To appear in *Proc. Workshop on Robust Vision for Vision-Based Control of Motion* (in association with 1998 IEEE International Conference on Robotics and Automation) Leuven, Belgium

Integrated Object Models for Robust Visual Tracking

Kevin Nickels
knickels@uiuc.edu

Seth Hutchinson
seth@uiuc.edu

Dept. of Electrical and Computer Engineering and The Beckman Institute
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

The robustness of visual tracking, or following the movement of objects in images, can be improved with an explicit model for the objects being tracked. In this paper, we investigate the use of an object model in this way. An object geometric model can tell us what feature movements to expect, and what those movements reveal about object motion. We characterize the tracking problem as one of parameter estimation from incomplete feature tracking data, and apply the Extended Kalman Filtering algorithm to the situation. Having an object model integrated into the tracking system overconstrains feature trackers, so that erroneous tracking results are selectively ignored and feasible tracking results are used to optimally update the object configuration estimate.

1 Introduction

Visual object tracking is the following of the movements of objects in a scene by analysis of computer images of that scene. The tracking of features on an object is a necessary precursor to many tasks in vision-based control, such as positioning the gripper in a predefined orientation with respect to a known object in an unknown configuration in the workspace. Object tracking is helpful when the shape and geometry of an object are known, but the current configuration is not known. This is particularly useful when kinematically complex objects, such as those with internal degrees of freedom, are involved.

In object tracking, information about the movement of individual features in images is used to update the *state* of the tracked object. The state is defined to be the position, orientation, and any internal degrees of freedom of the object. In feature tracking, each feature is assigned a state (here, a state would include size, orientation, and position of the *feature*). One advantage of object tracking is that the overconstrained nature of the problem may be used to reject faulty feature tracking results. Hashimoto, *et al* [11] have shown that increasing the number of redundant features increases the robustness of visual servoing.

The problem of determining object state from image feature locations has been extensively studied. The earliest approach to this problem was direct analysis of the transformation from object state (typically, position and orientation of a rigid object) to image feature locations, and study of the relationships involved. Nagel gives a review of the early work in [26]. This type of analytical work typically assumed that the 2D image feature measurements were given [7] [24]. More recent work in this area has focused on the analysis of uncertainty and the propagation of error through this transformation [19] [20] [13] [8]. In the 1980's, researchers began using an algorithm called Kalman Filtering to estimate object state by modeling the system structure. Kalman Filtering is an optimal state estimation process applied to a dynamic system [5], and is described in Section 5. Gennery [9] uses an algorithm very similar to the Kalman Filter to estimate the position and rotation of a rigid object. In [3], Broida and Chellappa employ a Kalman Filter to estimate the 2D rotation and 1D translation of an object from a sequence of 1D images. Dickmanns and Graffe [6] presented a seminal work describing the relationship of the modeled world (object state) to the real world and measured world (feature measurements), and describing the use of the framework in guiding autonomous vehicles. Since then, this dynamic machine vision approach has been used in many situations, to estimate the depth of points in a scene [22] [32], to estimate the state of objects in the world [2] [39], and for visual servoing [31] [36] [15] [37].

In addition to its use as an algorithm for incorporating feature measurements into a state estimate, many researchers have benefited from the use of the Kalman Filter as a purely spatio-temporal filter [25] [21] [34] [40]. In this way, spurious measurements and noisy data can be handled in a numerically robust manner. The filtered measurements can then be used in an analysis of the scene to generate more robust estimates for object states.

In this paper, we present a method for the use of object models for complex articulated objects, instantiated in an EKF framework, for the visual tracking of these objects. Furthermore, we describe how the overconstrained nature of the problem can be exploited to optimally update the object state, given the results from the tracking of the object features. We utilize explicit models for the articulated object and the imaging system to achieve this. Finally, we present some tracking results for the case where the object of interest is a three degree of freedom robotic arm.

2 Models Used

In this paper, five kinds of models will be used: the overall system model, the object geometric model, the object appearance model, the imaging model, and the object dynamic model. In this section, we will describe these models and discuss the possible assumptions that can be made for each.

2.1 System Model

We utilize a general nonlinear model in our tracking system. The other models we will describe relate to assumptions about object geometry, appearance, and dynamics, as well as the the imaging system, and are realized by instantiating functions in this system model. In this model, the *state vector* at time $k+1$, $\mathbf{x}_{k+1}$, is given by a vector valued function $\mathbf{f}_k$ of the state vector at time k, $\mathbf{x}_k$, with additive noise. Observations of the system are given by another vector valued function $\mathbf{h}_k$ of the state vector at time k, $\mathbf{x}_k$, with additive noise. The equations for the nonlinear model used for the system are of the form

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{w}_k \tag{1}$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k. \tag{2}$$

See Table 1 for a notational summary. We make the usual (in the Kalman Filtering literature) assumptions with respect to the correlation of the noise and initial conditions:

$$E(\mathbf{w}_k \mathbf{w}_l^T) = \mathbf{Q}_k \delta_{kl}$$
$$E(\mathbf{v}_k \mathbf{v}_l^T) = \mathbf{R}_k \delta_{kl}$$
$$E(\mathbf{w}_k \mathbf{v}_l^T) = E(\mathbf{w}_k \mathbf{x}_0^T) = E(\mathbf{v}_k \mathbf{x}_0^T) = 0.$$

2.2 Object Geometric Model

The object geometric model describes the size and shape of each link of the articulated object under consideration, and how the links move with respect to one another. This model is a superset of the standard kinematic model used in robotics, that only defines the interrelation of the coordinate systems of each link, but not the shapes of the links [33]. Thus, the object geometric model will determine the position of all features in the world coordinate frame, given the robot's configuration parameters. The use of complex explicit geometric models in object tracking has increased in recent years [12] [17] [30]. We assume that the object model is known a priori, and does not change.

2.3 Object Appearance Model

The object appearance model describes the color, texture, and materials used on each link described by the object geometric model. The combination of the object appearance model and the object geometric model defines the position and appearance of every point on the object. Specifically, this allows us to recover the 3D appearance of the area immediately surrounding a feature, given the feature location and the object configuration [18]. The method used to do this is described in Section 4.

In addition to direct modelling of object attributes, this model includes assumptions such as ambient lighting and background color that are not strictly object feature, but will effect the appearance of features.

2.4 Imaging Model

The imaging model mathematically describes the camera used to image the scene. This is a superset of the perspective or orthogonal projection models, including any lens modeling (such as vignetting, lens distortion, or focusing effects) for the camera. We include the position of the camera in the scene in this model. A fixed camera positon could also be assumed, by including the object position relative to the camera in the object geometric model. This model is assumed to be known and constant.

2.5 Object Dynamic Model

The dynamic model describes the assumptions about motion through the filter's *state space.* In our case, $\mathbf{x}_k$ contains the joint angles $\mathbf{q}$ and joint velocities $\dot{\mathbf{q}}$, so the object dynamic model describes movement through the robot's configuration space. For an in-depth description of configuration space concepts, see [33] or [14]. Intuitively, the dynamic model $\mathbf{f}_k(\mathbf{x}_k)$ is the best estimate (from a modeling standpoint) of the next state of the system given the current state $\mathbf{x}_k$.

For instance, if we know that the robot is being controlled such that it maintains a constant velocity in configuration space, we adopt a "constant velocity" dynamic model

$$\mathbf{f}_k(\mathbf{x}_k) = \mathbf{q}_k + \Delta\dot{\mathbf{q}}_k,$$

where Δ is the sampling interval. If we wish to avoid explicit modeling of motion, we can assume a "constant position" dynamic model,

$$\mathbf{f}_k(\mathbf{x}_k) = \mathbf{x}_k.$$

Note that since $\mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k) + \mathbf{w}_k$, this is not the same as assuming that the object never moves, but rather that the motion is completely unpredictable. Specifically, deviations from the motion model are assumed to be zero-mean, Gaussian, and white. If "no dynamic model" were assumed, the implicit assumption would be what we have described as a constant position dynamic model. In this case, the state vector is simply the object configuration parameters,

$$\mathbf{x}_k = \mathbf{q}_k.$$

If constant velocity is assumed, the velocity through configuration space needs to be estimated as well, so

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{q}_k \\ \dot{\mathbf{q}}_k \end{bmatrix}.$$

Motion models $\mathbf{f}$ could be also formulated to model constant workspace end-effector velocities or a number of other feasible dynamic models.

2.6 Errors in Models

Each of these models affects the robustness of the visual tracking process, and errors in each model will degrade the performance in different ways. Errors in the appearance and imaging models will affect the appearance of an feature template, and will degrade the feature tracking portion of the algorithm in as much as the appearance of the features in the input images no longer matches the expected appearance. As long as the "correct" location of a feature is most similar to the template, feature tracking will occur without errors. Since the our correlation measure does not account for changes in illumination [10], errors in the modeling of illumination would change the object appearance, and would affect feature tracking in much the same way.

Errors in the object geometric model would affect the system in a different way. First, the forward kinematics of the object describe the mapping between the object configuration space and the workspace. Section 3 describes this process. Thus, errors in this mapping will initialize the feature trackers at incorrect positions in the image, positions different from the actual locations of the features. Second, the forward kinematics as well as the partial derivatives of this map are used to compute a linearization of this nonlinear map for the purpose of assimilating feature tracking results into an updated object state. Section 5 describes this process. The implication of an incorrect geometric model on the process is that a minimum least squares fit to the object feature locations is computed, and the estimates of the object parameters would be incorrect in so far as the placement of the features is incorrect with respect to the object due to the geometric errors. Luckily, since the same geometric map is used for the forward projection of feature locations and the assimilation of feature tracking results, small errors in the geometric map result in proper feature tracking at the expense of absolute object configuration estimates. If feature location prediction and object dynamic models were used to a greater extent, such that only the most likely feature locations in the image were searched, errors in the geometric map might be more problematic.

Errors in the dynamic model will draw the state estimate off during the time projection portion of the system. This means that the feature locations will be incorrect to a certain extent. As long as the actual feature locations still fall within the search regions, the only result is that the correction portion of the prediction-correction update equation (see Section 5 for an explanation of the Extended Kalman Filter) will be larger. If too many features fall outside the search region due to an incorrect dynamic model, the feature trackers will not be able to bring the full state estimate, and tracking will fail.

3 Observation Function

Of great importance in our object tracking system is the map from the object state $\mathbf{x}_k$ to the location in the image plane of each feature $\mathbf{m}_k^f$. We call this map the observation function $\mathbf{h}_k(\mathbf{x}_k)$, a vector valued function with two elements for each feature of interest (the u and v location in the image plane of the image of that feature). In our case, the observation function is the composition of the forward kinematics of the robot and the projection equations used. Since we use a PUMA robotic arm with the proximal three joints free to move, and we assume perspective projection with focal lengths f_x and f_y, this reduces to

$$\mathbf{m}_k^f = CT_0^{c_f} p_f = C\Pi_{i=1}^{c_f} A_i p_f$$

$$\mathbf{h}_k^f = \begin{bmatrix} -\frac{\mathbf{m}_k^f[1]}{f_x \mathbf{w}_k^f[3]} \\ -\frac{\mathbf{m}_k^f[2]}{f_y \mathbf{w}_k^f[3]} \end{bmatrix},$$

where $\mathbf{m}_k^f$ is the position of feature f in the camera coordinate frame. The 4×4 transformation matrix C determines the mapping from the world coordinate frame to the camera coordinate frame, and $T_0^{c_f}$ is the standard homogeneous transformation from the coordinate frame in which feature f is specified, c_f, to the world coordinate frame. Transformation $T_0^{c_f}$ is dependent on $\mathbf{x}_k$, and can be decomposed into homogeneous transforms A_i corresponding to each link of the robot. See [33] for more discussion on the Denevit-Hartenberg convention for assigning coordinate transformations.

Since we have closed form expressions for each component of $\mathbf{h}_k(\mathbf{x}_k)$, we can compute the partial derivatives of $\mathbf{h}$ with respect to each component of $\mathbf{x}$ algebraically. These functions will play an important role in Section 5 when we describe the assimilation of feature tracking results into the state estimate update.

4 Feature Tracking

Given the object appearance model and the imaging model, each link can be rendered using

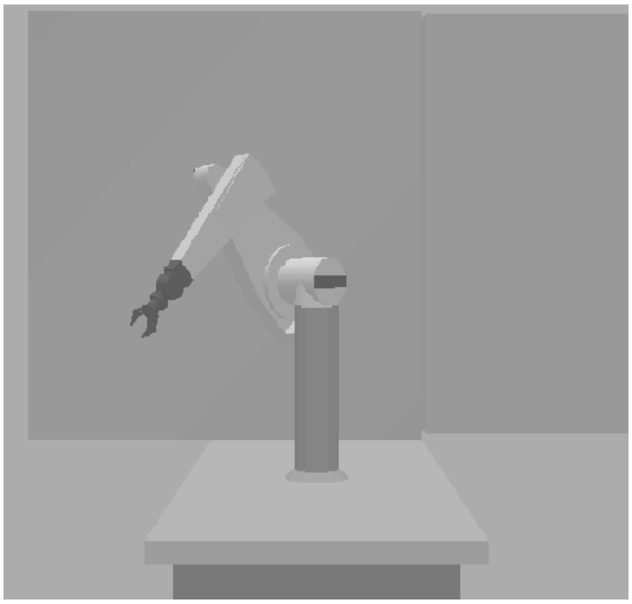


Figure 1: A Synthetic Scene

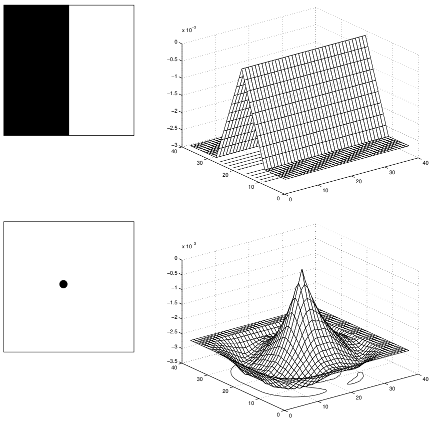
standard computer graphics techniques [38]. If a geometric model and object configuration is assumed, the links can be rendered in the appropriate position, and a complete *synthetic scene* can be rendered. This synthetic scene represents the best estimate, given the models and object configuration estimate, of what the input image will look like. An example synthetic scene is shown in Figure 1. This scene is then used for several computations.

The assumptions used to render the scene can also be used to compute the visibility of each point on the robot from the given viewpoint [35]. Specifically, points on the robot specified as *features* are checked for visibility at each step. A feature is defined to be a three dimensional point defined in a given link coordinate system. Features f determined to be non-visible at a given object configuration $\hat{\mathbf{x}}_k$ have the algebraic observation function $\mathbf{h}_k^f(\mathbf{x}_k) \triangleq [0\ 0]^T$, otherwise the observation function described in Section 3 is used.

Visible features have an expected feature location $\mathbf{h}_k^f(\mathbf{x}_k)$ defined by the modeling assumptions. Since the synthetic scene represents the best estimate of the appearance of the object at the given configuration, it also represents the best estimate for the appearance of the features at that configuration [18]. Therefore, we extract the region surrounding the expected feature location for use as a feature appearance template. Currently, we use a fixed sized rectangular template for each feature.

A correlation measure, in our case the Sum of Squared Differences (SSD) correlation measure, is used to compare the template with the region in the input image surrounding the expected feature location. As this similarity measure is used on each pixel in a region, the scores make up what we term the SSD Surface (SSDS). The shape of the SSDS can tell us how well the template matches the image content surrounding the expected feature location [1] [29]. The amount of drop off in the SSDS, or equivalently the degree to which the template discriminates the most likely feature location from the surrounding area, is an important contributor to increasing the robustness of the tracking system as a whole. If the system can detect when an individual feature tracker is failing, the results from that tracker can be selectively disregarded. This is a different concept from outlier rejection, which although feasible in the context of the EKF [41], is not implemented in the tracking system at the current time. In this case, we are looking strictly at the shape of the SSDS to determine the spatial discrimination power of the template. Figure 2 illustrates what the shape of the SSDS can reveal about feature tracking results. In the top row, vertical movement of the feature in the image cannot be detected. Since SSD scores in the vertical direction do not drop off, the SSDS reveals this fact. In contrast, the second row shows a feature discriminating well in two directions. The SSDS for this case shows SSD score drop offs in every direction away from the peak.

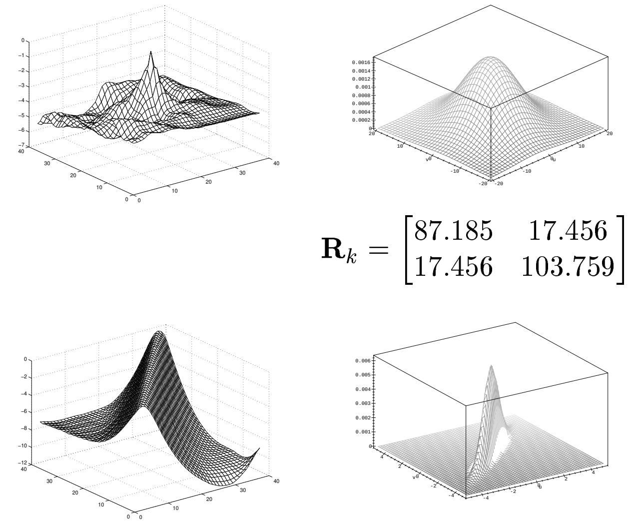
We follow [32] and interpret the SSDS probabilistically. The feature tracking reveals that the feature is most likely at the point of maximum similarity (the peak in Figure 2), and that the probability of the feature location being other places is proportional to the SSD score at that location. Thus the SSDS, scaled appropriately, can be used as an estimate of the probable location of a feature in the image plane. More importantly,



Template SSDS

Figure 2: SSDS Shape

SSDS GRV Density Function



$$\mathbf{R}_k = \begin{bmatrix} 87.185 & 17.456 \\ 17.456 & 103.759 \end{bmatrix}$$

$$\mathbf{R}_k = \begin{bmatrix} 115.130 & 102.345 \\ 102.345 & 96.569 \end{bmatrix}$$

Figure 3: SSDS and GRV density function for Two Features

the local shape of the SSDS can be used to derive a measure of the spatial certainty of that location.

As we will see in Section 5, the location of each feature in the image is modeled as a two dimensional Gaussian random vector (GRV).

The statistics of a GRV are completely determined by the mean $\boldsymbol{\mu}$ and covariance $\mathbf{R}$ of the GRV. We next present a method for estimating the mean and covariance for the GRV we use to represent the feature location in the image. We denote the measurement as the vector $\mathbf{z}_k$ and the covariance as the matrix $\mathbf{R}_k$.

The peak of the SSDS surface is determined, and this is taken to be the mean of the GRV.

$$\boldsymbol{\mu} = \text{argmin } SSD(u, v),$$

where argmin returns the u and v that minimize $SSD(u, v)$. Then the variance in the u and v directions, as well as the covariance of u and v are numerically computed from the scaled SSDS. These measures are used to compute the covariance matrix for the GRV.

$$\begin{aligned} \mathbf{R} &= \begin{bmatrix} Var^2(u) & Cov(u,v) \\ Cov(u,v) & Var^2(v) \end{bmatrix} \\ &= \begin{bmatrix} \sigma_u^2 & \rho_{uv}\sigma_u\sigma_v \\ \rho_{uv}\sigma_u\sigma_v & \sigma_v^2 \end{bmatrix}. \end{aligned}$$

The geometric interpretation of this approximation with respect to the SSDS is shown in Figure 3.

In this section, we will denote individual feature measurements by $\mathbf{z}_k^f$ and $\mathbf{R}_k^f$. Once $\mathbf{z}_k^f$ and $\mathbf{R}_k^f$ have been computed for each feature, the information from the individual feature trackers needs to be combined to make an observation vector and covariance matrix for the system as a whole. The combined measurements are denoted by $\mathbf{z}_k$ and $\mathbf{R}_k$.

To make the measurement vector, we concatenate the means of the individual vectors,

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{z}_k^1 \\ \vdots \\ \mathbf{z}_k^F \end{bmatrix},$$

where there are F features. If we make the assumption that the movement of each feature is independent of the movement of the other features, the covariance of $\mathbf{z}_k$ can be computed by constructing a block-diagonal matrix from the individual covariance matrices,

$$\mathbf{R}_k = \begin{bmatrix} \mathbf{R}_k^1(1,1) & \mathbf{R}_k^1(1,2) & \cdots & 0 & 0 \\ \mathbf{R}_k^1(2,1) & \mathbf{R}_k^1(2,2) & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & \mathbf{R}_k^F(1,1) & \mathbf{R}_k^F(1,2) \\ 0 & 0 & \cdots & \mathbf{R}_k^F(2,1) & \mathbf{R}_k^F(2,2) \end{bmatrix}$$

where the individual $\mathbf{R}_k^f$ are as described above. Modeling the interactions between the uncertainties of the individual feature trackers is a nontrivial task, and is a topic for future research. The interactions of the locations of the features are modeled, in that the movement of each feature plays a role in updating the state estimates $\hat{\mathbf{x}}_k$, as described below.

5 Extended Kalman Filter

This system utilizes the Extended Kalman Filter (EKF), an extension of the standard Kalman Filter [23] [16] to the case of nonlinear observation and dynamic functions. Many tracking researchers use KF or EKF to track individual features or pixels, thus estimating the motion of the feature in the image plane or the workspace [22] [21] [25] . After tracking, feature location and movement information is gathered into an object estimate. In contrast, we use the object configuration as our state vector, and model the forward kinematics of the robot and the projection of the imaging system as part of the EKF. This allows us to incorporate feature tracking results within the context of the object model.

The state space for our case is the configuration space of the robot $\mathbf{x}_k = \mathbf{q}$. We have presented the two main models associated with our modeling of the system, the observation model $\mathbf{h}_k(\mathbf{x}_k)$ and the object dynamic model $\mathbf{f}_k(\mathbf{x}_k)$. The complete nonlinear model used for the system is as given in (1) and (2). Table 1 summarizes the notation used in the EKF.

The following set of equations is generally collectively called the *extended Kalman filter* [4]:

Table 1: Notation Used in this Section

Symbol	Definition
$\mathbf{x}_k$	state (at the k^{th} time instant)
$\mathbf{f}_k$	system function
$\mathbf{w}_k$	white process noise
$\mathbf{z}_k$	measurement
$\mathbf{h}_k$	measurement function
$\mathbf{v}_k$	white measurement noise
$\mathbf{R}_k$	covariance matrix for $\mathbf{v}_k$
$\mathbf{Q}_k$	covariance matrix for $\mathbf{w}_k$
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k\|k}$	state estimate at time k given data $\mathbf{z}_0 \ldots \mathbf{z}_k$
$\hat{\mathbf{x}}_{k\|k-1}$	state estimate at time k given data $\mathbf{z}_0 \ldots \mathbf{z}_{k-1}$

$$\mathbf{P}_{0,0} = Var(\mathbf{x}_0)$$
$$\hat{\mathbf{x}}_0 = E(\mathbf{x}_0)$$
$$\mathbf{P}_{k,k-1} = \left[\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1})\right] \mathbf{P}_{k-1,k-1} \left[\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1})\right]^T + \mathbf{Q}_{k-1}$$
$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1})$$
$$\mathbf{K}_k = \mathbf{P}_{k,k-1} \left[\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]^T \times \left[\left[\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right] \mathbf{P}_{k,k-1} \left[\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]^T + \mathbf{R}_k\right]^{-1}$$
$$\mathbf{P}_{k,k} = \left[I - \mathbf{K}_k \left[\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]\right] \mathbf{P}_{k,k-1}$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1}))$$

6 Experimental Results

Laboratory experiments were performed to validate the tracking performance of the system. In the experiments presented here, the object being tracked is a PUMA robotic arm. The three proximal joints of the PUMA are free to move in these tests. The movement of the PUMA is under external control, but the speed is restricted sufficiently to prevent features from escaping the fixed search areas in a single sampling period.

As the arm moves, some subset of the full feature set is tracked as described in Section 4. The feature tracking results are used as input to the EKF algorithm as described in Section 5, resulting in state estimates. For the purposes of experimental validation, the actual joint angles are available and are shown below. Note that these values are not available to the tracking system.

In this experiment, the arm moves from a parking position down towards the table, as if picking up a block from the table. The arm then moves up away from the table, swings right (from the point of view of the camera), and moves down toward the table again. All three joints move significantly with constant velocity motion in joint space. Note that this fact is not exploited at the current time. The actual joint angles commanded, and the estimation of the joint angles by our filter, are shown in Figure 4. The error in the tracking is shown in Figure 5.

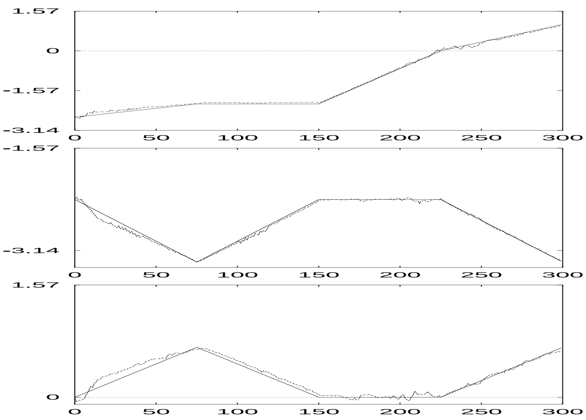


Figure 4: Exercising three degrees of freedom. Top: x_0 and $\hat{x}_0$, Middle: x_1 and $\hat{x}_2$, Bottom: x_2 and $\hat{x}_2$

One feature of the system is the characterization and use of the uncertainties in the kinematic and imaging chain, as modeled by our object and imaging models. See [27] for an in-depth discussion on this. Figure 6 illustrates the uncertainty present in the system during the experiment described above. The variance of the estimate for each joint angle is shown. The time scale is the

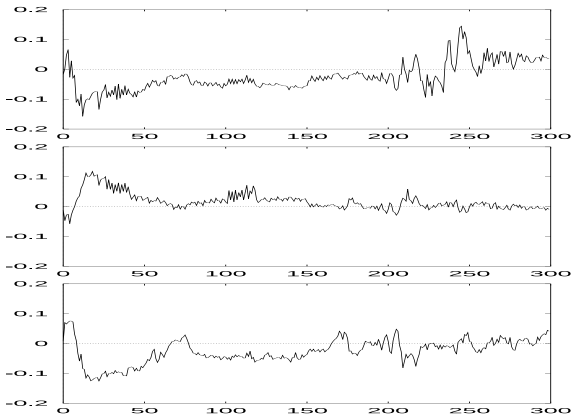


Figure 5: Tracking Error. Top: $x_0 - \hat{x}_0$, Middle: $x_1 - \hat{x}_1$, Bottom: $x_2 - \hat{x}_2$

same as in Figure 4. See [28] for a detailed analysis of the tracking behavior in this experiment. The web site `http://www-cvr.ai.uiuc.edu` contains other demonstrations of the tracking behavior, and animations showing tracking results along with the relevant input images.

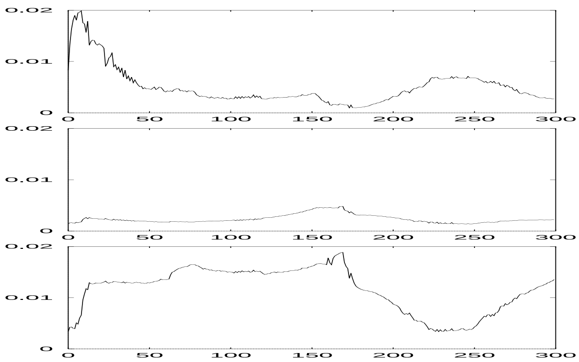


Figure 6: Uncertainty in the tracking system. Top: $P[0,0]$, Middle: $P[1,1]$, Bottom: $P[2,2]$

7 Conclusions

We have described an object tracking system that uses object modeling and online confidence measures to improve the robustness of tracking. The system uses Extended Kalman Filtering to optimally update the current state estimate for an integrated object model using uncertain feature tracking results.

By having an object model integrated into the EKF framework and analyzing feature tracking results in the context of this model, individual feature tracker failures will not cause mistracking. If Kalman Filters are used on individual feature trackers, knowledge about the kinematics of the object and the resultant information about expected motions in the image plane is not directly available.

By evaluating the spatial certainty of the feature tracking results online, templates that are not doing a good job of discriminating the most likely location of a feature can pass this information along to the object tracking algorithm. Thus, the information that the feature tracking is obtaining can be used to its maximum effectiveness. A system that only extracts the location of the minimum SSD score will not have this information available, and will implicitly assign the same confidence to every feature tracking result.

Areas for future research in the context of this system include the extension of this framework to track the full 6 degrees of freedom possible for our PUMA arm, and investigating tracking in situations where the geometric and appearance models for an object are incompletely specified.

References

[1] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *Int'l J. Computer Vision*, 2:283–310, 1989.

[2] N. A. Andersen, O. Ravn, and A. T. Sorensen. Real-time vision based control of servomechanical systems. In *Proc. 2nd Int. Symp. Experimental Robotics*, June 1991.

[3] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noise images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:90–99, January 1986.

[4] R. G. Brown. *Introduction to Random Signal Analysis and Kalman Filtering.* John Wiley & Sons, New York, NY, 1983.

[5] C. K. Chui and G. Chen. *Kalman Filtering with Real-Time Applications.* Springer Series in Information Sciences. Springer-Verlag, Berlin Heidelberg, 1987.

[6] E. E. Dickmanns and V. Graefe. Dynamic monocular machine vision. *Machine Vision and Applications*, pages 223–240, 1988.

[7] J. Q. Fang and T. S. Huang. Some experiments on estimating the 3D motion parameters of a rigid body from two consecutive image frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:545–554, 1984.

[8] A. Gee and R. Cipolla. Fast visual tracking by temporal consensus. *Image and Vision Computing*, pages 105–114, 1996.

[9] D. B. Gennery. Tracking known three-dimensional objects. *Proc. Am. Assoc. Art. Intell.*, pages 13–17, 1982.

[10] G. Hagar and P. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *Proc. 1996 Conf. Computer Vision and Pattern Recognition*, pages 403–410, 1996.

[11] K. Hashimoto, A. Aoki, and T. Noritsugu. Visual servoing with redundant features. In *Proc. of the 35th Conf. Decision and Control*, December 1996.

[12] A. Hauck, S. Lanser, and C. Zierl. Hierarchical recognition of articulated objects from single perspective views. In *Proc. 1997 Conf. Computer Vision and Pattern Recognition*, pages 870–883, 1997.

[13] D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge. Tracking non-rigid objects in complex scenes. In *Proc. Fourth Int'l Conf. Computer Vision*, pages 93–101, April 1993.

[14] J. C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Norwell, MA, 1991.

[15] J. W. Lee, M. S. Kim, and I. S. Kweon. A Kalman filter based visual tracking algorithm for an object moving in 3D. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 342–347, 1995.

[16] F. L. Lewis. *Optimal Estimation with an introduction to stochastic control theory*. John Wiley & Sons, New York, NY, 1986.

[17] John E. Lloyd, Jeffrey S. Beis, Dinesh K. Pai, and David G. Lowe. Model-based telerobotics with vision. In *International Conf. Robotics Automation Albequerque, NM*, April 1997.

[18] R. Lopez, A. Colmenarez, and T. S. Huang. Vision-based head and facial feature tracking. In *Advanced Displays and Interactive Displays Federated Laboratory Consortium, Annual Symposium*, January 1997.

[19] D. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 441–450, May 1991.

[20] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *Int'l J. Computer Vision*, pages 113–122, 1992.

[21] P. Matteucci, C. S. Regazzoni, and G. L. Foresti. Real-time approach to 3-D object tracking in complex scenes. *Electronics Letters*, pages 475–477, March 1994.

[22] L. Matthies, T. Kanade, and R. Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *Int'l J. Computer Vision*, pages 209–236, 1989.

[23] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 141-1 of *Mathematics in Science and Engineering*. Academic Press, Inc, 1979.

[24] A. Z. Meri. On monocular perspective of 3D moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:582–583, November 1980.

[25] D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6), June 1993.

[26] H. H. Nagel. *Overview on image sequence analysis*, pages 2–38. Image Sequence Processing and Dynamic Scene Analysis. Springer-Verlag: Berlin/Heidelberg, 1983.

[27] K. Nickels and S. Hutchinson. Characterizing the uncertainties in point feature motion for model-based object tracking. In *Proc. Workshop on New Trends in Image-Based Robot Servoing, Grenoble, France*, pages 53–63, 1997.

[28] K. Nickels and S. Hutchinson. Weighting observations: The use of kinematic models in object tracking. In *International Conf. Robotics Automation Leuven, Belgium*, 1998.

[29] N. P. Papanikolopoulos. Selection of features and evaluation of visual measurements during robotic visual servoing tasks. *Journal of Intelligent and Robotic Systems*, pages 279–304, July 1995.

[30] H. A. Rowley and J. M. Rehg. Analyzing articulated motion using expectation-maximization. In *Proc. 1997 Conf. Computer Vision and Pattern Recognition*, pages 935–941, 1997.

[31] A. Singh. Incremental estimation of image flow using a Kalman filter. *Journal of Visual Communication and Image Representation*, pages 39–57, 1992.

[32] A. Singh and P. Allen. Image flow computation: An estimation-theoretic framework and a unified perspective. *Comput. Vis. Image Understanding*, pages 152–177, September 1992.

[33] M. Spong and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley & Sons, New York, NY, 1989.

[34] K. Stark and S. Fuchs. A method for tracking the pose of known 3D objects based on an active contour model. In *International Conf. Pattern Recognition*, pages 905–909, 1996.

[35] A. Watt. *3-D Computer Graphics.* Addison-Wesley, Reading, Massachusetts, 2nd edition, 1995.

[36] W. J. Wilson. *Visual servo control of robots using Kalman filter estimates of robot pose relative to work-pieces*, chapter 3, pages 71–104. World Scientific, 1993.

[37] W. J. Wilson. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, pages 684–696, 1996.

[38] Mason Woo, Jackie Neider, and Tom Davis. *The OpenGL Programming Guide.* Addison-Wesley, Reading, Massachusetts, 1996.

[39] J. J. Wu, R. E. Rink, T. M. Caelli, and V. G. Gourishankar. Recovery of the 3D location and motion of a rigid object through camera image (an extended Kalman filter approach). *Int'l J. Computer Vision*, 3:373–394, 1988.

[40] P. Wunsch and G. Hirzinger. real-time visual tracking of 3D objects with dynamic handling of occlusion. In *International Conf. Robotics Automation*, pages 2868–2873, 1997.

[41] Ali Zolghadri. An algorithm for real-time failure detection in Kalman filters. *IEEE Trans. on Automatic Control*, 41(10), October 1996.