Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

A New Approach to the Use of Edge Extremities for Model-based Object Tracking*

Youngrock Yoon, Akio Kosaka, Jae Byung Park and Avinash C. Kak
Robot Vision Lab
Purdue University
West Lafayette, IN 47907 U.S.A.
{*yoony,kosaka,jbpark,kak*}*@ecn.purdue.edu*

***Abstract*— This paper presents a robust model-based visual tracking algorithm that can give accurate 3D pose of a rigid object. Our tracking algorithm uses an incremental pose update scheme in a prediction-verification framework. Extended Kalman filter is used to update the pose of a target incrementally to minimize the error between the expected map of the target model and the corresponding gradient edge in the image space. The main contributions of this paper include: 1) A novel approach to how we use the two extremities of straight-lines as features. By taking into account the measurement uncertainties associated with the locations of the extracted extremities of the straight-line, our approach can compare correctly two straight-lines of different lengths. 2) Our use of a *test of mean* criterion for initiating backtracking and our use of a variable threshold on the output of this criterion that makes nil-matching more effective. We have tested our tracking algorithm with image sequences containing highly cluttered backgrounds. The system successfully tracks objects even when they are highly occluded.**

***Index Terms*— object tracking, 3D pose estimation, feature representation, extended Kalman filter.**

I. INTRODUCTION

Recently, robotic control has advanced to the extent of enabling vision-guided robotic assembly on a moving line [1]. For this task, a vision system must provide accurate 6-DOF pose – both position and orientation – of an object in 3D Cartesian space. Furthermore, for a system to work in a real environment, it has to be able to handle robustly the various conditions in the working environment such as occlusions, complicated background, inconsistent lighting conditions, etc.

In this paper, we present a model-based object tracking algorithm that provides accurate 3-D pose of a target object. Of course, ours is not the first model-based object tracking algorithm. A commonly used approach [2] [3] uses numerous points on the wire-frame model of the target as model features. Algorithms in this approach minimize the squared sum of distances from the projection of these points to the matching scene features. For each model feature, a candidate scene feature is located by searching in a direction that is perpendicular to the projection of the model feature. For each model feature, the first edge point on this direction is chosen as the matching scene feature. For that reason, these algorithms work only when the model features are close to the true matching scene features, which is the case when the initial pose is fairly close to the true pose and the motion of the target is very smooth. To resolve this problem, voting-based schemes for estimating the pose parameters have been proposed [4] [5] [6]. The voting based algorithms are more robust than the previous algorithms, but they are time-consuming and not suitable for real-time applications.

Two other approaches we would like to mention use appearance image databases of a target for tracking [7] [8]. In these approaches, the appearance images of the target from a number of viewing aspects are acquired in the learning phase. During testing, for a given scene the target pose is first roughly estimated by registering the input scene to the closest image in the appearance image database. Then, the target pose is refined further by projecting the model features into the image plane using the rough pose and matching the projected model features with the features extracted from the scene.

An alternative approach is to search and select the matching scene feature for each model feature individually with a model that consists of small number of features [9] [10] [11]. By reducing the number of feature matchings, it becomes feasible to assess the validity of an individual match and use backtracking if the current matching leads to a large error. Straight edges are most widely used as features in model-based pose estimation systems because they are invariant to perspective projection. Since straight line features frequently show up fragmented in a scene image, such features are often represented in the Hough space for the purpose of matching. In the Hough space, each straight line feature is represented by its orientation (θ) and its perpendicular distance from the origin (ρ).

While such a comparison between a model edge and a scene edge can provide us with a great deal of robustness with regard to line breakage, it can suffer from a serious flaw, as we recently discovered in our work on model-based object tracking. The flaw has to do with the fact that a (ρ, θ) based comparison is highly dependent on the actual locations of the two edges that are being compared. For example as depicted in Fig. 1, in our own perception, the degree to which the line segment p_1 is matchable with the line segment p_2 is the same as the degree to which q_1 is matchable with q_2. However, if we use the Euclidean distance in the Hough space to assess, on one hand, the matchability of p_1 with p_2, and then, on the other, the

*This work is supported by the Ford Motor Company

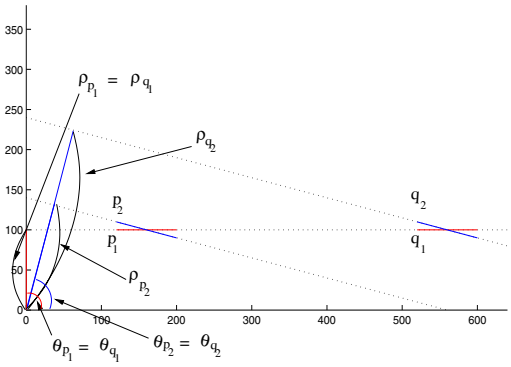


Fig. 1. Example picture that shows the drawback of comparing two straight line features with the Hough parameters. The (ρ, θ) space distance between p_1 and p_2 is different from that between q_1 and q_2

matchability of q_1 with q_2, we get entirely different results.

There has been work done previously (see, for example, [9]) in which the straight line edge features were represented by their extremities. But these previous contributions do not provide clean solutions to the problem of edge breakage and truncation caused by noise and occlusion. In our system, feature measurement uncertainties are assigned to the locations of the two extremities of a straight line edge. These uncertainties are a function of the difference between the lengths of the model edge and the corresponding scene edge. The uncertainties get plugged into an extended Kalman filter that is used for updating the pose parameters associated with the scene object. How exactly this is done is explained in Section II-C.

Our tracking system works by applying the model-based pose estimation algorithm to each consecutive frame in the input image sequence. We present the pose estimation algorithm in the next section. The algorithm that is used to search for a candidate scene edge for a given model edge is presented in Section III. Section V presents experimental results that show the accuracy and robustness of our tracking system with regard to complex background, occlusions and non-smooth motion with large change of viewing aspect. We will summarize this paper and address things yet to be done in Section VI.

II. Pose Estimation Algorithm

As is typical in model-based vision systems, our pose estimation system first projects the model edges of a target object into the scene image and finds the best matching scene edge for a given model edge. Then, it updates the pose to minimize the error between the matching edges. This process continues until all the model edges are matched. Fig. 2 depicts an overview of the pose estimation algorithm.

While the flow of execution depicted in Fig. 2 is generic and common to many model based vision systems (see, for example, [9]–[11]), what is novel about the approach presented here is the manner in which we associate uncertainties between a specific model-to-scene match. The exact manner of how these uncertainties are ascertained

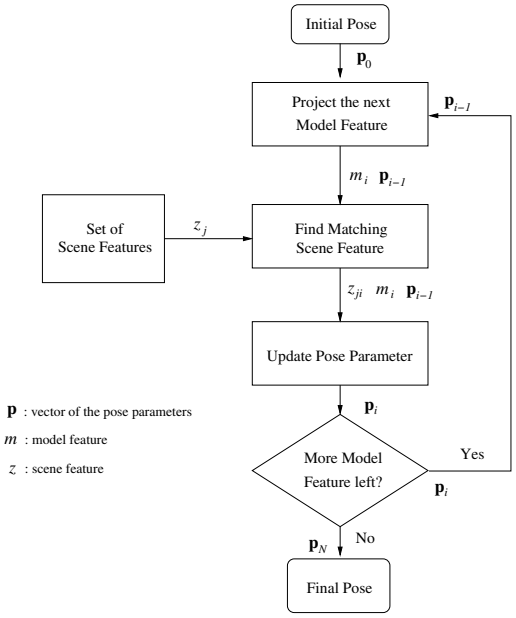


Fig. 2. Overview of our pose estimation algorithm

is presented in Section II-C. For now, we first want to delineate the overall estimation-theoretic framework in which these uncertainties are used.

Our pose estimation algorithm uses a maximum likelihood framework – that means it estimates the pose of the target by maximizing the probability of matching model edges to scene edges, the probabilities being conditioned on the pose parameters. Let m_i be a model edge and z_{j_i} be a scene edge that is matched with m_i and $\mathbf{p}$ be a random vector that represents the current pose of the target. We denote the conditional probability that m_i is matched with the z_{j_i} given $\mathbf{p}$ as follows:

$$prob[m_i \rightarrow z_{j_i}|\mathbf{p}] \tag{1}$$

where the symbol $m_i \rightarrow z_{j_i}$ denotes the event of m_i being matched with z_{j_i}. Our objective is to find the list of model and scene edge pairs that maximizes the following probability:

$$P_{match} = prob[m_1 \rightarrow z_{j_1}, \cdots, m_N \rightarrow z_{j_N}|\mathbf{p}] \tag{2}$$

where N is the total number of model edges. This joint probability can be decomposed in the following manner:

$$\begin{aligned} P_{match} = prob[m_1 \rightarrow z_{j_1}, \cdots, m_N \rightarrow z_{j_N}|\mathbf{p}] = \\ prob[m_1 \rightarrow z_{j_1}|\mathbf{p}] \times prob[m_2 \rightarrow z_{j_2}|m_1 \rightarrow z_{j_1}, \mathbf{p}] \times \cdots \\ \times prob[m_N \rightarrow z_{j_N}|m_1 \rightarrow z_{j_1}, \cdots, m_{N-1} \rightarrow z_{j_{N-1}}, \mathbf{p}] \end{aligned} \tag{3}$$

Let us focus on the second term in the product shown on the right:

$$prob[m_2 \rightarrow z_{j_2}|m_1 \rightarrow z_{j_1}, \mathbf{p}] \tag{4}$$

To write such individual terms more compactly, we will use the notion that the conditioning event $m_1 \rightarrow z_{j_1}$ will cause the pose vector $\mathbf{p}$ to get updated to presumably a more accurate pose vector that we denote $\mathbf{p_1}$ whose statistics

are denoted $(\bar{\mathbf{p}}_\mathbf{1}, \Sigma_{\mathbf{p_1}})$. The subscript in $\mathbf{p_1}$ signifies the revised statistics of $\mathbf{p}$. Therefore, the conditioning event $m_1 \rightarrow z_{j_1}, \mathbf{p}$ in Eq. (4) can be replaced by the updated vector $\mathbf{p_1}$ which is given by the previous feature matching event $m_1 \rightarrow z_{j_1}$.

Applying the same argument to all terms in Eq. (3), it can be rewritten as follows:

$$\begin{aligned} P_{match} = prob[m_1 \rightarrow z_{j_1}, \cdots, m_N \rightarrow z_{j_N}|\mathbf{p}] = \\ prob[m_1 \rightarrow z_{j_1}|\mathbf{p_0}] \times prob[m_2 \rightarrow z_{j_1}|\mathbf{p_1}] \times \\ \cdots \times prob[m_N \rightarrow z_{j_N}|\mathbf{p_{N-1}}] \end{aligned} \quad (5)$$

where $\mathbf{p_0} = \mathbf{p}$ is the initial pose given at the beginning of the pose estimation procedure. The maximization of this probability is now interpreted as a sequential search of z_{j_i} for all m_i that maximizes the probability. The detailed presentation of this search process will be given in Section III. In the following subsections, we will focus on an extended Kalman filter based algorithm that updates $\mathbf{p_i}$ when m_i is matched with z_{j_i}.

A. Incremental Pose Update Equations

In this section, we present equations that transform the pose vector $\mathbf{p_{i-1}}$ into $\mathbf{p_i}$ with new statistics when a scene edge z_{j_i} is matched with a model edge m_i. This update amounts to estimating $\mathbf{p_i}$ given the matching event $m_i \rightarrow z_{j_i}$ with $\mathbf{p_{i-1}}$ as its initial state. Assuming that $\mathbf{p_i}$ has Gaussian distribution, the best estimate of the vector $\mathbf{p_i}$ is given by its conditional mean $\bar{\mathbf{p}}_\mathbf{i} = E[\mathbf{p_i}|m_i \rightarrow z_{j_i}, \mathbf{p_i}]$ along with its covariance $\Sigma_{\mathbf{p_i}} = Cov[\mathbf{p_i}|m_i \rightarrow z_{j_i}, \mathbf{p_i}]$. The Kalman filter is known to be the optimal tool that estimates a Gaussian state variable that is conditioned by an observable measurement which, in this case, is the matching event $m_i \rightarrow z_{j_i}$.

Let us now consider what the matching event $m_i \rightarrow z_{j_i}$ practically means. Recall that our goal is to estimate $\bar{\mathbf{p}}_\mathbf{i}$ that minimizes the error between z_{j_i} and m_i when the target is at the pose $\mathbf{p_i}$. Ideally, this means that z_{j_i} is equal to the projection of m_i with the target at the pose $\mathbf{p_i}$, which we will denote by the following equation for a moment:

$$f(\mathbf{p_i}, m_i, z_{j_i}) = 0 \quad (6)$$

The detailed derivation of this equation is presented in Section II-B. In reality, however, this equation does not hold because of various feature measurement errors.[1] Let the vector $\hat{z}_{j_i}$ represent the measured parameter of z_{j_i}. We assume that the feature measurement error is additive white Gaussian, as incorporated in the following equation:

$$\hat{z}_{j_i} = z_{j_i} + \xi_{j_i} \quad (7)$$

where ξ_{j_i} is the feature measurement error with the following error statistics:

$$E[\xi_{j_i}] = 0 \quad (8)$$

[1]There are mainly three sources of measurement error. The first is the error induced by image quantization and camera distortion. The second is the fragmentation of straight edges that is caused by partial occlusion or poor edge detection. We will present details of these measurement errors and the way to handle these errors in Section II-C. The third is the error from feature mismatch. We will describe how we handle this error in Section III.

$$E[\xi_{j_i}\xi_{j_i}^T] = V_{j_i} \quad (9)$$

$$E[\xi_{j_i}\xi_{j_k}^T] = 0 \quad for \;\; j_i \neq j_k \quad (10)$$

By linearizing Eq. (6) in the vicinity of $\bar{\mathbf{p}}_\mathbf{i-1}$ and $\hat{z}_{j_i}$ using Taylor's series expansion, we can get the following equation:

$$f(\bar{\mathbf{p}}_\mathbf{i-1}, m_i, \hat{z}_{j_i}) + \frac{\partial f}{\partial z}(z_{j_i} - \hat{z}_{j_i}) + \frac{\partial f}{\partial \mathbf{p}}(\mathbf{p_i} - \bar{\mathbf{p}}_\mathbf{i-1}) = \mathbf{0} \quad (11)$$

This equation may now be rewritten in the following fashion that is standard to extended Kalman filter development:

$$y_i = M_i \mathbf{p_i} + u_{j_i} \quad (12)$$

where

$$y_i = -f(\bar{\mathbf{p}}_\mathbf{i-1}, m_i, \hat{z}_{j_i}) + \frac{\partial f}{\partial \mathbf{p}}\bar{\mathbf{p}}_\mathbf{i-1} \quad (13)$$

$$M_i = \frac{\partial f}{\partial \mathbf{p}} \quad (14)$$

$$u_{j_i} = \frac{\partial f}{\partial z}(z_{j_i} - \hat{z}_{j_i}) \quad (15)$$

The statistics of the vector u_i are easily obtained from Eq. (8) through Eq. (10) and are given by:

$$E[u_{j_i}] = 0 \quad (16)$$

$$U_{j_i} = E[u_{j_i}, u_{j_i}^T] = \frac{\partial f}{\partial z} V_{j_i} \frac{\partial f}{\partial z}^T \quad (17)$$

In all the above equations, the partial derivatives $\frac{\partial f}{\partial z}$ and $\frac{\partial f}{\partial \mathbf{p}}$ are calculated at the point $(m_i, \hat{z}_{j_i}, \bar{\mathbf{p}}_\mathbf{i-1})$.

Using the extended Kalman filter theory, the statistics of the state vector $\mathbf{p_i}$ are updated by the following equations:

$$\bar{\mathbf{p}}_\mathbf{i} = \bar{\mathbf{p}}_\mathbf{i-1} + K_i(y_i - M_i\bar{\mathbf{p}}_\mathbf{i-1}) = \bar{\mathbf{p}}_\mathbf{i-1} - K_i f(\bar{\mathbf{p}}_\mathbf{i-1}, m_i, \hat{z}_{j_i}) \quad (18)$$

$$K_i = \Sigma_{\mathbf{p_{i-1}}} M_i^T (U_{j_i} + M_i \Sigma_{\mathbf{p_{i-1}}} M_i^T)^{-1} \quad (19)$$

$$\Sigma_{\mathbf{p_i}} = (I - K_i M_i)\Sigma_{\mathbf{p_{i-1}}} \quad (20)$$

B. Constraint Equation

Eq. (6) represents the projection of m_i to the input scene when the target is at the pose $\mathbf{p_i}$. Since this projection is closely related to the pose of the target, the model edge and the camera projection model, we will first present the detailed definitions of these terms as used in this paper.

The pose of the target object in our system is defined as the homogeneous transformation of the target object coordinate frame T with respect to the input camera coordinate frame C. We denote the target pose $\mathbf{p_i}$ as six-dimensional vector $(t_i^x, t_i^y, t_i^z, \phi_i^x, \phi_i^y, \phi_i^z)^T$, where t^x, t^y, t^z are the translation and ϕ^x, ϕ^y, ϕ^z are the Euler III angles of the rotation along the three axes x, y, z of the camera coordinate frame respectively. We denote the homogeneous transformation matrix that is associated with this vector as ${}^{\mathbf{C}}\mathbf{H}(\mathbf{p_i})_\mathbf{T}$.

The target object coordinate frame is a reference frame that is placed in the center of the target object model. All the model edges are defined with respect to it. Model edges are represented with the Cartesian coordinates of their two

extremities. We denote a model edge m_i as two of three dimensional vectors $m_i^k = (x_i^k, y_i^k, z_i^k)^T,\ k = 1, 2$ which are the 3D Cartesian coordinates of the two extremities with respect to the target object coordinate frame. Consequently, the matching scene edge z_{j_i} is denoted in a similar manner as two of two-dimensional vectors $(u_{j_i}^k, v_{j_i}^k),\ k = 1, 2$ which are the 2D image coordinates of the projection of the corresponding model edge m_i.

The camera that is used for the input image sequences is calibrated using pin-hole camera model [12], [13]. Using the perspective projection model, the projection of m_i with respect to $\mathbf{p_i}$ is denoted by the following equation:

$$\begin{pmatrix} u_{m_i}^k W \\ v_{m_i}^k W \\ W \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} {}^{\mathbf{C}}\mathbf{H}(\mathbf{p_i})_{\mathbf{T}} \begin{pmatrix} x_i^k \\ y_i^k \\ z_i^k \\ 1 \end{pmatrix} \tag{21}$$

where $k = 1, 2$ and $\alpha_u, \alpha_v, u_0, v_0$ are the intrinsic parameters that are given by the camera calibration. For convenience of notation, we will name the rows of homogeneous transformation matrix ${}^{\mathbf{C}}\mathbf{H}(\mathbf{p_i})_{\mathbf{T}}$ as follows:

$${}^{\mathbf{C}}\mathbf{H}(\mathbf{p_i})_{\mathbf{T}} = \begin{pmatrix} \vec{\mathbf{h}}_1 \\ \vec{\mathbf{h}}_2 \\ \vec{\mathbf{h}}_3 \\ \vec{\mathbf{h}}_4 \end{pmatrix} \tag{22}$$

Then, the normalized image coordinates of the projection of m_i can be written as follows:

$$\tilde{u}_{m_i}^k = \frac{u_{m_i}^k - u_0}{\alpha_u} = \frac{\vec{\mathbf{h}}_1 m_i^k}{\vec{\mathbf{h}}_3 m_i^k}, k = 1, 2 \tag{23}$$

$$\tilde{v}_{m_i}^k = \frac{v_{m_i}^k - v_0}{\alpha_v} = \frac{\vec{\mathbf{h}}_2 m_i^k}{\vec{\mathbf{h}}_3 m_i^k}, k = 1, 2 \tag{24}$$

Using equations (23) and (24), Eq. (6) is represented in the following manner:

$$f(\mathbf{p_i}, m_i, z_{j_i}) = \begin{pmatrix} \frac{\vec{\mathbf{h}}_1 m_i^1}{\vec{\mathbf{h}}_3 m_i^1} - \frac{u_{j_i}^1 - u_0}{\alpha_u} \\ \frac{\vec{\mathbf{h}}_2 m_i^1}{\vec{\mathbf{h}}_3 m_i^1} - \frac{v_{j_i}^1 - v_0}{\alpha_v} \\ \frac{\vec{\mathbf{h}}_1 m_i^2}{\vec{\mathbf{h}}_3 m_i^2} - \frac{u_{j_i}^2 - u_0}{\alpha_u} \\ \frac{\vec{\mathbf{h}}_2 m_i^2}{\vec{\mathbf{h}}_3 m_i^2} - \frac{v_{j_i}^2 - v_0}{\alpha_v} \end{pmatrix} = 0 \tag{25}$$

C. Scene Feature Measurement Uncertainty

Detected edges in the input scene are often fragmented because of either image noise or occlusion. Since we use the coordinates of the edge extremities to define the matching error between a projected model edge and a scene edge, it becomes ambiguous as to how to update the target pose when the match is with a fragmented portion of an actual scene edge. In our system, this problem is handled conveniently by assigning measurement uncertainties to the extremities of the extracted scene edges in the following manner.

Recall that in Eq. (7) we used the notation $\hat{z}_{j_i}$ to denote the scene edge fragment that corresponds to the real scene edge z_{j_i} and that gets matched with the model edge m_i. If z_{j_i} was fragmented yet correctly matched with m_i, we can assume that the locations of the extremities of z_{j_i} are placed somewhere in the extension of $\hat{z}_{j_i}$. Since the length of $\hat{z}_{j_i}$ should be roughly the same as that of m_i, we would want the uncertainty ellipses to be of such a size so that the extremities of z_{j_i} lie within these ellipses as centered at the extremities of $\hat{z}_{j_i}$. Each such uncertainty ellipse will be denoted by (l_1, l_2) where l_1 is the length of the semi-major axis and l_2 the length of the semi-minor axis (see Fig. 3). We obviously want l_1 to be proportional to the length difference between the projection of m_i and $\hat{z}_{j_i}$. With regard to a value for l_2, its purpose is to account for the measurement errors in a direction perpendicular to the orientation of a line. Such errors are usually small and arise on account of sampling effects. In our experiments, we have chosen a value of 4 pixels for l_2. In general, this value would depend on the imaging geometry and image sampling rates.

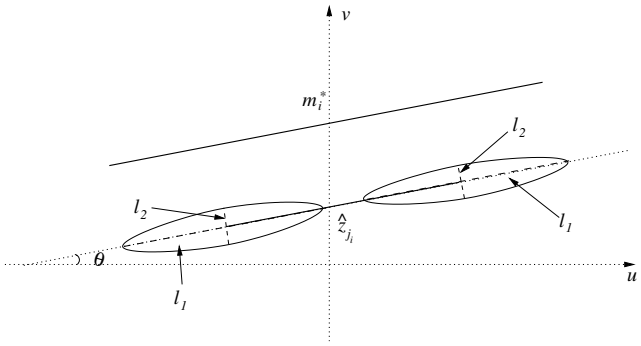


Fig. 3. Feature measurement uncertainty.

With the extremity positional uncertainties l_1, l_2 and with the slope of $\hat{z}_{j_i}$ denoted as θ given, the measurement uncertainty $V_{j_i}^k$ can be calculated by considering l_1, l_2 as its eigenvalues and θ and $\theta + \pi/2$ as the slopes of the corresponding eigenvectors respectively as follows:

$$V_{j_i}^k = \Phi \Lambda^2 \Phi^T \tag{26}$$

where

$$\Phi = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}, \Lambda = \begin{pmatrix} l_1 & 0 \\ 0 & l_2 \end{pmatrix} \tag{27}$$

III. Feature Matching Algorithm

In this section, we explain how we find the match between model edges and scene edges. Two modules are used for this purpose: One is the search module that searches the best candidate scene edges given an initial pose of the target object. The other is the verification module that verifies a match that is found by the search module using the final estimated pose. This verification module provides decision if the search module has to backtrack to find other matches. Details of these two modules will be explained in the following subsections.

A. Search Module

As presented in Section II, all model edges are sequentially matched with corresponding scene edges to maximize the conditional probability that is given by Eq. (5). This means that the statistics of the target pose $\mathbf{p_{i-1}}$ which was updated by the $i-1$th match constrains the match for m_i. We will denote the projection of m_i as m_i^* which is a random vector whose statistics can be calculated as follows: Let $(\bar{u}^1_{m_i}, \bar{v}^1_{m_i})$ and $(\bar{u}^2_{m_i}, \bar{v}^2_{m_i})$ be the mean of the pixel coordinates of m_i^*. Then, these mean values are given by the following equation:

$$\begin{pmatrix} \bar{u}^k_{m_i}\bar{W}^k_i \\ \bar{v}^k_{m_i}\bar{W}^k_i \\ \bar{W}^k_i \end{pmatrix} = (Int_C)\,{}^{\mathbf{C}}\mathbf{H}(\bar{\mathbf{p}}_{\mathbf{i-1}})_{\mathbf{T}} \begin{pmatrix} x^k_i \\ y^k_i \\ z^k_i \\ 1 \end{pmatrix}, k = 1, 2 \tag{28}$$

where (x^k_i, y^k_i, z^k_i), $k = 1, 2$ are the 3D coordinates of the extremities of m_i and Int_C is the matrix of intrinsic camera calibration parameters that is given in Eq. (21). The uncertainty of m_i^*, which is denoted as $\Sigma_{m_i^*}$ can be calculated as follows:

$$\Sigma_{m_i^*} = J(m_i^*, \mathbf{p_{i-1}})\Sigma_{\mathbf{p_{i-1}}}J(m_i^*, \mathbf{p_{i-1}})^T \tag{29}$$

where $J(m_i^*, \mathbf{p_{i-1}})$ is a Jacobian matrix that can be easily derived from Eq. (25) as follows:

$$J(m_i^*, \mathbf{p_{i-1}}) = \frac{\partial f(\mathbf{p_{i-1}}, m_i, z_{ji})}{\partial \mathbf{p_{i-1}}} \tag{30}$$

The projected uncertainty $\Sigma_{m_i^*}$ serve as the search region in which the matching scene feature for m_i^* is searched. Extracted straight-line edges in this search region are considered as candidate scene features.

There may be more than one candidate matching scene feature present in the search region. In this case, we select the scene edge that has smallest Mahalanobis distance to m_i^*. Note that the constraint equation that was defined in Eq. (25) represent the pixel displacements between m_i^* and $\hat{z}_{j_i}$. Using this constraint equation, the Mahalanobis distance between the two is given as follows [14]:

$$d(m_i^*, \hat{z}_{j_i}, \bar{p}_{i-1}) = f(m_i, \bar{z}_{j_i}, \bar{\mathbf{p}}_{\mathbf{i-1}})^T \Sigma_f f(m_i, \bar{z}_{j_i}, \bar{\mathbf{p}}_{\mathbf{i-1}}) \tag{31}$$

where $\bar{z}_{j_i}$ is the mean of $\hat{z}_{j_i}$ and Σ_f is the uncertainty of f which is given by the following equation:

$$\Sigma_f = \frac{\partial f}{\partial \mathbf{p_{i-1}}}^T \Sigma_{\mathbf{p_{i-1}}} \frac{\partial f}{\partial \mathbf{p_{i-1}}} + \frac{\partial f}{\partial z_{j_i}}^T \Sigma_{\hat{z}_{j_i}} \frac{\partial f}{\partial z_{j_i}} \tag{32}$$

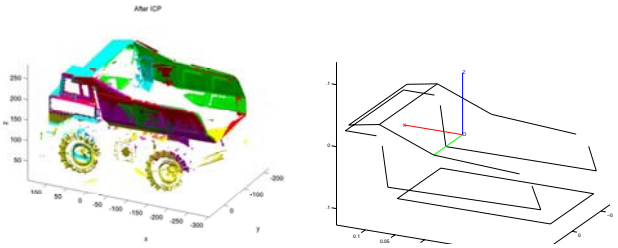
Also, there may be no candidate appropriate for match because of occlusion. This case is handled by allowing nil-match for a model edge if it does not have any candidate scene edge in the search region.

B. Verification Module

We use a *test of mean* method to decide if the match given by the search module is acceptable, i.e., if any model edge was matched with a wrong scene edge that makes large pose estimation error. Recall that the evaluation of the constraint equation $f(m_i, \hat{z}_{j_i}, \mathbf{p_N})$ represents the pixel displacements between the projection of m_i with respect to the final estimated pose $\mathbf{p_N}$ and $\hat{z}_{j_i}$. If this match is wrong with respect to $\mathbf{p_N}$, then the Mahalanobis distance $d(m_i, \hat{z}_{j_i}, \mathbf{p_N})$ that was defined in Eq. (31) should be significantly larger than zero. Hence, we test if the sum of the Mahalanobis distance between all model edges and their corresponding scene edges with respect to $\mathbf{p_N}$ have zero mean. For all model features that are not nil-matched, we calculate the following sum of distances

$$D = \sum_{i \in Q} f(m_i, \bar{z}_{j_i}, \bar{\mathbf{p}}_{\mathbf{N}})^T \Sigma_f^{-1} f(m_i, \bar{z}_{j_i}, \bar{\mathbf{p}}_{\mathbf{N}}) \tag{33}$$

where Q is the set of indices of model features that have non-nil matches. Since $f(m_i, \bar{z}_{j_i}, \bar{\mathbf{p}}_{\mathbf{N}})$ has Gaussian distribution, D has chi-squared distribution with $4(N-q)$ degrees of freedom, where q is the number of nil-matches. Hence, we can decide if the search module should backtrack by comparing D with a threshold that is set appropriately considering the distribution of D.



(a) Rendered range data (b) Wire-frame model

Fig. 4. Model of a target object.

IV. Modelling the Target Object

The 3D models of target objects are represented by approximated polyhedra that consist of straight edges in Cartesian space. In constructing 3D model of an object for pose estimation, short line segments are not favored because their accurate localization is vulnerable to noise in the input image. Furthermore, our system can estimate accurate pose of an object by matching a small number of features. Hence, only a small number of long line segments are selected for 3D models of target objects. The line-segments that are used for object models were delineated off-line by a human from 3D range data of target objects which were acquired by structured-light scanner [15]. An example picture of the rendered range data image of a target object is depicted in Fig. 4-(a), and its corresponding wire-frame model in Fig. 4-(b) where the three colored axes represent the object-centered target coordinate frame.

Since our system tracks a target object that undergoes unlimited degrees of rotation, there are two other groups of line segments that are not suitable for feature matching depending on the viewing direction. One is the group of self-occluded line segments. The Binary Space Partitioning tree [16] is used to remove self-occluded line segments at a specific viewing angle. The other is that of line segments that are parallel to the viewing angle. The lines that are

parallel to the viewing angle are projected to a point on an input scene, hence it cannot be used for line segment match. These lines are also removed for feature matching.

V. Experimental Results

A. Pose Estimation Accuracy Analysis

In order to analyze the pose estimation accuracy, we need actual pose – ground truth – of the target object when the target moves in the vicinity of the camera. However, measuring the ground truth pose of the target object while it moves is virtually impossible. Instead, we fixed the location of the target object and captured the input scenes when the camera is at known positions. To do so, we mounted the camera on a robot end-effector and moved it in three arbitrary paths that was defined on the surface of imaginary spheres of radius 750, 650, and 550 mm respectively. The pose of the camera is calculated from the robot kinematics that can be acquired from the robot controller and camera hand-eye calibration. The ground truth pose of the target can be easily calculated from the pose of the target at the first scene and the relative pose of the camera at each consecutive position where the input scene was captured. The pose of the target at the first scene was measured visually by aligning the projection of model to the scene.

The target object that is used for this experiment is a toy truck whose dimension is 41 cm (length) $\times$ 25 cm (height) $\times$ 20 cm (width). Table I shows the statistics of the error between the ground truth pose of the target and the estimated pose. One can clearly see the error in

Distance (mm)	Stat	X	Y	Z	Yaw	Pitch	Roll
		(mm)			($degree$)		
750	abs mean	2.3	2.9	6.7	1.0	1.7	0.4
	std	1.6	1.8	3.9	0.6	1.2	0.3
	abs max	6.9	7.6	19.5	2.9	5.6	1.3
650	abs mean	1.5	2.8	8.5	0.5	1.5	0.5
	std	1.2	1.7	4.7	0.4	1.2	0.3
	abs max	5.2	7.6	25.7	2.0	4.4	1.7
550	abs mean	1.6	4.1	7.3	0.5	1.3	0.3
	std	1.0	2.5	4.8	0.4	0.9	0.3
	abs max	5.1	11.4	24.2	2.1	4.9	1.3

TABLE I
Statistics of pose estimation error

Z component of the target object pose, which represents the distance between the target and the camera, is larger than the others. It is because the depth is estimated by considering the scale of the target in the input scene since we use monocular image. This is, of course, an inherent problem in monocular vision. We believe this error can be reduced by using a multiple camera system.

B. Tracking Results

Three image sequences with complex background (*Complex*), occlusion (*Occlusion*) and non-smooth object motion with large change of viewing aspect (*Non-smooth motion*) were captured to test the tracking system. Since it was very difficult to get the ground truth for these sequences, we provide only qualitative results by presenting the movies that show the tracking results.[2] Some example pictures that show the pose estimation results for the sequences are depicted in Fig. 5.

C. Pose Estimation Performance Analysis

The proposed tracking system is tested on a personal computer with AMD Athelon 2200+ 2.0 GHz processor and 512 Mb system memory. For the three test sequences, the average pose estimation time was 0.033 second which was close to the frame rate but the average straight edge extraction time was 0.32 second. The straight edge extraction process includes the median filtering of the input scene which is needed because we use a interlaced-scan camera that captures the edges of moving object cluttered. We are working on improving the performance by using progressive-scan camera for input scenes and employing signal processing hardware.

The performance statistics for the three sequences are presented in Table II. Fig. 6 depicts the plots of processing time for the three test sequences. As can be seen from Fig. 6, pose estimation process takes quite significant amount of time for some frames. This is because the feature matching process currently backtracks indefinitely until it finds a match that breaks the backtracking condition in Eq. (33). We believe this situation can be avoided by setting a limit in backtracking or pruning the search process.

Image Seq.	Pose Estimation (seconds)		Feature Extraction (seconds)	
	mean	std	mean	std
Complex	0.040	0.096	0.317	0.017
Occlusion	0.042	0.137	0.326	0.013
Non-smooth motion	0.016	0.022	0.318	0.012

TABLE II
Statistics of tracking performance

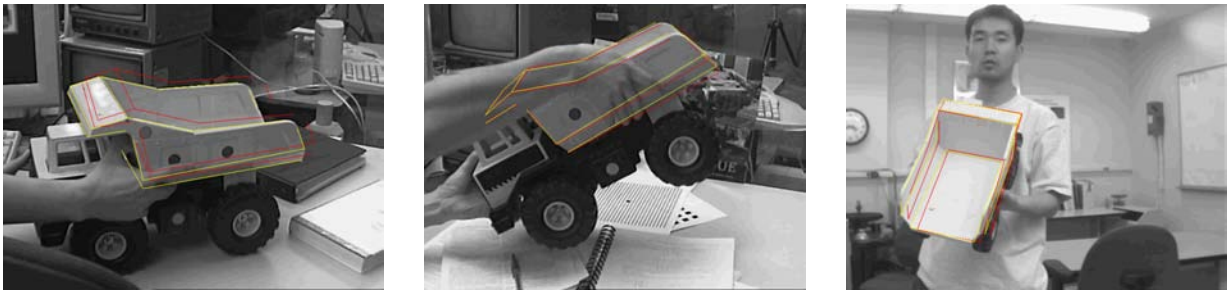
VI. Conclusion and Future Works

In this paper, we proposed a new approach to the use of edge extremities in a generic model-based object tracking framework. Using the proposed approach, the extended Kalman filter that is used for pose estimation can handle edge fragmentation effectively by assigning measurement uncertainties to the extremities of extracted edges.

The pose estimation error analysis shows that our tracking system can give accurate pose of a target object while tracking. Also, our tracking experiment shows that our system can track the target object successfully under various conditions – such as occlusion, complex background and non-smooth motion with large change of viewing aspect of the target.

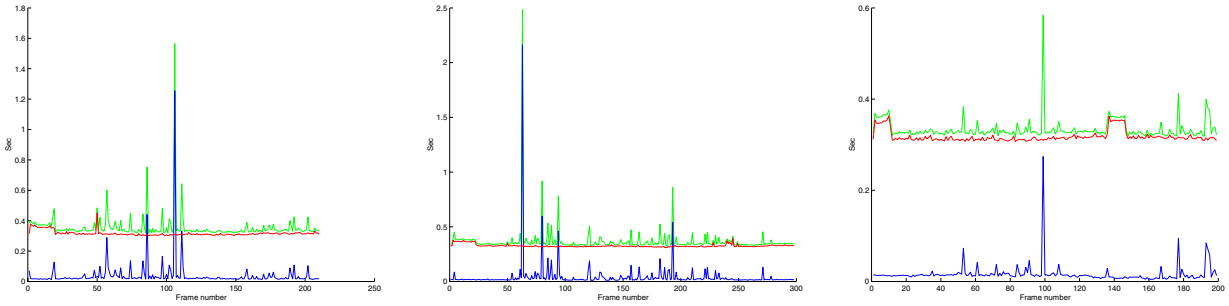
There are two issues still remain to be solved. One is related to the fact that the slow feature extraction process is a significant bottleneck in the overall tracking procedure.

[2]Movies that show the tracking results are posted on the following URL:
http://rvl1.ecn.purdue.edu/RVL/Projects/ModelBasedTracking/



(a) *Complex* (b) *Occlusion* (c) *Non-smooth motion*

Fig. 5. Example images of pose estimation results for the three test image sequences. Red plots show the initial pose of the target and yellow plots show the estimated pose.



(a) *Complex* (b) *Occlusion* (c) *Non-smooth motion*

Fig. 6. Processing time for the three sequences. Red plots show the feature extraction time, blue plots the pose estimation time and green plots the total elapsed time.

Since the feature extraction process mostly consists of low-level image processing, we are currently working on enhancing the feature extraction performance with the help of signal processing hardware. The other is that our backtracking criterion can slow down the pose estimation process although it can handle nil-matches effectively. To solve this problem, we are currently investigating the possibility of limiting the extent of backtracking and devising an appropriate pruning procedure for the elimination of inapplicable matches.

REFERENCES

[1] G. N. DeSouza and A. C. Kak, "A subsumptive, hierarchical, and distributed vision-based architecture for smart robotics," *IEEE Trans. Systems, Man and Cybernetics*, vol. 34, no. 5, pp. 1988–2002, 2004.

[2] F. Martin and R. Horaud, "Multiple-camera tracking of rigid objects," *Int'l J. Robotics Research*, vol. 21, no. 2, pp. 97–113, Feb. 2002.

[3] T. Drummond and R. Cipollar, "Real-time visual tracking of complex structures," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 932–946, July 2002.

[4] T. N. Tan, G. D. Sullivan, and K. D. Baker, "Pose determination and recognition of vehicles in traffic scenes," in *Proc. the 1994 European Conf. Computer Vision*, May 1994, pp. 501–506.

[5] E. Marchand, P. Bouthemy, and F. Chaumette, "A 2d-3d model-based approach to real-time visual tracking," *Image and Vision Computing*, vol. 19, no. 13, pp. 941–955, Nov. 2001.

[6] F. Jurie, "Tracking objects with a recognition algorithm," *Pattern Recognition Letters*, vol. 19, no. 3-4, pp. 331–340, 1998.

[7] P. Mittrapiyanuruk, G. N. DeSouza, and A. C. Kak, "Calculating the 3d-pose of rigid objects using active appearance models," in *Proc. the 2004 IEEE Int'l Conf. Robotics and Automation*, pp. 5147–5152.

[8] L. Vacchetti, V. Lepetit, and P. Fua, "Stable real-time 3d tracking using online and offline information," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1385–1391, 2004.

[9] D. G. Lowe, "Robust model-based motion tracking through the integration of search and estimation," *Int'l J. Computer Vision*, vol. 8, no. 2, pp. 113–122, 1992.

[10] A. Kosaka and A. C. Kak, "Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties," *Computer Vision, Graphics and Image Processing:Image Understanding*, vol. 56, no. 3, pp. 271–329, 1992.

[11] A. Kosaka and G. Nakazawa, "Vision-based motion tracking of rigid objects using prediction of uncertainties," in *Proc. 1995 IEEE Int'l Conf. Robotics and Automation, Nagoya, Japan*, 1995.

[12] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, 1992.

[13] Z. Zhang, "A flexible new technique for camera calibration," Microsoft Research, Tech. Rep. MSR-TR-98-71, Mar. 1999.

[14] N. Ayache, *Artificial Vision for Mobile Robots*. MIT Press, 1991, ch. 11, pp. 187–189.

[15] J. Park and A. C. Kak, "Multi-peak range imaging for accurate 3d reconstruction of specular objects," in *Proc. the Sixth Asian Conf. Computer Vision*, Jan. 2004.

[16] M. Paterson and F. Yao, "Efficient binary space partitions for hidden surface removal and solid modeling," *Discrete and Computational Geometry*, vol. 5, no. 5, pp. 279–304, 1990.