Model-Based Tracking of Complex Articulated Objects

Kevin Nickels, Seth Hutchinson

Abstract— **In this paper we present methods for tracking complex, articulated objects. We assume that an appearance model and the kinematic structure of the object to be tracked are given, leading to what is termed a *model-based* object tracker. At each time step, this tracker observes a new monocular grayscale image of the scene and combines information gathered from this image with knowledge of the previous configuration of the object to estimate the configuration of the object at the time the image was acquired. Each degree of freedom in the model has an uncertainty associated with it, indicating the confidence in the current estimate for that degree of freedom. These uncertainty estimates are updated after each observation. An extended Kalman filter with appropriate observation and system models is used to implement this updating process. The methods that we describe are potentially beneficial to areas such as automated visual tracking in general, visual servo control, and human computer interaction.**

Keywords— **Object Tracking, Kalman Filtering**

I. Introduction

With recent advances in computing power and image interpretation algorithms, it is now feasible to *track* the movements of complex objects in a scene. This allows the construction of systems that can, for example, recognize objects and allow a remote operator to specify tasks in terms of objects rather than directly manipulating a robotic arm. The task of *object tracking* is a necessary prerequisite to complex systems such as this. In this paper, we consider the problem of tracking complex, articulated objects in a scene, such as an articulated robotic arm.

Object tracking is the problem of estimating and updating the configuration of an object over time. An object's configuration is defined by its pose parameters (both position and orientation), as well as parameters that define any internal degrees of freedom of the object [3], [9], [18]. In contrast, feature tracking is the problem of estimating the image locations of features in an image sequence [11]. In the visual servo community, another variation of the tracking problem is to control the motion of an active camera such that positions of features in the image are regulated to desired locations [28].

In this paper, we present a model-based object tracking system. Our system exploits known geometric, kinematic, dynamic, and appearance models of the object (in our experiments, a robot arm and a human arm) during tracking. Thus, object tracking becomes the problem of instantiating the parameters of the models given an observed image sequence. Our approach iteratively updates selected model parameters (e.g., internal joint angles or kinematic parameters of an arm) each time a new image in the sequence is observed. In particular, for each frame in the image sequence the following operations are performed. First, the current estimate of the object configuration is used by a graphics engine to compute feature templates. These feature templates are then used by an SSD tracker to estimate the feature locations in the current image. Finally, these observed feature locations are then used to drive an extended Kalman filter (EKF), which updates both the estimate of the object configuration and the internal state of the EKF.

Our system functions on a sequence of monocular grayscale images. It accounts for both self-occlusion and external occlusion of features, and it weights feature observations according to their expected value in disambiguating the state, as well as the amount of spatial uncertainty present in the feature observation.

The remainder of the paper is organized as follows. In Section II we describe our approach to feature tracking and the assimilation of feature tracking results into an object tracking framework. This section includes a review of the related literature on tracking. In Section III, we give a high level description of our tracking system. This section begins with a more detailed discussion of the models that are used, along with a qualitative evaluation of the effects of modeling errors on tracking performance. It then deals with the specific issues of generating appearance templates, assimilating feature tracking results into the object tracking system, and dealing with feature occlusion. In Section IV, we describe several increasingly complex tracking situations that the framework can handle. Each situation illustrates a particular portion of the overall tracking system. In Section V, we present experimental results that illustrate the effectiveness of the tracking system in several situations. Finally, in Section VII, we offer some conclusions that can be drawn from this research, and discuss possible future work.

II. Feature and Object Tracking

There are two main approaches to object tracking. The first derives an *optical flow field*, or a dense motion field, for the sequence, then analyzes the structure of this flow field to infer structure, motion, or both for the objects in the image [25] [2]. The second approach, which we use in our work, is based on the correspondence of discrete features on an object in one image with those features in a subsequent image. This approach, discussed in Section II-A,

K. Nickels is with the Department of Engineering Science, Trinity University, San Antonio TX, USA (email: knickels@engr.trinity.edu)
S. Hutchinson is with the Department of Electrical and Computer Engineering and The Beckman Institute, University of Illinois, Urbana IL, USA (email: seth@uiuc.edu)

typically searches for the locations of features in an image by template matching or through other search techniques, and then infers object motion from these correspondences. The use of the results from this search for the purpose of object tracking is reviewed in Sections II-B.

A. Feature Tracking

Feature tracking provides estimates of the image locations of features in successive images. Commonly, edges, corners, and regions of high visual contrast are used as features in tracking research [9], [18], [4], although any measurable relationship in an image could be used. There is a large and rapidly growing literature on feature tracking; summaries can be found in [7], [15], [20].

Much recent work has focused on the computation of new invariants for use as features in feature tracking. Instead of standard features such as corners and lines, *projective invariants* have been used by Hagar [11]. Projective invariants are image features that are independent of camera position. Hagar and Belhumeur [12] have proposed parametric measures that are insensitive to both changes in geometry such as pose changes relative to the camera and changes in illumination such as pose changes relative to light sources. These measures are combined to arrive at feature similarity measures that are insensitive to these changes, and are used to track image regions in sequences.

B. Object Tracking

The motion of a 3D rigid object can be extracted by analyzing the motion of features in a sequence of images. This is a well-established area of research, for which overviews can be found in [20], [14], [7]. If geometric models for the object are known a priori, these can be exploited in determining the object's motion [10] [5] [6]. If multiple viewpoints of the object are available [8] [17], tracking results from each viewpoint can be integrated to aid in the object tracking. For example, Gennery [9] tracks rigid polyhedral objects. His object model includes a wire frame of the object and constant reflectivity coefficients for each face of the object. His dynamic model includes the position, velocity, and acceleration of the object. Bray [3] also tracks polyhedral objects, utilizing a similar object model. Dellaert et al. [5] track a planar patch, simultaneously estimating the pose of the patch and the texture appearing on the planar portion of the object. Lowe [18] tracks objects with internal degrees of freedom, for example, a box with one degree of freedom (a hinged lid). The object model is a wire frame model, with parameters for the position and orientation of the base and an extra parameter for the angle between the box body and lid. Not all object models must have a geometric interpretation, however. Stephens [27] tracks rigid polyhedral objects using a local Hough transform, so his object model is a point in the 6-D Hough space. In [24], frequency analysis of the spatio-temporal curves recovered from human activity is used to recognize and categorize semi-repetitive activities.

Recently, several research efforts have aimed at tracking humn motion in image sequences [16], [2], [4]. In [16], humans are modelled as a set of connected planar patches. In [6], a cylinder model of the body is used. In [8], tapered super-quadrics are used as 3D limb models. In [2], Eigenspace techniques are used to track objects that change shape as they move (in this paper, a human hand). In [4], the kinematics of the human body are modelled using the exponential map that has become popular in the robotics literature (see, e.g., [19]).

All these systems have the unifying characteristic that there is an underlying object model, and results from the tracking of features are used to update the object parameters. One advantage to this extra modeling is the ability to treat features as items of secondary interest, and to track the object in spite of the occlusion of some features. Therefore, if an individual feature tracker fails but other trackers succeed in localizing their features, the object will still be tracked. Kakadiaris and Metaxas [17] utilize object models with multiple cameras to compute expected visibility of body parts from each camera. Hashimoto et al. [13] have shown, e.g., that increasing the number of redundant features increases the robustness of visual servoing.

III. A model-based object tracking system

In this section, our system for model-based object tracking is described. This system updates the internal parameters of an object model from monocular grayscale images of that object. The system accounts for both self-occlusion and external occlusion of features, and weights feature observations according to their predicted usefulness in disambiguating the state, as well as by the amount of spatial uncertainty present in the feature observation.

We begin by presenting a summary of the various models used by our system. We then examine several components used in the tracking algorithm: feature tracking and measurement uncertainty estimation, assimilation of feature tracking results, and finally provisions for self-occlusion and external occlusion of features. Most issues related to using the extended Kalman filter in our tracking system are deferred to Section IV.

A. Models Used

Our system uses five models: the overall system model, the object geometric model, the object appearance model, the imaging model, and the object dynamic model. We now describe these models and discuss the assumptions that can be made for each. We conclude this section with a brief discussion of the effects of errors in the model on system performance.

We use a general nonlinear **system model**. 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. The system state at time k is denoted by the *state vector* $\mathbf{x}_k$. The nonlinear system model specifies $\mathbf{x}_{k+1}$ by the vector valued function, $\mathbf{f}_k(\mathbf{x}_k)$, plus additive noise $\mathbf{w}_k$. Observations of the system are given by another vector valued function $\mathbf{h}_k$, also a function of state $\mathbf{x}_k$ and additive noise $\mathbf{v}_k$. The equations for the

For each time step k

1. Capture new image I of scene
2. For each feature $f \in 1..F$
   (a) Use object geometric model M and estimated configuration $\hat{\mathbf{x}}_{k|k-1}$ to:
      i. determine visibility of f in the image,
      ii. create appearance template T of f,
      iii. compute expected location $\mathbf{h}(\hat{\mathbf{x}}_{k|k-1})$ of f.
   (b) Compute SSD error measure between T and image patch surrounding image location $\mathbf{h}(\hat{\mathbf{x}}_{k|k-1})$.
   (c) Take min. SSD score as measurement $\mathbf{z}_k^f$.
   (d) Estimate covariance $\mathbf{R}_k^f$ of measurement by analysis of SSDS in local area surrounding $\mathbf{z}_k^f$.
3. Combine feature measurements $\mathbf{z}_f^k$ and covariances $\mathbf{R}_k^f, f = 0..F$ into aggregate measurement $\mathbf{z}_k$ with measurement covariance $\mathbf{R}_k$.
4. Combine $\mathbf{z}_k$, $\mathbf{R}_k$ with current state estimate $\hat{\mathbf{x}}_{k|k-1}$ and current state covariance $\mathbf{P}_{k|k-1}$ to arrive at optimal updated state estimate $\hat{\mathbf{x}}_{k|k}$ and updated covariance $\mathbf{P}_{k|k}$.

Fig. 1. TRACK-OBJECT, an algorithm for object tracking.

model are of the form

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{w}_k, \qquad \mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k.$$

We make the usual (in the Kalman filtering literature) assumptions with respect to the correlation of the noise and initial conditions. That is, the noise sequences $\mathbf{w}_k$ and $\mathbf{v}_k$ are zero-mean white noise sequences, uncorrelated with each other and with the initial state $\mathbf{x}_0$.

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, which defines only the interrelation of the coordinate systems of each link, but not the shapes of the links [26]. Thus, the object geometric model will determine the position of all features in the world coordinate frame, given the robot's configuration parameters. We assume that the object geometric model is known *a priori*, and that it is either constant or can be parameterized by a small number of variables.

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.

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 position 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, and in our experiments we have used perspective projection.

The **dynamic model** describes the assumptions about motion through the *state space*. In our case, for experiments that involve tracking a robot arm, $\mathbf{x}_k$ contains the joint angles $\mathbf{q}$ (which can be used to define a configuration space) and joint velocities $\dot{\mathbf{q}}$; the object dynamic model thus describes movement through the robot's state space.

The state will generally include velocity information (i.e., $\mathbf{x}_k = [\mathbf{q}_k \ \dot{\mathbf{q}}_k]^T$). Motion models $\mathbf{f}$ could also be formulated to model constant workspace end-effector velocities or a number of other feasible dynamic models.

In tracking research, the type of motion allowed is often constrained [1]. Sometimes this constraint is on the relative motion of the camera and the object, as in our case [28], and sometimes the constraint is on the internal motion of the object, for example by assuming rigid objects [9].

Each of these models affects the robustness of the visual tracking process, and errors in each model will degrade the performance in different ways. We now discuss the impact of errors in each model described above.

Errors in the appearance and imaging models affect the appearance of the generated feature templates and degrade the ability of the system to track individual features. While the system monitors the quality of the feature tracking [22], serious errors in these models will cause feature tracking to fail completely.

Errors in the object geometric model affect the system in two ways. The forward kinematics of the object describe the mapping between the object configuration space and the workspace. Thus, the first effect of errors in this mapping is that the feature trackers are initialized at incorrect positions in the image. This could cause the feature tracking to fail. Secondly, the EKF uses this model to perform a minimum least squares fit of the measurement to the object feature locations. Thus, the estimates of the object parameters based on these locations would be incorrect.

Errors in the dynamic model will draw the state estimate off during the time projection portion of the object-tracking algorithm. 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 (6) will be larger.

B. Generating Feature Appearance Templates

The object geometric model and configuration completely specify the relative locations of the link coordinate frames with respect to the world coordinate frame. In order to determine the appearance of the scene, an imaging model such as described in Section III-A is assumed. With these models in place a synthetic scene such as that shown in Figure 2 can be generated.[1]

Feature points are currently manually specified as 3D points in a link coordinate system. Given the geometric

[1]Note that the image produced by the program uses 24 bits of grayscale and currently is produced at a resolution of 512 × 485. Therefore, the quality of the image in this paper does not reflect the quality of the synthetic scene actually used by the system.

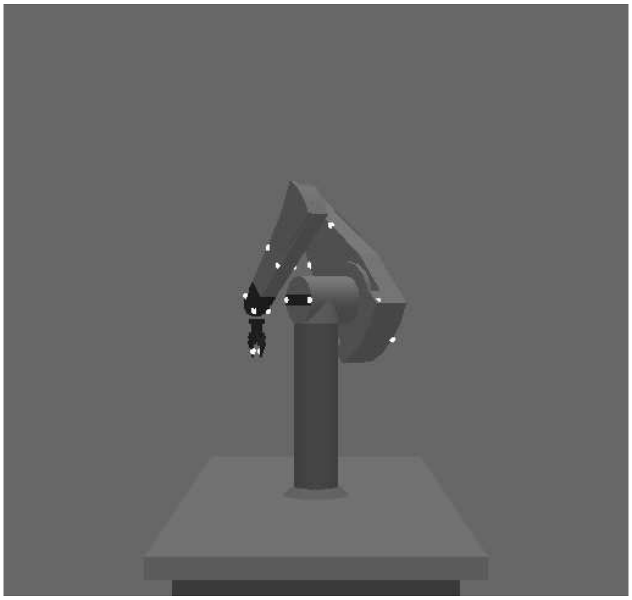


Fig. 2. A synthetic image, showing expected feature locations.

model and an estimated configuration, the (estimated) position of these points in the world coordinate frame can be determined. We define the *expected feature location* as the projection of each feature point onto the image plane. An example set of these locations is shown in Figure 2.

If a feature point is visible in a given configuration, it can be expected that the area in the input image surrounding that feature point will resemble the area in the synthetic image surrounding the expected feature location. This is the principle upon which we base our method for automatically generating templates for arbitrary complex features. A fixed rectangular area of the synthetic image called the *feature appearance template*, centered on the expected feature location, is saved for use during feature tracking.

C. Assimilation of Feature Tracking Results

When the individual feature measurements, $\mathbf{z}_k^f$, and their covariances, $\mathbf{R}_k^f$, have been computed by the individual feature trackers, they are combined to make observation and covariance vectors for the system as a whole. These are denoted as the $2F \times 1$ vector $\mathbf{z}_k$ and the $2F \times 2F$ block-diagonal matrix $\mathbf{R}_k$, and defined by

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{z}_k^1 & \cdots & \mathbf{z}_k^F \end{bmatrix}^T, \qquad \mathbf{R}_k = diag(\mathbf{R}_k^1, \ldots, \mathbf{R}_k^F)$$

in which there are F features. These definitions assume that the additive noise in the feature measurements is independent, and thus that the covariance of $\mathbf{z}_k$ is a block-diagonal matrix whose non-zero entries are the individual covariance matrices. Modeling the interactions between the *uncertainties* of the individual feature trackers is a nontrivial task [25] 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 $\hat{\mathbf{x}}_k$.

D. Treatment of Feature Occlusion

Feature occlusion occurs when a feature is not completely visible from the viewpoint of the camera. Whether a particular feature is occluded is a function of the camera viewpoint, the object configuration, and the position of external objects that may obstruct the view of the object. Note that in our framework, the loss of a feature due to occlusion causes mistracking only to the extent that the erroneous feature tracking information causes errors in the state estimate.

We define *off-screen* occlusion to be the case that a feature projects to a portion of the (infinite) image plane that is not observed. We define *self-occlusion* to be the occlusion of a feature by another part of the modeled object. If a feature f is found to be not visible due to being off-screen or due to self-occlusion, the relevant elements of the *observation function* of the extended Kalman filter ($h_k^{2f}(\mathbf{x}_k)$ and $h_k^{2f+1}(\mathbf{x}_k)$, see Equation (1) of Section IV-A) are set to a constant. This will cause

$$\frac{\partial h_k^{2f}}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k) = \frac{\partial h_k^{2f+1}}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k) = 0,$$

and results from tracking the non-visible feature will not be used in the state update (Equation (3) of Section IV-A). Indeed, in the interest of performance, tracking is not even performed on non-visible features.

External occlusion is occlusion that cannot be predicted from the object model and estimated configuration. This includes features predicted to be visible that are not visible, as well as features that have some unmodeled object obscuring the visibility of the feature.

When a feature is externally occluded, the template will not match any area of the image very well. The feature tracking will detect this (see [22] for a discussion of the estimation of measurement uncertainty in feature tracking), and the feature tracking algorithm will indicate high spatial uncertainty on the returned feature location. Measurements with high spatial uncertainty due to occlusion will be selectively ignored according to the weights computed in Equation (4) in Section IV-A. Therefore, the assumed dynamic model will be used to a greater extent when updating the state estimate.

IV. Application of Kalman filtering framework

In this section, we apply the extended Kalman filtering framework to several examples of interest. We begin by describing the general case, where a general kinematic structure is assumed. Then, we see how this would be used in the simple case of a 3DOF PUMA robotic arm. Finally, we show how this framework can be used in the case of a partially unspecified kinematic chain, such as one with unknown (but constant) link lengths.

A. General Case

The extended Kalman Filter is a flexible and robust tool for tracking. In this section, we describe its use, in conjunction with the modeling described above, for the visual tracking of complex articulated objects.

The state vector will in general contain the joint angles $\mathbf{q}_k = [q_1 \ \dots \ q_n]^T$ and velocities for each of the n joints of the robot. The evolution of this state vector is predicted by the the constant velocity dynamic model as described in Section III-A

$$\mathbf{x}_k \triangleq \begin{bmatrix}\mathbf{q}_k \\ \dot{\mathbf{q}}_k\end{bmatrix}, \ \mathbf{f}_k = \begin{bmatrix}\mathbf{q}_k + \Delta\dot{\mathbf{q}}_k \\ \dot{\mathbf{q}}_k\end{bmatrix}, \ \frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k} = \begin{bmatrix}\mathbf{I}_n & \Delta \times \mathbf{I}_n \\ 0_n & \mathbf{I}_n\end{bmatrix},$$

where $\mathbf{I}_n$ is a $n \times n$ identity matrix, and 0_n is an $n \times n$ matrix of zeros.

The observation function is determined by the object geometric model, the imaging model and the position of the camera in space. The position of the camera in the world coordinate system is known and fixed, given by a homogeneous transformation matrix C. We will now introduce some additional notation to simplify the observation function. A feature is described by a vector $\mathbf{p}$, specifying the location of the point that projects onto the feature, and a number, specifying the link to which the coordinate system containing $\mathbf{p}$ is specified. We will describe a point $\mathbf{p}$ in some coordinate frame by a four-vector $\mathbf{p} = [x \ y \ z \ 1]^T$, where x, y, and z are the coordinates of the point. We define the function $\mathbf{T}(f, \mathbf{x}_k)$ to be the transformation from the world frame to the frame containing feature f.

The function PERSPECT () returns the perspective projection of the point specified, as a column vector. The observation equation, as a function of the current state, is comprised of the composition of the forward kinematics of the robot arm, the position of the feature with respect to the link, the transformation from the world coordinate system to the camera coordinate system, and the imaging transform. With this additional notation, the observation equation becomes

$$\mathbf{h}_k(\mathbf{x}_k) = \begin{bmatrix}\text{PERSPECT}(C\ \mathbf{T}(1, \mathbf{x}_k)\mathbf{p})^T \\ \vdots \\ \text{PERSPECT}(C\ \mathbf{T}(F, \mathbf{x}_k)\mathbf{p})^T\end{bmatrix}, \tag{1}$$

where there are F features to be observed. Note that $\mathbf{h}()$ is a column vector, and contains the projected image plane coordinates of each feature. Although $\mathbf{h}(\mathbf{x}_k)$ is dependent on the state, the existence of a fully specified geometric model allows closed form solutions for $\mathbf{h}(\mathbf{x}_k)$ and $\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\mathbf{x}_k)$ to be found. We assume for simplicity that the process noise does not vary with the state.

With the above model definitions, the extended Kalman filter for this general tracking scenario is given by:

$$\begin{aligned}\mathbf{P}_{k,k-1} &= \left[\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1})\right] \mathbf{P}_{k-1,k-1} \left[\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1})\right]^T \\ &\quad + \mathbf{Q}_{k-1} \\ &= \begin{bmatrix}\mathbf{I}_n & \Delta\mathbf{I}_n \\ 0_n & \mathbf{I}_n\end{bmatrix} \mathbf{P}_{k-1,k-1} \begin{bmatrix}\mathbf{I}_n & 0_n \\ \Delta\mathbf{I}_n & \mathbf{I}_n\end{bmatrix} + \mathbf{Q}_{k-1} \end{aligned} \tag{2}$$

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}) = \begin{bmatrix}\widehat{\mathbf{q}}_{k-1} + \Delta\widehat{\dot{\mathbf{q}}}_{k-1} \\ \widehat{\dot{\mathbf{q}}}_{k-1}\end{bmatrix} \tag{3}$$

$$\mathbf{K}_k = \mathbf{P}_{k,k-1}\left[\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]^T \times \left[\left[\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]\mathbf{P}_{k,k-1}\left[\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]^T + \mathbf{R}_k\right]^{-1} \tag{4}$$

$$\mathbf{P}_{k,k} = \left[\mathbf{I} - \mathbf{K}_k\left[\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1})\right]\right]\mathbf{P}_{k,k-1} \tag{5}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1})), \tag{6}$$

where $\mathbf{h}_k(\mathbf{x}_k)$ is given in (1). The *measurement noise* $\mathbf{R}_k$ is based on analysis of the observations, describing the efficacy of the measurements in each dimension of the observation space. See [22] for more discussion on the computation of $\mathbf{R}_k$. The *process noise* $\mathbf{Q}_k$ is a measure of how much uncertainty is introduced into the state estimate at each iteration. This parameter is often set *a priori*. The *state uncertainty* $\mathbf{P}_{k,k}$ monitors the amount of confidence in the state estimate $\hat{\mathbf{x}}k|k$. The *Kalman gain matrix* $\mathbf{K}_k$ determines the relative weighting of the *a priori* state estimate with the update in the prediction-correction equation (6).

Note that the derivation of a closed form solution of observation equation is the main modeling task in implementing this framework. As described above, the observation equation is the composition of the location of the feature set with respect to the robot's links, the forward kinematics of the arm, the camera's location in space with respect to the robot and the camera's imaging model. Once a composite function has been found, the partials of this function with respect to each state element must also be found in closed form. This algebraic and trigonometric manipulation eliminates the need to compute differences of discrete quantities, a process well-known to be sensitive to noise in the quantities.

B. Case 1: 3DOF PUMA Robotic Arm

In this section, we present a tracking scenario driving this research, that of a three degree of freedom robotic arm under perspective projection. In this case, the three degrees of freedom used to position the tool at any position in the workspace are tracked. The final three degrees of freedom determine the orientation of the tool about this point in space, and are not currently tracked.

As described above, we assume a constant velocity motion model in state space. We again use a state vector containing the joint angles and velocities for each of the first three joints, $\mathbf{x}_k = [q_1 \ q_2 \ q_3 \ \dot{q}_1 \ \dot{q}_2 \ \dot{q}_3]^T$. We assume a fixed pinhole camera imaging model, with perspective projection as above. With these assumptions, a closed form solution is found for $\mathbf{h}_k(\mathbf{x}_k)$, and expressions are derived off-line for $\dfrac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\mathbf{x}_k)$. These expressions are not given here due to space constraints, but can be found in [23].

C. Case 2: Incomplete Object Model

This section describes the application of a tracking system to a two-link arm with two degrees of freedom under perspective projection, where the lengths of the two links will be estimated along with the configuration parameters.

A simple augmentation of $\mathbf{x}_k$ is all that is required to use the system in this case. The state vector in this case contains the joint angles $\mathbf{q}$ of the robot, the velocities of those angles, and the link lengths $\mathbf{a}$ of the robot, $\mathbf{x}_k = [q_1 \quad q_2 \quad \dot{q}_1 \quad \dot{q}_2 \quad a_1 \quad a_2]^T$. Note that the velocities of the lengths are not included in the state vector, as we assume that the lengths are unknown constants. The dynamic model incorporating these assumptions is given by

$$\mathbf{f}_k(\mathbf{x}_k) = \left[q_1 + \dot{q}_1\Delta \quad q_2 + \dot{q}_2\Delta \quad \dot{q}_1 \quad \dot{q}_2 \quad a_1 \quad a_2\right]^T.$$

The observation function remains the same as in the previous case, except that there are four configuration parameters, and the link lengths a_1 and a_2 are now variable.

V. Experimental Results

In this section we illustrate the effectiveness of the tracking system in several situations. Each situation illustrates a particular feature of the tracking system. We begin by showing tracking results for an arm with three degrees of freedom in several situations, and we use this case to illustrate the features of the system[2]. Then we show the flexibility of the system, by showing the tracking of a 2DOF arm with unknown link lengths. In this case, the object appearance model for the arm is also extremely simple, illustrating the flexibility of the object models for gross motion tracking.

In these experiments, the position of the object is manually initialized (although not always correctly — see Section V-D). As can be seen from the results, the joint angle estimates are not always highly accurate, particularly when movement of that joint causes motion orthogonal to the (single) camera. One interesting aspect of this work is its ability to characterize and tolerate this uncertainty. Many systems exist that use multiple cameras (e.g. [8] [17]) to avoid this situation.

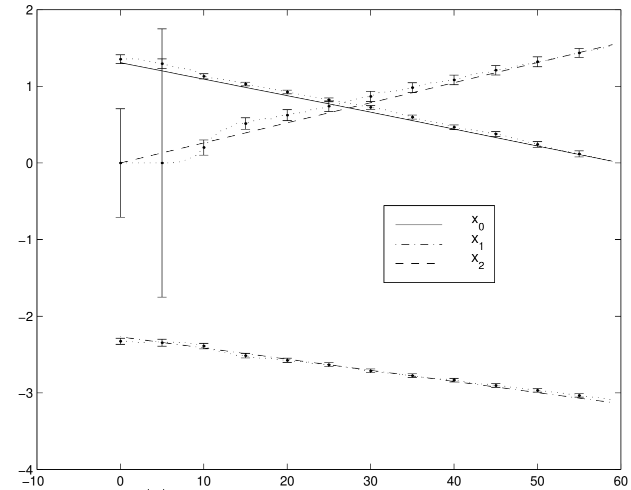
Due primarily to the SSD computation, this system was implemented off-line on Sparc-10 workstations. Inter-frame point feature motion is limited by the search regions of the feature tracking, set to 39×39 regions in these experiments.

A. Case 1b: 3DOF PUMA: Self-Occlusion of Features

In this section, we illustrate how the system handles self-occlusion of features. An overview of the theory of this operation is given in Section III-D. For example, in certain configurations of the object, the upper arm link of the robot is between the camera and the gripper. In these configurations, any features on the gripper would be said to be self-occluded.

Figure 3(a) illustrates the evolution of the state and uncertainty estimates during tracking. Figure 3(b) shows the expected visibility of each of the features throughout tracking.

[2] These results are more easily seen using 3D animations. At the time of this writing, these results can be found at `http://www.engr.trinity.edu/~knickels/`



(a) State and Uncertainty Estimates

(b) Feature Visibility Estimates

Fig. 3. Tracking results for Case 1b. Uncertainty estimates have been subsampled for display.

B. Case 1c: 3DOF PUMA: External Occlusion of Features

In this section, we illustrate the treatment of external occlusion, such as described in Section III-D. Recall that external occlusion is defined as occlusion of features that cannot be predicted from object models, such as a person walking between the camera and the tracked object. Figure 4 illustrates the evolution of the state and uncertainty estimates during tracking.

C. Case 1d: 3DOF PUMA: General Case

When the system is in full operation, the features described in the previous sections operate as appropriate, yielding an object-tracking system that can track features from widely varying viewpoints and can intelligently deal with the appearance and disappearance of features on an object. In this section, we illustrate this with a tracking example without the restrictions imposed above. Figure 5 illustrates the evolution of the state and uncertainty estimates during tracking.

D. Case 2b: 2DOF ARM: Incorrect link lengths

In this experiment, the lengths of the arm and the initial joint angles are initialized in substantially incorrect values. Figure 7 illustrates the initial conditions of the filter with respect to the initial conditions of the actual arm, and shows the tracking behavior of the system. In this experiment, the joint angles of the arm do not change for approximately the first 15 frames of the sequence.

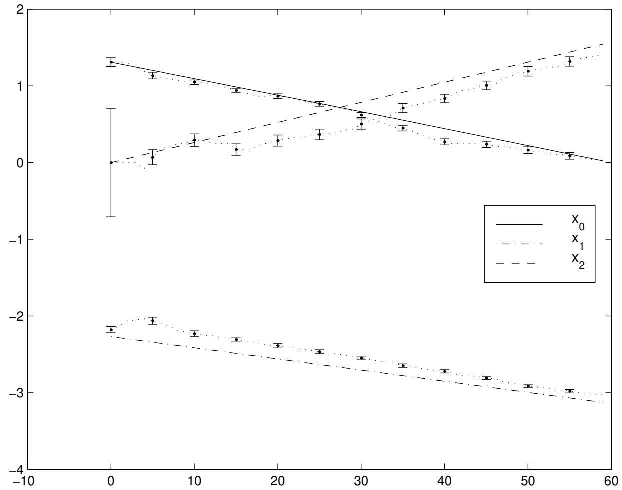


Fig. 4. Tracking results for Case 1c. Uncertainty estimates have been subsampled for display.

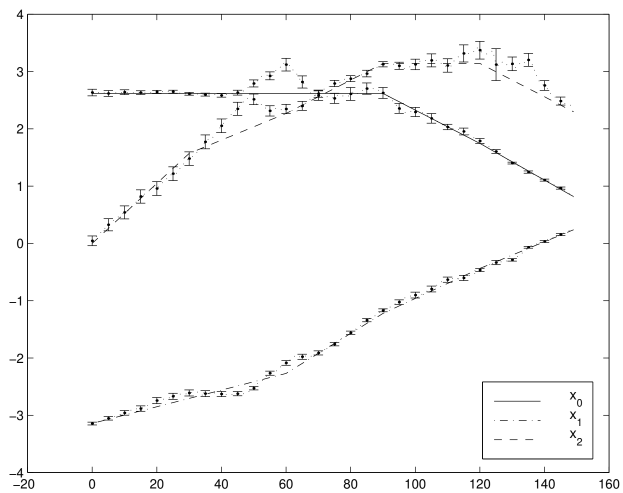


Fig. 5. Tracking results for Case 1d. Uncertainty estimates have been subsampled for display.

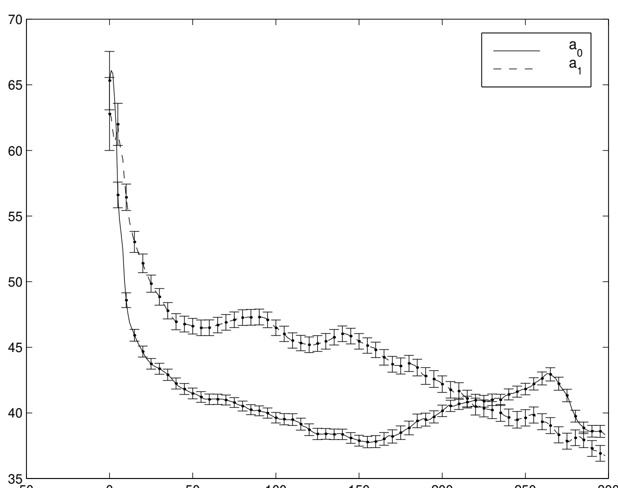


Fig. 6. Link Length Estimation results for Case 2b. Link Length and uncertainty estimates have been subsampled for display.

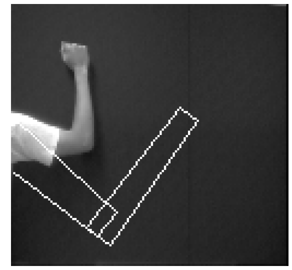


Fig. 7. Initial Conditions for Case 2b.

VI. Future Work

The framework introduced in this paper is general and flexible, and could be applied in many situations not covered in this paper. However, there are many portions of the system that have not been developed to their full extent and that could benefit from more sophisticated techniques. In this section, we will investigate some directions these extensions could take, as well as some novel situations that could benefit from such a general framework.

One aspect of the system that could be extended is the choice for the search area and template size. Currently, a fixed size rectangular search area and fixed size rectangular template are used, regardless of the size and expected motion of the feature. Knowledge about the object geometric model and expected feature motion could be used to define an image region most likely to contain the feature, for the purpose of using as a search region.

The extension of the observation equations to the case where there are multiple cameras is straightforward. This would allow better 3D feature localization, and improved accuracy in tracking. This would be particularly helpful in cases like the PUMA described in Section IV-B where the workspace is large relative to the feature size. The observation equations could also be parameterized to allow active vision.

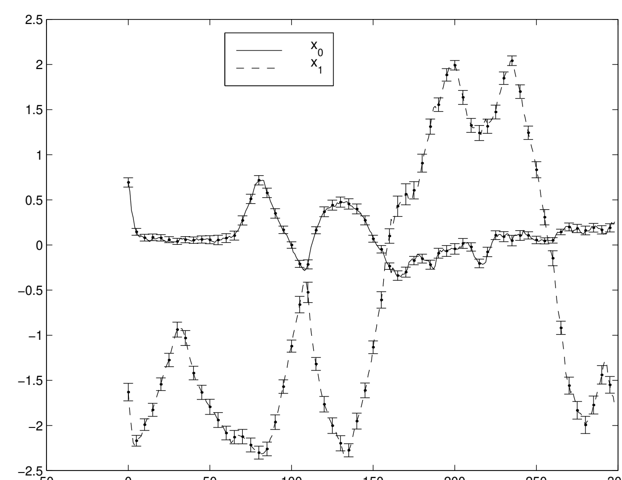


Fig. 8. Joint Angle Tracking results for Case 2b. Uncertainty estimates have been subsampled for display.

This framework is easily adaptable to the case where an optimal object trajectory is known. This would allow non-contact monitoring in certain situations of articulated object under external control. This has many potential applications in the fields of visual servoing, automated assembly, augmented reality, and human-computer interaction.

VII. Conclusions

This paper presents a model-based object-tracking system capable of tracking complex articulated objects on the basis of monocular grayscale images of that object as it moves. The object model at the heart of the system is assumed to be known *a priori*, and is assumed to be perfect. In certain cases this restriction can be relaxed by parameterizing unknown facets of the object model.

We have described the synergy that exists between the object model and the low-level feature tracking used in the system. The optimal filtering framework exploits the object model to overconstrain the feature trackers to operate in a manner consistent with the known kinematics of the object under consideration, so that the inaccurate feature-tracking results are not fatal to the system. The feature-tracking results are used in turn to optimally update the state vector, so that the object model is incrementally brought into accordance with observed data. This system combines the top-down approach of imposing an integral object model on the observed data with the bottom-up approach of using observed data to modify the assumptions of the system in a mathematically robust framework.

The use of spatial certainty measures allows the feature-tracking algorithm to convey the quality of the feature-tracking results to the object-tracking algorithm. Thus, the information that the feature-tracking algorithm obtains 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.

Several features of this tracking system are novel. We have developed a method for generating feature templates for complex features from widely varying viewpoints, a method for estimating the reliability of feature tracking results during tracking, [21], and a characterization of the use of the object models in considering what point feature motion reveals about object motion in images. Each of these represents a contribution to the field of computer vision, and an advance in the state of the art in using object geometric models in tracking complex articulated objects.

References

[1] J. K. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images - A review. *Proceedings of the IEEE*, 76(8):917–935, August 1988.

[2] M. J. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[3] A. J. Bray. Tracking objects using image disparities. *Image and Vision Computing*, 8(1):4–9, February 1990.

[4] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, 1998.

[5] F. Dellaert, S. Thurn, , and C. Thorpe. Jacobian images of super-resolved texture maps for model-based motion estimation and tracking. In *Proceedings IEEE Workshop on Applications of Computer Vision (WACV'98)*, pages 2–7, October 1998.

[6] S. Dettmer, A. Seetharamaiah, L. Wang, and M. Shah. Model-based approach for recognizing human activities from video sequences. Technical report, University of Central Florida, June 1998.

[7] Olivier Faugeras. *Three Dimensional Computer Vision*. MIT Press, 1993.

[8] D. M. Gavrila and L. S. Davis. 3-D model-based tracking of humans in action: A multi-view approach. In *Proceedings IEEE Computer Society Conference on Computer Vision Pattern RecognitionSan Francisco, CA*, 1996.

[9] D. B. Gennery. Visual tracking of known three-dimensional objects. *International Journal of Computer Vision*, 7(3):243–270, 1992.

[10] L. Goncalves, E. DiBernardo, E. Ursella, and P. Perona. Monocular tracking of the human arm in 3D. In *Proceedings International Conference Computer Vision*, 1995.

[11] G. Hager. Real-time feature tracking and projective invariance as a basis for hand-eye coordination. In *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, pages 533–539, 1994.

[12] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[13] K. Hashimoto, A. Aoki, and T. Noritsugu. Visual servoing with redundant features. In *Proceedings of the Conference on Decision and Control*, pages 2483–2483, December 1996.

[14] T. S. Huang and A. N. Netravali. Motion and structure from feature correspondences: A review. *Proceedings of the IEEE*, 82(2):252–268, February 1994.

[15] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, October 1996.

[16] S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *2nd International Conference on Automatic Face and Gesture Recognition*, pages 38–44, October 1996.

[17] Ioannis A. Kakadiaris and Dimittris Metaxas. Model-based estimation of 3d human motion with occlusion based on active multi-viewpoint selection. In *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, page 81, 1996.

[18] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, August 1992.

[19] R. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, Florida, 1994.

[20] H. H. Nagel. Overview on image sequence analysis. In T. S. Huang, editor, *Image Sequence Processing and Dynamic Scene Analysis*, pages 2–38. Springer-Verlag, Berlin, 1983.

[21] K. Nickels and S. Hutchinson. Weighting observations: The use of kinematic models in object tracking. In *Proceedings IEEE International Conference Robotics and Automation*, 1998.

[22] K. Nickels and S. Hutchinson. Measurement error estimation for feature tracking. In *Proceedings IEEE International Conference Robotics and Automation, Detroit, Michigan*, pages 3230–3235, 1999.

[23] K. M. Nickels. *Model Based Articulated Object Tracking*. PhD thesis, University of Illinois, Urbana, IL, 61821, October 1998.

[24] R. Polana and R.C. Nelson. Detecting activities. In *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, pages 2–7, June 1993.

[25] A. Singh and P. Allen. Image flow computation: An estimation-theoretic framework and a unified perspective. *Computer Vision Graphics and Image Processing: Image Understanding*, 56(2):152–177, September 1992.

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

[27] R. S. Stephens. Real-time 3D object tracking. *Image and Vision Computing*, 8(1):91–96, February 1990.

[28] W. J. Wilson, C. C. W. Hulls, and G. S. Bell. Relative end-effector control using Cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696, October 1996.

Kevin Nickels recieved the Ph.D. and M. S. degrees in electrical engineering in 1998 and 1996, respectively, from the University of Illinois at Urbana-Champaign. He received a B.S. degree with distinction in computer and electrical engineering from Purdue University in 1993.

Dr. Nickels in an Assistant Professor with the Department of Engineering Science at Trinity University. His research interests include computer vision and image analysis, visual control of robotic manipulators, and the study of engineering education.

Seth Hutchinson received the Ph.D. degree from Purdue University, West Lafayette, IN, in 1988.

He spent 1989 as a Visiting Assistant Professor of Electrical Engineering at Purdue University. In 1990, he joined the faculty at the University of Illinois in Urbana-Champaign, where he is currently an Associate Professor in the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute for the Advanced Science and Technology.

Dr. Hutchinson is currently a senior editor of the *IEEE Transactions on Robotics and Automation*. In 1996 he was a guest editor for a special section of the *Transactions* devoted to the topic of visual servo control, and in 1994 he was co-chair of an IEEE Workshop on Visual Servoing. In 1996 and 1998 he co-authored papers that were finalists for the King-Sun Fu Memorial Best Transactions Paper Award. He was co-chair of IEEE Robotics and Automation Society Technical Committee on Computer and Robot Vision from 1992 to 1996, and has served on the program committees for more than thirty conferences related to robotics and computer vision.