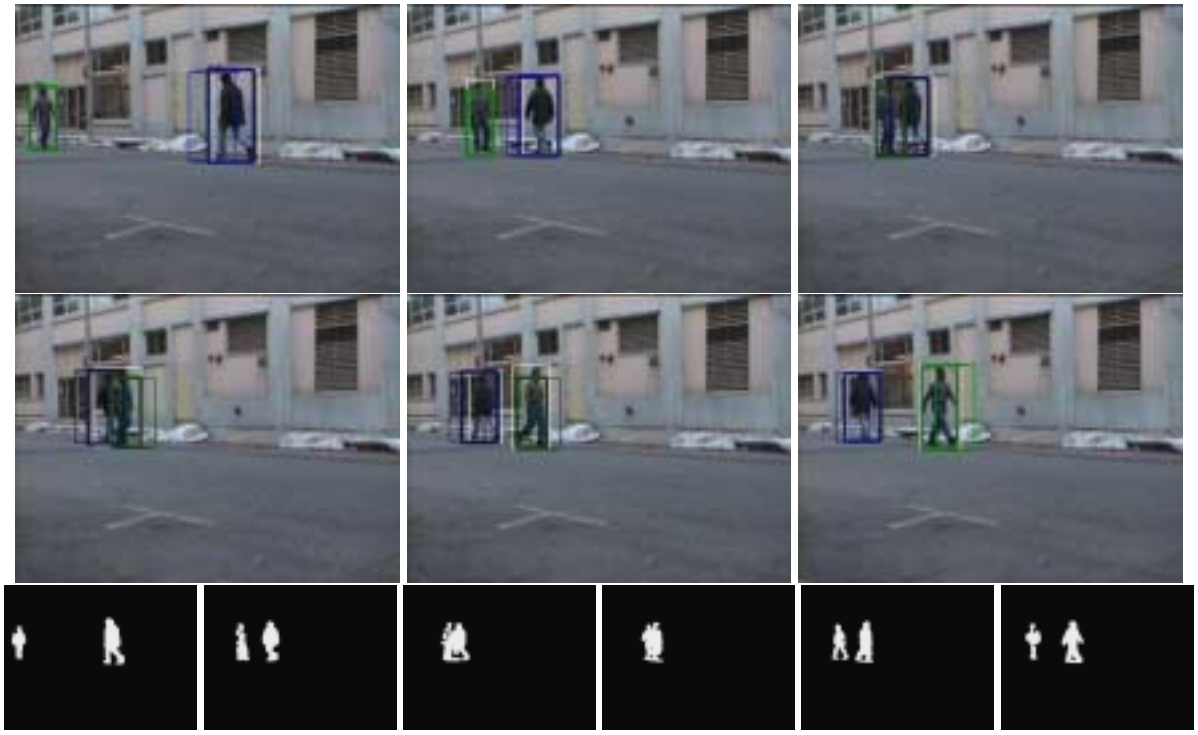


Figure 8: Tracking example: input sequence, and blob segmentation results.

8 Experimental Results

The framework was implemented and experiments were conducted on a SGI O2 R5K workstation. In some 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. In addition, we conducted sensitivity experiments with artificial sequences. Our test and data sequences have been made publicaly available at [57].

8.1 Tracking and 3D Trajectory Estimation

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. To emphasize the framework's robustness to parameter setting, all experiments use exactly the same system parameter settings. The scene learning phase took 1.5 seconds for each example (45 frames).

Results for one example test sequence are shown in Figs. 8–11. As shown in Fig. 8, the ten second sequence shows two people walking. It shows the standard behavior of the system when motions are roughly on a linear path. Note that motion trajectories are not assumed parallel to the camera's imaging plane. This results in nonlinear image trajectories due to perspective projection effects (under these conditions, the

standard 2D Kalman Filter would be a bad estimator). In each frame, 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 $\frac{1}{3}$ seconds ahead based on the EKF's current estimate of the velocity, using a first order model. The white box is the measurement obtained from the low-level processes. When one object is occluded by another both objects share the same white box (but not the colored boxes). 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 object-centered coordinate frames extracted. The moving object is resampled in the object-centered frame throughout the tracking sequence, despite changes in scale and position. The estimated bounding box is used to resample the moving blob to a locomotion-normalized view that can be used as input to motion recognition modules.

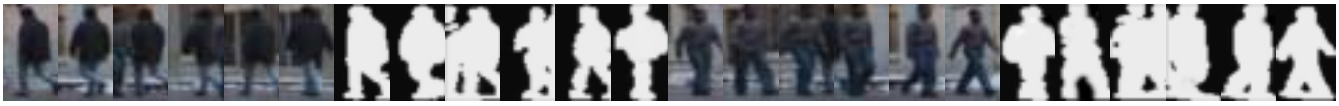


Figure 9: Normalized views of the 2 bodies in the sequence. 6 frames with their respective regions of support.

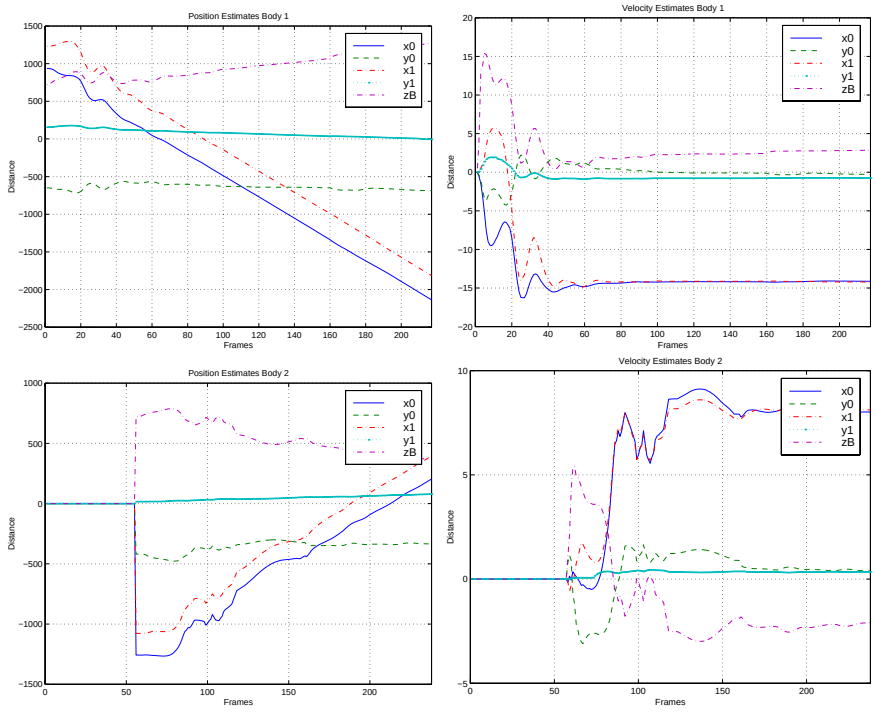


Figure 10: Estimated positions and velocities in 3D given by the EKF. The top row shows position and velocity predictions for body 1. The bottom row shows predictions for body 2. Note that body 2 appears in scene about 55 frames after body 1.

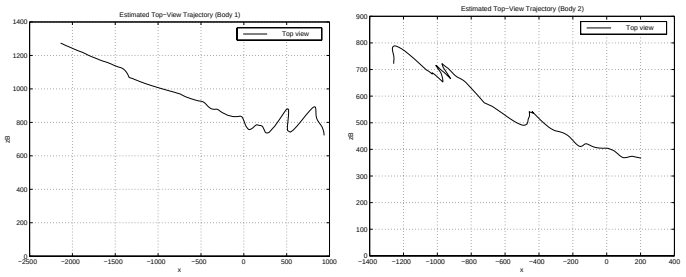


Figure 11: Recovered (top view) motion. Note that motion of body 1 is from right to left. Body 2 goes in the opposite direction. Estimates at the beginning of the sequence are not very accurate, because error covariance in the EKF is higher. Uncertainty is reduced as enough samples are given.

Fig. 10 shows the evolution in the estimates of 3D velocity and position computed via the 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. Therefore, the scale of the depth estimate is not necessarily the same scale as in the 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.

Due to space limitations, readers are referred to [56] for results on other sequences. This first set of experiments on twenty sequences was intended to test the system on real scenes 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 more interesting issue.

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 possibly lose track. This is shown in the estimates during and after occlusion in Fig. 12. Recall 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. 12 we can see the wrong prediction for the subject moving from left to right. Our model assumed that the object continued along its observed trajectory.

Speed of re-convergence depends on the current error covariance, the expected measurement noise, and the model noise. This example illustrates how even though there is a sudden change in direction, the tracker

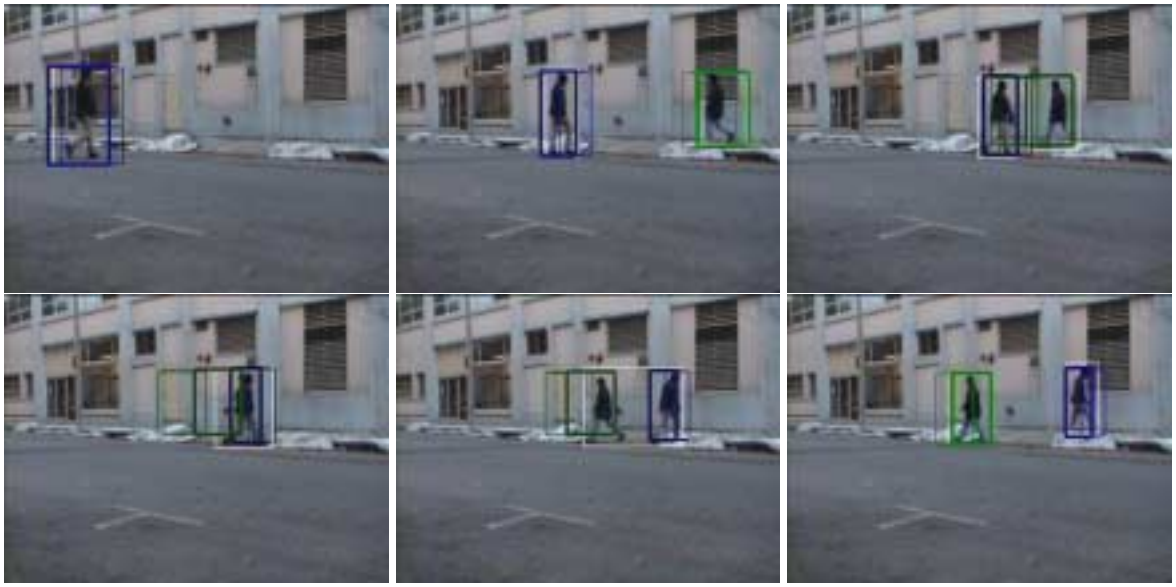


Figure 12: Tracking results, non-linear trajectories. Bodies are tracked, but prediction is less accurate during occlusion because of sudden change in direction. The accuracy in the estimate is recovered once some observations are obtained after occlusion, illustrating how the estimate can recover

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. However, in cases where trajectories change considerably during occlusions, the system will fail if the correspondence solution is not correct. A simple situation is when during occlusion, the subjects decide to go back (in the opposite direction initially taken). To correctly handle such error cases, we would need to augment our basic trajectory-based approach. To solve this problem, trajectory information is not enough. For instance, object color distribution or shape could be used to disambiguate these instances.

8.2 Learning and Recognition of Actions

In order to test the full action recognition framework, we collected sequences (three hours total) of random people performing different actions on a pedestrian pathway 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 at 45° and -45° 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 different trials. 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

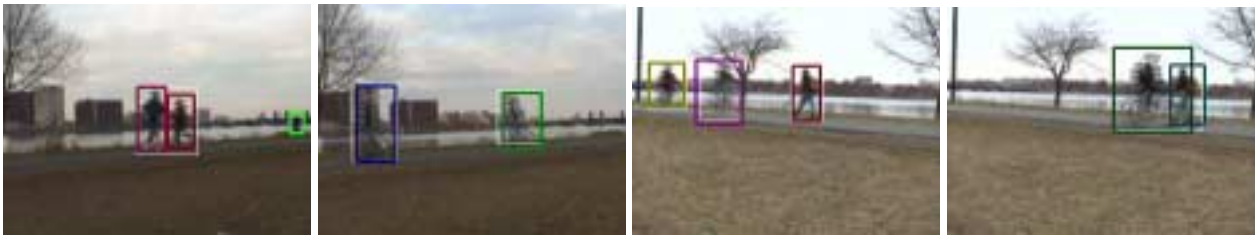


Figure 13: Example frames taken from the river sequences.

view angles used, the 56 examples where 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. 13. 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 heading directions, we reduce the total number of examples needed.

Fig. 14 shows the distribution of observed heading directions for the walking action. Fig. 14(a) shows the frontal hemisphere with respect to the camera view point, Fig. 14(b) shows the back hemisphere, Fig. 14(c) shows a top view of the sphere. Only headings with error covariance smaller than a 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 where 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_j is the probability of 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).

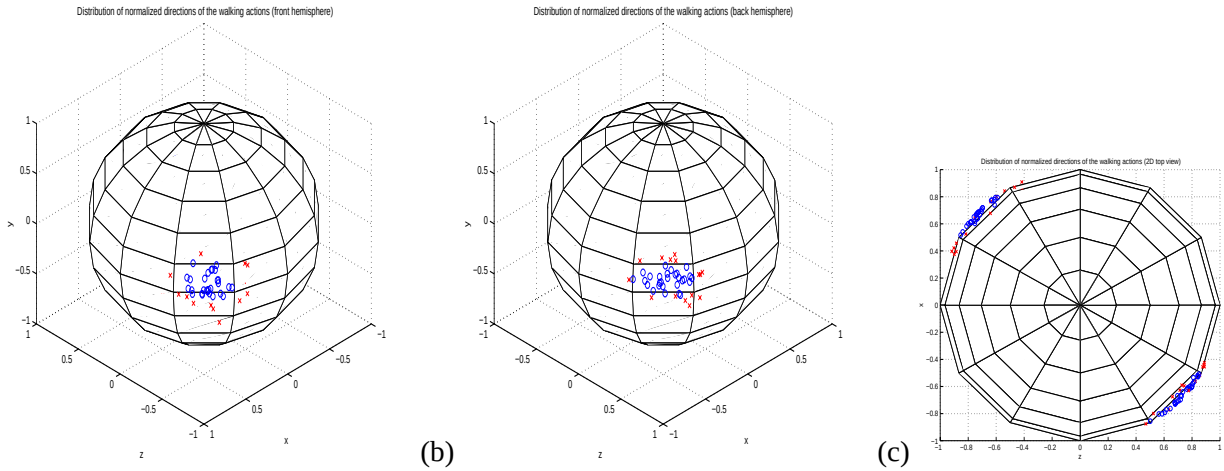


Figure 14: Distribution of normalized trajectory directions for the walking actions. Frontal view of the trajectory direction sphere (a), back view (b), and top view (c). Blue circles indicate data points falling on tessellations where the class could be statistically characterized, red cross-marks indicate data falling in other tessellations.

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).

The high confusion rate between walking and running can be attributed to the similarity of these actions given the representation employed. A straightforward way to disambiguate these two actions would be to include estimated speed as input feature for the classifier. The roller blading action also suffers from this representational similarity. Other sources of error include the limited descriptive power of Hu moments, and the low-spatial resolution. On average the size of the bounding box was approximately 50x26 pixels in a MJPEG compressed format.

8.3 Sensitivity Experiments

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. 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 $\dot{z}$. In order to provide a more comprehensive analysis of the 3D trajectory estimation technique, we tested its sensitivity with synthesized trajectory 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 unrestricted directions in 3D from a given starting position. The 3D box was then projected onto the image plane using our camera model (Eq.:3) 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 $\Delta\theta = \Delta\gamma = \pi/24$. Therefore, all possible directions of motions were tested.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: $\mathcal{E}_k = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{x}}_k^{(i)} - \mathbf{x}_k^{(i)})^2$, where $\hat{\mathbf{x}}_{\mathbf{k}}^{(\mathbf{i})}$ is our estimate of the position of feature i on the image plane at time k, $\mathbf{x}_{\mathbf{k}}^{(\mathbf{i})}$ 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.

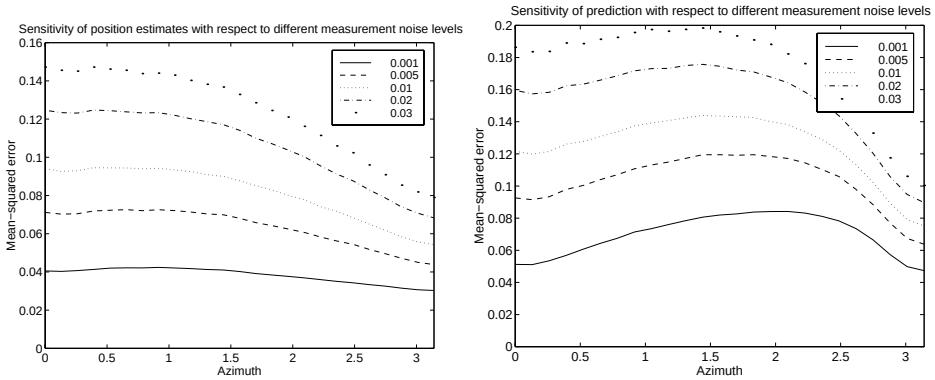


Figure 15: Graphs showing the sensitivity with respect to varying levels of measurement noise. The first graph shows the normalized mean-square error in the state estimate over various trajectory directions. The second graph shows the normalized mean-square error in the state predicted for the future frame $k + 10$.

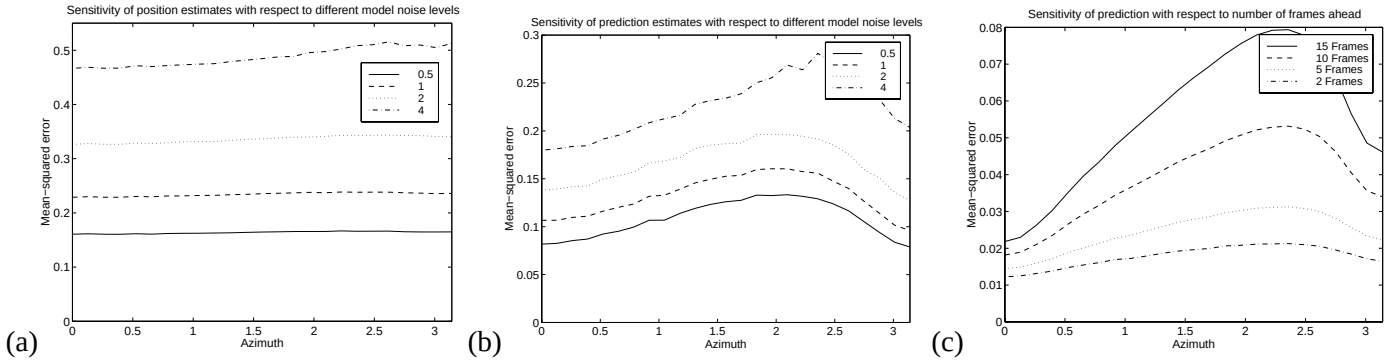


Figure 16: Graphs (a)-(b) shows sensitivity with respect to varying levels of noise in the 3D motion trajectory. Graph (a) shows the normalized mean-square error in the state estimate over various trajectory directions. Graph (b) shows the normalized mean-square error in the state predicted $k+10$. Graph (c) shows the sensitivity with respect to number of frames ahead used to calculate the prediction. Normalized mean-square error over $\theta = [0, \pi]$

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. 15 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, $\mathbf{Q} = \mathbf{I}$, $\mathbf{R} = 0.02\mathbf{I}$, 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 effects the accuracy of "look ahead" needed to foresee an occlusion, and to predict the state during occlusions.

The graph in Fig. 16(a)-(b) 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, $\mathbf{Q} = 0.5\mathbf{I}$, $\mathbf{R} = 0.01\mathbf{I}$, and $\sigma_M^2 = 0.0001$ is kept relatively small. Note that σ_S^2 is set to very high values with respect to the expected model noise. The normalized mean-square error in the position estimates is relatively constant over θ for each different level of noise and is also higher than the prediction error. The main reason for this is that the expected low measurement error tends to pull the estimates towards highly noisy measurements. 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.16(c). 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 linearization.

9 Analysis of Assumptions and Limitations

Despite our efforts to address limitations of previous systems, some important limitations and assumptions remain. We now briefly explain the limitations of the approach presented, clarifying our assumptions and relating them with possible failures of the system.

Our EKF formulation assumes a planar rectangular box moving along a linear 3D trajectory at constant velocity. Changes in direction and velocity are modeled with a Gaussian noise model. Furthermore, changes in bounding box size are assumed due to projective effects; other changes in bounding box size are modeled as Gaussian noise. In practice, the bounding box is not totally rigid (*e.g.,*due to arm swinging, leg extension, etc). Obviously, the formulation performs best when the observed motion matches the EKF model. However, no model intends to explain all aspects of the problem, therefore we should ask, how sensitive is the model performance under the imperfect world it tries to characterize? As explained in Sec. 5.3, the EKF formulation provides good ways to deal with violations of these basic assumptions, and the sensitivity of the formulation evaluated favorably in experiments described in Sec. 8.3.

In our system, it is assumed that the object's trajectory is locally linear or the direction varies slowly. In this aspect, our approach is most sensitive when, during occlusion, one of the occluded objects violates the “locally linear plus noise” velocity model. When the dynamical model is violated but the object can

be observed, the EKF can adjust (Sec. 5.3) to the newly observed direction (*e.g.*,when turning). During occlusion, the object cannot be observed so there is no data to adjust the estate parameters. This is what happens in Fig. 12. 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. After the object is *visible*, the estimate converges to the right value. In general, however, object color distributions could be used to improve correspondence after an occlusion.

Our HGR framework uses the current heading direction to index into appropriate bin on the direction sphere. This heading estimate is based upon the EKF, and therefore subject to the EKF's motion model assumptions. Furthermore, HGR assumes that the heading is approximately constant over the motion template's time interval. Therefore, the system performance would degrade if this assumption is severely violated. Action recognition when we allow the orientation of the person to vary with no constraint **and** this orientation is independent from the action, is a considerably more difficult problem, not addressed here. However, multiple actions performed at different headings by the same person at different time intervals can be recognized.

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 partly because the trajectory estimate is not very reliable when the object first appears in the scene. 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 EKF state error estimate. The use of other features, like color, motion, or shape, would be a natural extension to provide a more robust tracking.

Segmentation of different objects when they have not been tracked before is accomplished mainly on the basis of connectivity. If new objects appear in the image within the boundary of other moving objects, then they may be incorrectly merged into one object. One possible method for disambiguating such a situation is to employ motion features; however, this will not address the case where multiple objects move at the same rate.

Finally, motion template representation demonstrated in our framework (moments of MHIs/MEIs) does not discriminate reliably between actions such as running and walking. Unfortunately these are two of the most common actions in human locomotion. We have chosen the MHI/MEI action representation to demonstrate our approach; however, it should be made clear that other appearance models could be employed (*e.g.*,, [7, 48]). In addition, information about velocity and 3D trajectory obtained via the EKF could be used to gain improved discrimination between certain actions (*e.g.*,, running and walking).

10 Conclusion

It is important to create theoretically sound models that are also computationally practical. Often in machine vision, there is a trade-off between these competing goals. In this framework, the goal was to find a balance. We employ approximate models for segmentation, 3D trajectory estimation, correspondence (tracking), and recognition, that are at the same time computationally *efficient*.

We have shown how to integrate low-level and mid-level mechanisms to achieve more robust and general tracking. With the proposed extended Kalman filter formulation, 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.

Recovery of trajectories of non-rigid objects typically has been approached assuming no structure (*e.g.*,tracking subjects seen as points) or using domain constraints (*e.g.*,ground plane is known, or collapsing one dimension by using a top-view of the scene, or motion is parallel to the camera's imaging plane). 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.

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) [10, 8]. 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.

We utilized 3D trajectory estimates in a new framework: Heading-Guided Recognition (HGR). This general method significantly reduces the complexity of action classification, and could be used with other techniques (*e.g.*,, [7, 48]). Our tracking approach allows the construction of an object-centered representation. The resulting locomotion-stabilized images are then fed to the HGR action recognition module that selects the appropriate classifier based on instantaneous 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 subsystem were also tested using synthetic data sequences.

The motivation for using a mixture model as class conditional distribution 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. We believe that our framework's classification performance is encouraging considering the complexity of the task, low video resolution and quality, as well as the unrestricted environment where training and testing data were acquired.

Appendix: Mixture of Gaussians Formulation

Each action class is modeled as a mixture of Gaussians. This, in turn, can be parameterized as $\Theta = (\mu, \mathbf{\Sigma}, \mathbf{\Lambda})$. $\mu = (\mu_1..\mu_m)$ is the set of mean vectors, $\mathbf{\Sigma} = (\Sigma_1...\Sigma_m)$ is the set of covariance matrices, in our implementation, we use a diagonal $\Sigma_k = (\sigma_{ij})_k$ with $i, j = 1..M$ (M was chosen to be 5 from PCA results), $\Lambda = (\lambda_1, .., \lambda_m)^T$ 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):

$$Q(\Theta|\Theta^{(P)}) = \sum_{i=1}^{n}\sum_{k=1}^{m} z_{ik}(\log\lambda_k - \log 2\pi - \log(\prod_{l=1}^{d}\sigma_{ll}) - \frac{1}{2}\sum_{l=1}^{d}(y_{il} - \mu_{ki})^2\sigma_{ll}^2) \tag{15}$$

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 [18]:

$$z_{ik} = \frac{P(\phi_i|Z_i = k, \Theta^{(P)})P(Z_i = k|\Theta^{(P)})}{\sum_{j=1}^{n} P(Z_i = j|\Theta^{(P)})P(\phi_i|Z_i = j, \Theta^{(P)})}. \tag{16}$$

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

$$\mu_k^{(P+1)} = \frac{\sum_{i=1}^{n} z_{ik}\phi_i}{\sum_{i=1}^{n} z_{ik}}\ ; \quad \sigma_{k_{ll}}^{2^{(P+1)}} = \frac{\sum_{i=1}^{n} z_{ik}(\phi_{il} - \mu_{kl})^2}{\sum_{i=1}^{n} z_{ik}}\ ; \quad \lambda_k^{(P+1)} = \frac{\sum_{i=1}^{n} z_{ik}}{\sum_{l=1}^{m}\sum_{i=1}^{n} z_{il}} \tag{17}$$

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. For each class, we have :

$$\text{MDL} : \arg\max_{\Theta,k}(\log p(\mathbf{\Phi}|\Theta) - \frac{k}{2}\log n), \tag{18}$$

where Φ represents all training vectors for the given class, 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 [55], where we provide a comparative study of some possible probability models for use in human action recognition.

References

[1] A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal lenght. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6), 1995.

[2] C. Barron and I. Kakadiaris. Estimating anthropometry and pose from a single uncalibrated image. *Computer Vision and Image Understanding*, 81(3), 2001.

[3] 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.

[4] A Baumberg and D Hogg. Learning flexible models from image sequences. In *Proc. European Conference on Computer Vision*, volume 1, pages 299–308, May 1994.

[5] M. Bichsel. Segmenting simply connected moving objects in a static scene. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16 (11):1138-1142, 1994.

[6] M. Black and A. Jepson. A probabilitic framework for matching temporal trajectories: Condensation-based recognition of gestures and expressions. In *Proc. European Conference on Computer Vision*, 1998.

[7] M. Black and Y Yacoob. Tracking and recognizing rigid and non-rigid facial motion using local parametric models of image motion. In *Proc. InternationalConference on Computer Vision*, 1995.

[8] H. Blom. An efficient filter for abruptly changing systems. In *Proc. Conference on Decision Control*, 1984.

[9] A. Bobick and J. Davis. The representation and recognition of action using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3)257-267, 2001.

[10] K. Bradshaw, I. Reid, and D. Murray. The active recovery of 3d motion trajectories and their use in prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3), 1997.

[11] M. Brand. Shadow puppetry. In *Proc. InternationalConference on Computer Vision*, 1999.

[12] C. Bregler. Tracking people with twists and exponential maps. In *Proc. Computer Vision and Pattern Recognition*, 1998.

[13] T. Broida and R. Chellappa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):497-513, 1991.

[14] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *Proc. Computer Vision and Pattern Recognition*, 2000.

[15] T. Darrell and A. Pentland. Classifying hand gestures with a view-based distributed representation. In *Neural Information Processing Systems 8*, 1995.

[16] J. Davis. Human motion analysis using hierarchical motion history images. *IEEE Workshop on Detection and Recognition of Events in Video*, 23(3)257-267, 2001.

[17] Q. Delamarre and O. Faugeras. 3d articulated models and multiview tracking with physical forces. *Computer Vision and Image Understanding*, 81(3), 2001.

[18] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data. *Journal of the Royal Statistical Society*, 39(1), 1977.

[19] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Proc. Computer Vision and Pattern Recognition*, 2000.

[20] S. Fejes and L. Davis. What can projections of flow fields tell us about the visual motion. In *Proc. InternationalConference on Computer Vision*, 1998.

[21] H. Fujiyoshi and A. Lipton. Real-time human motion analysis by image skeletonization. In *Workshop on Applications of Computer Vision*, 1998.

[22] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1972.

[23] A. Galata, N. Johnson, and D. Hogg. Learning variable-length markov models of behavior. *Computer Vision and Image Understanding*, 81(3), 2001.

[24] D. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1), 1999.

[25] D. Gavrila and L. Davis. Tracking of humans in action: a 3-d model-based approac. In *Proc. ARPA Image Understanding Workshop, Palm Springs*, 1996.

[26] W.E.L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proc. Computer Vision and Pattern Recognition*, 1998.

[27] I. Haritaouglu, R. Cutler, D. Harwood, and L.S. Davis. Backpack: Detection of people carrying objects using silhouettes. *Computer Vision and Image Understanding*, 81(3), 2001.

[28] N. Howe, M. Leventon, and B. Freeman. Bayesian reconstruction of 3d human motion from single-camera video. In *Neural Information Processing Systems 12*, 2000.

[29] L. Davis I. Haritaouglu, D. Harwood. W4s: A realtime system for detecting and tracking people in 2.5d. In *Proc. European Conference on Computer Vision*, 1998.

[30] S. Intille and A. Bobick. Recognizing planned, multiperson action. *Computer Vision and Image Understanding*, 81(3), 2001.

[31] S. Intille and A. F. Bobick. Real time close world tracking. In *Proc. Computer Vision and Pattern Recognition*, 1997.

[32] M. Irani and P. Anandan. A unified approach to moving object detection in 2d and 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6), 1998.

[33] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1): 5-28, 1998.

[34] M. Isard and A. Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proc. European Conference on Computer Vision*, 1998.

[35] N. Johnson and D. C. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14:609–615, 1996.

[36] S. Ju, M. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Proc. International Conference on Automatic Face and Gesture Recognition*, 1996.

[37] I. Kakadiaris, D. Metaxas, and R. Bajcsy. Active part-decomposition, shape and motion estimation of articulated objects: A physics-based approach. In *Proc. Computer Vision and Pattern Recognition*, 1994.

[38] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59-69, 1982.

[39] P. Kornprobst and G. Mendioni. Tracking segmented objects using tensor voting. In *Proc. Computer Vision and Pattern Recognition*, 2000.

[40] M.W. Krueger. *Virtual Reality II*. Addison-Wesley, 1990.

[41] D. Metaxas and D. Terzopoulos. Shape and non-rigid motion estimation through physics-based synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580-591, 1993.

[42] T. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3), 2001.

[43] D. Morris and J. Rehg. Singularity analysis for articulated object tracking. In *Proc. Computer Vision and Pattern Recognition*, 1998.

[44] R. Nelson. Qualitative detection of motion by a moving observer. *International Journal of Computer Vision*, 7(1), 1991.

[45] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Special Issue Video Surveillance, 2001.

[46] A. Pentland and B. Horowitz. Recovery of non-rigid motion and structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):730–742, 1991.

[47] R. Plankers and P. Fua. Tracking and modeling people in video sequences. *Computer Vision and Image Understanding*, 81(3), 2001.

[48] R. Polana and R. Nelson. Low level recognition of human motion. In *Proc. IEEE Workshop on Nonrigid and Articulate Motion*, 1994.

[49] B. Rao, H. Durrant-Whyte, , and J. Sheen. A fully decentralized multi-sensor system for tracking and surveillance. *International Journal of Robotics Research*, 12(1), 1993.

[50] C. Rao and M. Shah. A view-invariant representation of human action. In *International Conference on Control, Automation, Robotics and Vision,ICARCV*, 2000.

[51] J. M. Regh and T. Kanade. Model-based tracking of self-occluding articulated objects. In *Proc. InternationalConference on Computer Vision*, 1995.

[52] P. Remagnino, A. Baumberg, T. Grove, D. Hogg, T. Tan, A. Worrall, and K. Baker. An intergrated traffic and pedestrian model-bassed vision system. In *British Machine Vision Conference*, 1994.

[53] D. Reynard, A. Wildenberg, A. Blake, and J. Marchant. Learning dynamics of complex motions from image sequences. In *Proc. European Conference on Computer Vision*, 1996.

[54] K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image Understanding*, 59(1):94-115, 1994.

[55] R. Rosales. Recognition of human action using moment-based features. Technical Report BU 98-020, Boston University, 1998.

[56] R. Rosales. A collection of results on tracking people in real scenes. In *http://cs-people.bu.edu/rrosales/research/IMHT/*, 2000.

[57] R. Rosales. Street and charles river people sequences. In *http://cs-people.bu.edu/rrosales/SCRSeqs/*, 2000.

[58] 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.

[59] R. Rosales and S. Sclaroff. Learning body pose using specialized maps. In *Neural Information Processing Systems 14*, 2001.

[60] R. Rosales and Stan Sclaroff. Inferring body pose without tracking body parts. In *Proc. Computer Vision and Pattern Recognition*, 2000.

[61] M. Shah and R. Jain. *Motion-Based Recognition*. Kluwer Academic, 1997.

[62] H. Siddenbladh, M. Black, and D. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Proc. European Conference on Computer Vision*, 2000.

[63] Y. Song, L. Goncalves, E. DiBernardo, and P. Perona. Monocular perception of biological motion in johansson displays. *Computer Vision and Image Understanding*, 81(3), 2001.

[64] H.W. Sorenson. Least-squares estimation: From gauss to kalman. *IEEE Spectrum*, Vol. 7, pp. 63-68, 1970.

[65] C. Stauffer and W.E.L. Grimson and. Adaptive background mixture models for real-time tracking. In *Proc. Computer Vision and Pattern Recognition*, 1999.

[66] R. Szeliski and S. Bing Kang. Recovering 3d shape and motion from image streams using non-linear least squares. In *Proc. Computer Vision and Pattern Recognition*, 1993.

[67] G. Welch and G. Bishop. An introduction to the kalman filter,. Technical Report TR 95-041, Computer Science, UNC Chapel Hill, 1995.

[68] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real time tracking of the human body. Technical Report TR 353, MIT Media Lab, 1996.

[69] M. Yamamoto and et. al. Incremental tracking of human actions from multiple views. In *Proc. Computer Vision and Pattern Recognition*, 1998.

[70] Z. Zhang and O. Faugeras. *3D Dynamic Scene Analysis: A Stereo Based Approach*. Springer, Berlin, 1992.