BU CS TR98-019 rev. 2. To appear in **Proc. IEEE Conf. on Computer Vision and Pattern Recognition**, June 1999*

3D Trajectory Recovery for Tracking Multiple Objects and Trajectory Guided Recognition of Actions

Rómer Rosales and Stan Sclaroff
Computer Science Department, Boston University

Abstract

A mechanism is proposed that integrates low-level (image processing), mid-level (recursive 3D trajectory estimation), and high-level (action recognition) processes. It is assumed that the system observes multiple moving objects via a single, uncalibrated video camera. 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 utlized in extracting stabilized views of the moving object. 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 Gaussian classifier. The system was tested in recognizing various dynamic human outdoor activities; e.g., running, walking, roller blading, and cycling. Experiments with synthetic data sets are used to evaluate stability of the trajectory estimator with respect to noise.

1 Introduction

Tracking non-rigid objects and classifying their motion is a challenging problem. The importance of tracking and motion recognition problems is evidenced by the increasing attention they have received in recent years [26]. Effective solutions to these problems would lead to breakthroughs in areas such as video surveillance, motion analysis, virtual reality interfaces, robot navigation and recognition.

Low-level image processing methods have been shown to work surprisingly well in restricted domains despite the lack of high-level models [9, 8, 30]. 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 [6, 12, 15, 16, 18, 21, 23], such methods typically require careful placement of the initial model.

These limitations arise because object tracking, 3D 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 3D 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 to handle multiple moving objects, 2.) to explicitly model occlusion, 3.) to estimate and predict 3D motion trajectories, and 4.) to recognize nonrigid motions. An improved feedback mechanism is proposed that combines low-level (image segmentation) and mid-level (recursive trajectory estimation), and high-level (action recognition) modules. 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 multiple moving objects before, during, and after occlusion. The 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. The system has been tested in recognizing various dynamic human outdoor activities; *e.g.*, running, walking, roller blading, and cycling. Finally, the system's noise stability properties have been evaluated using synthetic data sets, and results are encouraging.

2 Related Work

The extended Kalman filter (EKF) has proven to be very useful in recovery of rigid motion and structure from image sequences [7, 1, 5, 22, 20, 24]. Most of these approaches assume rigid motion. One of the first important results on recursive structure and motion estimation was the work of [7]. 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, and use this information to estimate 3D translational trajectories (up to a scale factor).

In order to model trajectories, [5] assumed that the surface on which the motions occur was known, and also that this surface was a plane. Each objects was represented as a point moving in the plane, partially avoiding problems related to changes in shape. It is also possible to reduc-

ing tracking to a plane, if the projective effect is avoided through the use of a top, distant view [14]. It is also possible to use some heuristics about body part relations and motion on image plane like [13]. In our work, we do not assume planar motion or detailed knowledge about the object and our formulation can handle some changes in shape.

More complicated representations like [16, 12, 18, 23, 15, 21], use model-based techniques, generally articulated models comprised of 2D or 3D solid primitive, sometimes accounting for self-occlusion by having an explicit object model. Shape models were used by [2] for human tracking.

The most relevant motion recognition approaches related to our work are those that employ view-based models [3, 9, 19]. In particular, [9] uses motion history images (MHI) and motion energy images (MEI), temporal templates that are matched using a nearest neighbor approach against examples of given motions already learned. The main problem of this method is the requirement of having stationary objects, and the insufficiency of the representation to discriminate among similar motions. Motion analysis techniques have had the problem that registration of useful, filtered information is a hard labor by itself [9, 4, 19]. Our system provides a functional front-end that supports such tasks.

3 Basic Approach

A diagram of our approach is illustrated in Fig. 1. The first stage of the algorithm is based on the background subtraction methods of [30]. 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. The idea is to have a confidence map 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. The input video stream is low-pass filtered to reduce noise effects. A connected components analysis is then applied to the resulting image. Initial segmentation is usually noisy, so morphological operations and size filters are applied.

If there are occlusions, or strong similarities between the empty scene model and the objects, then it is possible that regions belonging to the same object may be classified as part of different objects or vice versa. To solve this problem, two sources of information are used: temporal analysis and trajectory prediction.

In temporal analysis, 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.

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.

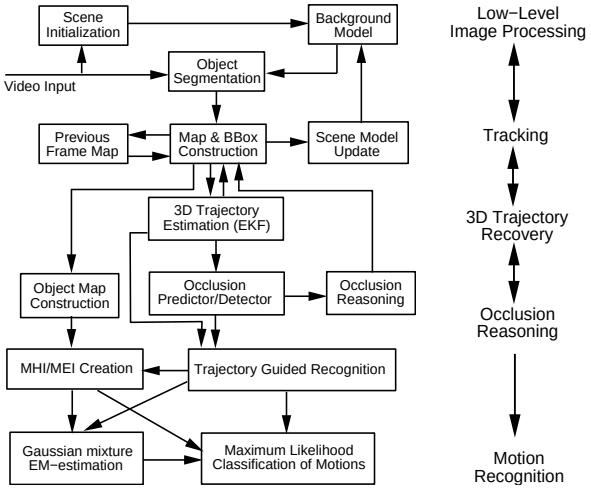

Figure 1: System Diagram.

Occlusion prediction is performed based on the current EKF estimates. Given that we know 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.

For each frame, 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. This yields a stabilized of the moving object throughout the tracking sequence, despite changes in scale and position.

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) [4, 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 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 hemisphere 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 deter-

mine the direction-partitioned feature space. This allows automatic learning and adaptation of the direction space to those directions that are commonly observed.

4 3D Trajectory from 2D Image Motion

Our method requires moving blob segmentation and connected components analysis as input to the tracking module. Due to space limitations, readers are to [24] for details of these modules. To reduce the complexity of the tracking problem, two feature points are selected: two opposite corners of the blob's bounding box. Using a blob's bounding box alleviates need to searching for corresponding point features in consecutive frames. In general we think that a detailed tracking of features is neither necessary nor easily tenable for non-rigid motion tracking at low resolution.

It is 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.

For our representation a 3D central projection model similar to [28, 1] is used:

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

where $(x, y, z)^T$ is the real 3D feature location in the camera reference frame, $(u, v)^T$ 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. The model is numerically well defined even in the case of orthographic projection.

Our state models a 3D planar rectangular bounding box moving along a linear trajectory at constant velocity. 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 and the robustness of the estimates. Our state vector then becomes:

$$\mathbf{x} = (x_0, y_0, x_1, y_1, z\beta, \dot{x}_0, \dot{y}_0, \dot{x}_1, \dot{y}_1, \dot{z\beta})^T, \tag{2}$$

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 vector $(\dot{x}, \dot{y}, \dot{z\beta})^T$ represents a corner's 3D velocity relative to the camera.

The sensitivity in $\dot{x}$ and $\dot{y}$ is directly dependent on the object depth as objects that are farther away from the camera tend to project to fewer image pixels. The sensitivity of $\dot{z}$ is an inverse function of camera focal length, becoming zero in the orthographic case.

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, trajectories in the image plane are nonlinear, and 2D models are therefore inaccurate.

4.1 Extended Kalman Filter Formulation

Trajectory estimates are obtained via an extended Kalman Filter (EKF) formulation. Our state is guided by the following linear equation:

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{w}_k, \tag{3}$$

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 space 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 [22].

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. The EKF time update equation becomes:

$$\hat{\mathbf{x}}^-_{k+1} = \mathbf{A}_k \hat{\mathbf{x}}_k \tag{4}$$

$$\mathbf{P}^-_{k+1} = \mathbf{A}_k \mathbf{P}_k \mathbf{A}^T_k + \mathbf{Q}_k \tag{5}$$

where $\mathbf{Q}_k$ is the process noise covariance.

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:

$$\mathbf{K}_k = \mathbf{P}^-_k \mathbf{H}^T_k (\mathbf{H}_k \mathbf{P}^-_k \mathbf{H}^T_k + \mathbf{V}\mathbf{R}_k \mathbf{V}^T)^{-1} \tag{6}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}^-_k + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}^-_k, 0)) \tag{7}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}^-_k, \tag{8}$$

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

$$\mathbf{H}_k = \begin{bmatrix} \frac{1}{\lambda} & 0 & 0 & 0 & -\frac{x_0}{\lambda^2} \\ 0 & \frac{1}{\lambda} & 0 & 0 & -\frac{y_0}{\lambda^2} \\ 0 & 0 & \frac{1}{\lambda} & 0 & -\frac{x_1}{\lambda^2} \\ 0 & 0 & 0 & \frac{1}{\lambda} & -\frac{y_1}{\lambda^2} \end{bmatrix}, \tag{9}$$

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 $p(\mathbf{w}_k) \sim N(0, \mathbf{Q}_k)$, and $p(v_k) \sim N(0, \mathbf{V}\mathbf{R}_k\mathbf{V}^T)$. For more detail, see [27, 29].

Obviously, 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 reseting 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.

5 Motion Recognition

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 represented in terms of motion energy images (MEI's) and motion history images (MHI's) [4, 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 stabilized input sequences, it is possible to make the MEI/MHI approach invariant to unrestricted 3D translational motion.

The seven Hu moment invariants are then computed for both the MHI and MEI [4, 9]. The resulting features are combined in a 14-dimensional vector. The dimension of this vector is reduced via principal components analysis (PCA) [11]. In our experiments, the PCA allowed a dimensionality reduction of 64% (dim=5), while capturing 90% of the variance of our training data. The reduced feature vector ϕ is then fed into a maximum likelihood classifier that is based on a mixture of Gaussians model.

In the mixture model, each action class i is represented by a set of mixture model parameters, Θ_i. 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 [10]. Given $P(\phi|\Theta_i)$, we calculate $P(\Theta_i|\phi)$ using Bayes rule.

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. 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:

$$MDL: \arg\max_{\Theta_i,k}(\log P(\phi|\Theta_i) - \frac{k}{2}\log n), \qquad (10)$$

where k is the number of parameters in the model (defining the Gaussian mixture), and n is the number of samples in the training data. For further details about the model estimation module, see [25].

5.1 Trajectory-Guided Recognition

In theory it is necessary to learn representations of every action from all possible directions. We denote $\mathcal{P}_j$ to be the set of PDF's used to represent m actions under direction j. Each action i has its own PDF, $P(\Theta_i^{(j)}|\phi_k) \in \mathcal{P}_j$. Acquiring 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 TGR, we partition the direction hemisphere based on the trajectories estimated in the training data. Each partition j corresponds to a group of similar trajectory directions. The PDF's can then be automatically estimated for each bin in the trajectory direction space. Therefore, we only need to use a single camera viewpoint and sample the space based on the estimates of trajectories.

This approach has the following advantages. In many practical situations, not all trajectories are observed, so the 3D space can be divided into fewer areas (*e.g.*,see Sec.6.1). 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. Finally, it is possible to adapt the partioning of the direction space based on distribution of data [17]. For a complete description of the TGR approach, readers are directed to [25].

TGR is performed at every time step k as follows:

For each moving object l

Compute EKF trajectory estimate $\mathbf{x}_k$ and error covariance $\mathbf{P}_k$

If $Trace(\mathbf{P}_k) < \epsilon$ and object l is not occluded then

1. Compute stabilized MEI and MHI
2. Compute PCA feature vector ϕ_k
3. Find $\mathcal{P}_j$ whose orientation is closest to $\mathbf{x}_k$.
4. Find $i(\phi_k) = \arg\max_i(P(\Theta_i^{(j)}|\phi_k))$, $P(\Theta_i^{(j)}|\phi) \in \mathcal{P}_j$

The certainty threshold ϵ is set depending on how much accuracy is required in the trajectory estimate $\mathbf{x_k}$ in order to perform recognition/learning. Note that the process for learning is basically the same, except step 4.) is replaced with: "Use ϕ_k in estimating $P(\Theta_i^{(j)}|\phi) \in \mathcal{P}_j$."

6 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). In order to test the system, we collected a number of sequences of people moving and occluding each other in public environments.

The first example is meant to demonstrate the basic approach. The image sequence shown in Fig. 2, consists of frames taken from a ten second multiple body walking sequence. The estimated bounding boxes are shown drawn on each image. It shows the standard behavior of the system when motions are roughly on a linear path. Note that the motion trajectories are not parallel to the camera plane. This causes non-linear trajectories in the image. 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. 3 shows the normalized coordinate frames extracted. The moving object is resampled in a stabilized view throughout the tracking sequence, despite changes in scale and position. This estimated bounding box is used as input to motion recognition modules.

Finally, given the information recovered, it is possible to construct a top-view map, showing the trajectories of the bodies Fig. 4. Recall that depth is estimated up to a scale factor. The first body moves from right to left, while

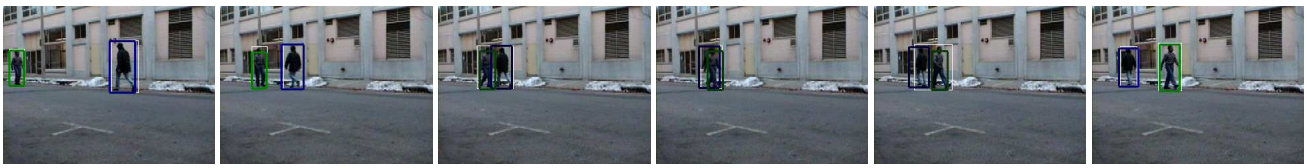

Figure 2: Tracking example: 2 bodies walking in different trajectories, occluding each other.

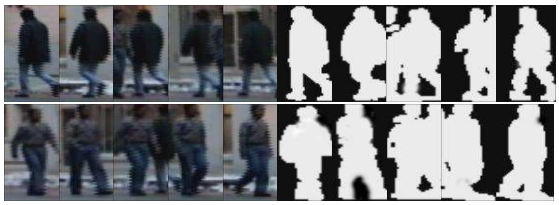

Figure 3: Normalized views of the 2 bodies in the sequence, one row per body. 6 views with their respective regions of support.

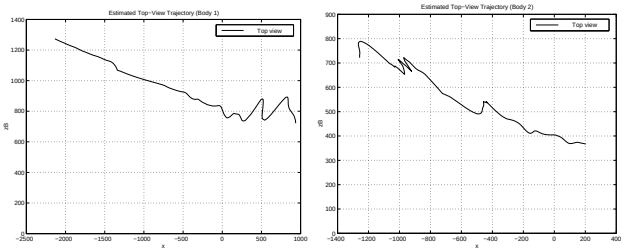

Figure 4: Recovered (top view) motion. Note that motion of body 1 is from right to left. Body 2 goes in the opposite direction. Estimates at the beginning of the sequence are not very accurate, because error covariance in the EKF is higher. Uncertainty is reduced as enough samples are given.

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 trajectories.

6.1 Learning and Recognition of Actions

In order to test our the full action recognition system, we collected sequences (3 hours total) of random people performing different actions on a pedestrian road in a 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.

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).

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

Example frames from the data set are shown in Fig. 5. As before, the estimated bounding boxes for each moving object are shown overlaid on the input video image. Our approach indicated that only two trajectories where mainly

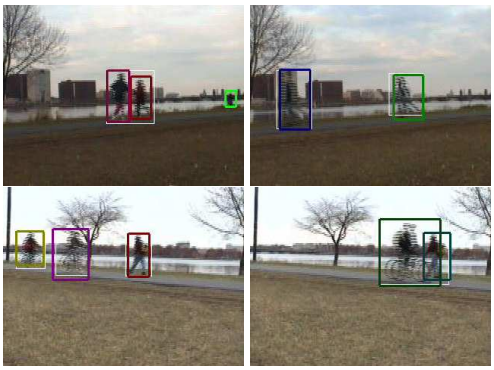

Figure 5: Example frames taken from the river sequences.

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).

observed: either direction along the foot path. Therefore, the system learned just two sets of $m = 4$ PDF's ($(\mathcal{P})_1 and (\mathcal{P})_2$). Results for either view were almost the same; average rates are therefore presented.

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 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).

6.2 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 directions in 3D from a given starting position. The 3D box was then projected

onto the image plane using our camera model (Eq.:1) with unit focal length. Each synthetic image sequence was 100 frames long. The set of directions was sampled by the azimuth θ and the rotation γ around the vector corresponding to $\theta = 0$. For sampling we use $\Delta\theta = \Delta\gamma = \pi/24$ ($12 \times 48 = 576$ different directions). For each experiment, each of the 576 possible trajectory directions was tested using 15 sequences with randomly perturbed inputs.

All synthetic video sequences were sampled at a pixel resolution of 512×512. This was mapped to a physical viewport 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. 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, 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.

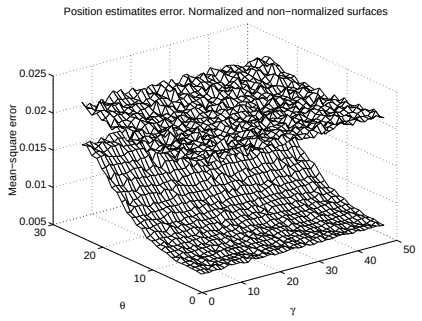

Figure 6: Example of performance with significant measurement noise ($\sigma_M^2 = 1.5, \sigma_S^2 = 0.001$). The plots show error as a function trajectory direction (γ, θ). Normalized mean-square error (upper surface) and non-normalized mean-square erro r error surfaces (lower surface) are shown. Note that mean-square error varies almost exclusively with θ (azimuth).

To test the sensitivity of the system to noise in the measurements, time varying 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, time varying 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. An illustrative example, Fig. 6 shows error surfaces for $\sigma_M^2 = 1.5$ and $\sigma_S^2 = 0.001$. The plot shows the mean-square error and normalized mean-square error , over all possible directions. As can be seen,

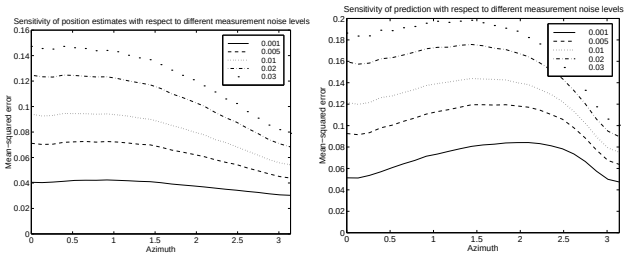

Figure 7: Graphs showing the sensitivity with respect to varying levels of measurement noise. The first graph shows the normalized mean-square error in the state estimate over various trajectory directions. The second graph shows the normalized mean-square error in the state predicted for the future frame $k + 10$.

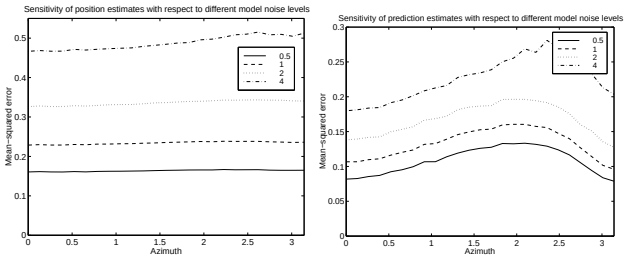

Figure 8: Sensitivity with respect to varying levels of noise in the 3D motion trajectory. The first graph shows the normalized mean-square error in the state estimate over various trajectory directions. The second graph shows the normalized mean-square error in the state predicted $k + 10$.

mean-square error is relatively invariant to γ. Due to this result, our graphs drop γ by averaging over it, so that the complexity of visualization is reduced.

Fig. 7 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. 8 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

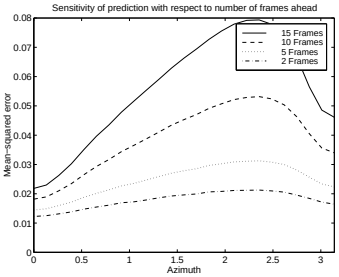

Figure 9: Sensitivity with number of frames ahead used to calculate the prediction. Normalized mean-square error over $\theta = [0, \pi]$

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 the trajectory prediction, at varying Δk frames into the future. Results are shown in Fig.9. 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.

7 Conclusion

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.

We utilized EKF 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.*,, [3, 19]). 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 subsystem 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.

References

[1]A. Azarbayejani and A. Pentland. Recursive estimation of motion, structure, and focal lenght. *PAMI*, 17(6), 1995.

[2]A. Baumberg and D. Hogg. Learning flexible models from image sequences. *ECCV*, 1994.

[3]M. Black and Y Yacoob. Tracking and recognizing rigid and non-rigid facial motion using local parametric models of image motion. *ICCV*, 1995.

[4]A. Bobick and J Davis. An appearance-based representation of action. *ICPR*, 1996.

[5]K. Bradshaw, I. Reid, and D. Murray. The active recovery of 3d motion trajectories and their use in prediction. *PAMI*, 19(3), 1997.

[6]C. Bregler. Tracking people with twists and exponential maps. *CVPR98*, 1998.

[7]R. Chellappa T. Broida. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *PAMI*, 13(6):497-513, 1991.

[8]T. Darrell and A. Pentland. Classifying hand gestures with a view-based distributed representation. *NIPS*, 1994.

[9]J. Davis and A. F. Bobick. The representation and recognition of human movement using temporal templates. *CVPR*, 1997.

[10]A. Dempster, N. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data. *J. of the Royal Stat. Soc. (B)*, 39(1), 1977.

[11]K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1972.

[12]D. Gavrila and L. Davis. Tracking of humans in action: a 3-d model-based approac. *Proc. ARPA IUE Workshop*, 1996.

[13]L. Davis I. Haritaouglu, D. Harwood. W4s: A realtime system for detecting and tracking people in 2.5d. *ECCV*, 1998.

[14]S. Intille and A. F. Bobick. Real time close world tracking. *CVPR*, 1997.

[15]S. Ju, M. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. *Proc. Gesture Recognition*, 1996.

[16]I. Kakadiaris, D. Metaxas, and R. Bajcsy. Active part-decomposition, shape and motion estimation of articulated objects: A physics-based approach. *CVPR*, 1994.

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

[18]A. Pentland and B. Horowitz. Recovery of non-rigid motion and structure. *PAMI*, 13(7):730–742, 1991.

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

[20]B. Rao, H. Durrant-Whyte, , and J. Sheen. A fully decentralized multi-sensor system for tracking and surveillance. *IJRR*, 12(1), 1993.

[21]J. M. Regh and T. Kanade. Model-based tracking of self-occluding articulated objects. *ICCV*, 1995.

[22]D. Reynard, A. Wildenberg, A. Blake, and J. Marchant. Learning dynamics of complex motions from image sequences. *ECCV*, 1996.

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

[24]R. Rosales and S. Sclaroff. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. *IEEE CVPR Workshop on the Interp. of Visual Motion*, 1998.

[25]R. Rosales and S. Sclaroff. Trajectory guided tracking and recognition actions. TR BU-CS- 99-002, Boston U., 1999.

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

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

[28]R. Szeliski and S. Kang. Recovering 3d shape and motion from image streams using non-linear least squares. *CVPR*, 1993.

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

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