# Integration of Model-based and Model-free Cues for Visual Object Tracking in 3D*

Ville Kyrki
*Laboratory of Information Processing*
*Lappeenranta University of Technology*
*P.O. Box 20, FIN-53851 Lappeenranta, Finland*
*ville.kyrki@lut.fi*

Danica Kragic
*Centre for Autonomous Systems*
*Royal Institute of Technology*
*S-100 44 Stockholm, Sweden*
*danik@nada.kth.se*

***Abstract*— Vision is one of the most powerful sensory modalities in robotics, allowing operation in dynamic environments. One of our long-term research interests is mobile manipulation, where precise location of the target object is commonly required during task execution. Recently, a number of approaches have been proposed for real-time 3D tracking and most of them utilize an edge (wireframe) model of the target. However, the use of an edge model has significant problems in complex scenes due to occlusions and multiple responses, especially in terms of initialization.**

**In this paper, we propose a new tracking method based on integration of model-based cues with automatically generated model-free cues, in order to improve tracking accuracy and to avoid weaknesses of edge based tracking. The integration is performed in a Kalman filter framework that operates in real-time. Experimental evaluation shows that the inclusion of model-free cues offers superior performance.**

***Index Terms*— model-based tracking, model-free tracking, cue integration, iterated extended Kalman filter**

## I. Introduction

Real-time visual tracking of rigid 3-D objects is commonly used in a number of robotic tasks ranging from micro-manipulation to human-robot interaction. In particular, visual servo algorithms as well as grasp planners require efficient pose estimation and tracking. Our application focus is on robotic manipulation using eye-in-hand systems. In applications such as object manipulation, the tracking is typically model-based, because the grasping can only be performed after aligning the manipulator and the object precisely (if no additional sensory modalities are available). Thus, the absolute pose of the object with respect to the manipulator-mounted camera needs to be recovered.

A number of successful systems have been proposed, e.g., [1]–[3]. which are based on contour tracking using a wireframe model. Nevertheless, the use of such a model is difficult when the background is complex, as it is difficult to distinguish between background and object edges, as well as multiple edges on the object itself. Tracking of textured objects can also be problematic because the "signal-to-noise ratio" is small, that is, only a fraction of the detected edges really belong to the desired outline of the object. The confusion between object and background edges is illustrated in Fig. 1, where the sequence of images on top row shows the tracking result of the Lie algebra method proposed in [3]. Here, the tracker gets stuck on a leg of a table in the background.

In this paper, we propose a solution to the 3-D tracking problem that, in addition to wireframe based tracking, uses automatically initialized model-free trackers. The additional features are used in order to address the problems related to complex scenes, while still permitting the estimation of the absolute pose using the model. The benefit of using the integration can be seen in the second row of images in Fig. 1. We show that the different types of features can be integrated in a Kalman filter framework that operates in real-time. Our application focus is on robotic manipulation and grasping, and on learning-by-demonstration systems [4].

We begin by reviewing related literature in Section II. Then, in Section III we describe the image analysis methods used. Section IV presents our approach of integrating cues using the Iterated Extended Kalman filter. Experimens are presented in Section V, and finally, the method is discussed and conclusions drawn in Section VI.

## II. Related work

Integration of vision based cues has been found to provide robustness in tracking and has been used successfully in many applications [5], [6]. Nevertheless, multiple cues have been applied mostly in image plane tracking. Only recently they have been proposed for 3D tracking [7].

While appearance based systems have been proposed for pose tracking [8], most current systems for 3D pose tracking are contour feature-based. RAPID [1] uses the dynamic vision approach presented by Dickmanns [9], which is based on the use of extended Kalman filtering to integrate image measurements through a non-linear measurement function to estimate the pose. Lowe presented a method that chains together edges, which are then matched and fitted to model [10]. Wunsch et al. perform iterative minimization to find the pose transformation that aligns a set of lines and ellipses, after which the pose is integrated in a linear Kalman filter over time [2]. Taylor et al. propose to integrate color, edge, and texture cues from a textured 3D model [7].

*This work was partially supported by Academy of Finland grant to V. Kyrki

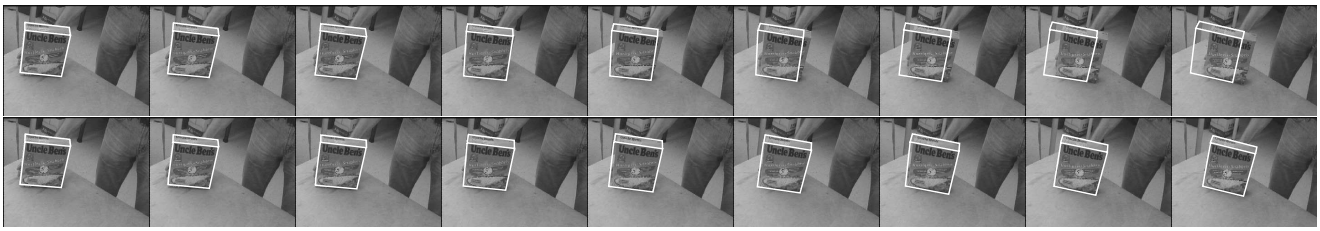

Fig. 1. Model-based features vs integration.

## III. Image feature detection

We begin this section by introducing the automatic pose initialization approach. Then, we describe the pose tracking using the Lie algebra formalism. The initialization of the automatically generated point features is presented thereafter, followed with the image plane point tracking.

### A. Model initialization

One of the problems to cope with during the initialization step is that the objects considered for manipulation are highly textured and therefore not suited for matching approaches based on, for example, line features [2], [11], [12]. The initialization step uses the SIFT point matching method proposed in [13]. Here, reference images of the object with known pose are used to perform initialization of the pose in the first image. An example of the initialization step can be seen in Figure 2.

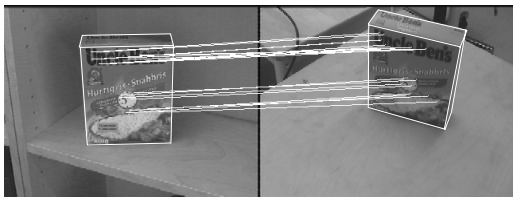

Fig. 2. Automatic initialization of the pose using a reference image and SIFT features.

### B. Model-based tracking

We shortly present the underlying wireframe tracking algorithm. Details about object modeling can be found in [14]. Let us consider the perspective camera model

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{AE} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where **A** and **E** represent the intrinsic and the extrinsic camera parameters, respectively. Image coordinates are then given by:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} u/w \\ v/w \end{bmatrix}. \tag{2}$$

During tracking, the current pose estimate is used to estimate the visibility of each edge. Then, camera parameters are used to project the model of the object onto the image plane. For each visible edge, a number of points are generated along the edge at regular intervals in image coordinates.

In each generated point $\mathbf{p}_p$ along a visible edge, the perpendicular distance $d_p^{\perp}$ to the nearby edge is determined using one-dimensional search. The search region is denoted by $\{S_p^j, j \in [-s, s]\}$. The search starts at the projected model point $\mathbf{p}_p$ and the traversal continues simultaneously in opposite search directions until the first local maximum is found. The normal direction is approximated with four directions, see Fig. 3. The size of the search region $s$ is predetermined based on the expected motion.

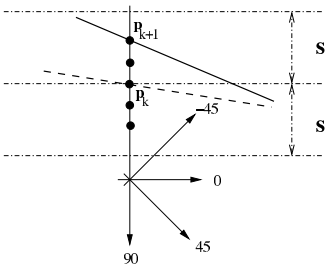


Fig. 3. Determining normal displacements for points on a contour in consecutive frames. The search direction is approximated with four directions: $\{-45, 0, 45, 90\}$ degrees.

The Lie group and Lie algebra formalism is used as the basis for representing the motion of a rigid body. Rigid body motion can be represented as a six-dimensional Lie group where the six generators of the group are the translations in the direction of $x$, $y$, and $z$ axes and rotations about them. Using $4\times4$ homogeneous matrices, the six generators of the Lie group are:

$$G_1 = \begin{bmatrix} 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix},\ G_2 = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix},\ G_3 = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{bmatrix},$$
$$G_4 = \begin{bmatrix} 0&0&0&0 \\ 0&0&-1&0 \\ 0&1&0&0 \\ 0&0&0&0 \end{bmatrix},\ G_5 = \begin{bmatrix} 0&0&1&0 \\ 0&0&0&0 \\ -1&0&0&0 \\ 0&0&0&0 \end{bmatrix},\ G_6 = \begin{bmatrix} 0&-1&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix}. \tag{3}$$

Using (1), partial derivatives of projective image coordinates under $i$-th generating motion are

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \mathbf{PG}_i \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4}$$

where **P** represents the projective camera matrix. This can be used to determine the motion that would be observed

in an image point $[x \;\; y]^T$ for $i$-th generator:

$$\mathbf{L}_i = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{u'}{w} + \frac{uw'}{w^2} \\ \frac{v'}{w} + \frac{vw'}{w^2} \end{bmatrix} \tag{5}$$

Using (5), a vector $\mathbf{L}_i^p$ describing the image motion of the $p$-th point for the $i$-th generator may be computed. Using the normal vector of the edge $\mathbf{n}^p$, $\mathbf{L}_i^p \cdot \mathbf{n}^p$ is the magnitude of the edge normal motion that would be observed in the image. $\mathbf{L}_i^p \cdot \mathbf{n}^p$ for $p = \{1, \ldots, n\}$ can be considered an $n$-dimensional vector which describes the motion in the image with respect to generator $\mathbf{G}_i$. Using the vector of measured normal displacements $\mathbf{d}$, the least squares approximator for the motion is

$$\mathbf{O}_i = \sum_p \mathbf{d}^p (\mathbf{L}_i^p \cdot \mathbf{n}^p) \tag{6}$$

$$\mathbf{C}_{ij} = \sum_p (\mathbf{L}_i^p \cdot \mathbf{n}^p)(\mathbf{L}_j^p \cdot \mathbf{n}^p) \tag{7}$$

$$\alpha_i = \sum_i \mathbf{C}_{ij}^{-1} \mathbf{O}_j \tag{8}$$

where $\alpha_i$ represent the quantity for each Euclidean motion observed.

Denoting the pose of the object by a 4×4 homogeneous matrix $\mathbf{T}$, this matrix can be written as a product of six homogeneous matrices each of them representing one of the rotations and translations. Then, the complete transformation between two consecutive frames is:

$$\Delta\mathbf{T} = \exp(\sum \alpha_i \mathbf{G}_i) \tag{9}$$

Since $\Delta\mathbf{T}$ is assumed to be small between frames, it can be approximated by a linear term as

$$\Delta\mathbf{T} \approx \mathbf{I} + \sum \alpha_i \mathbf{G}_i, \tag{10}$$

Finally, the new pose of the object relative to the camera is given by:

$$\mathbf{T}_{k+1} = \mathbf{T}_k \, \Delta\mathbf{T} \tag{11}$$

### C. Point initialization

The image plane tracker is initialized using interest points extracted using the Harris corner detector [15], which was selected because it is suitable for a real-time operation and describes well interest points of a textured surface. The current state estimate is used to only initialize points on the frontal plane of the object. After the interest points are chosen, the local neighborhood around each interest point is stored. It is sufficient to perform the corner detection every 10th frame because the rotation changes between the object and the camera frames are only moderately fast and the appearance of the local neighborhood changes gradually.

### D. Point tracking

The point tracking is based on minimizing the sum of squared differences in the RGB values, formally solving the displacement

$$\mathbf{d} = \arg\min \sum_{\mathbf{x} \in W} \|T(\mathbf{x}) - I(\mathbf{x} + \mathbf{d})\|^2, \tag{12}$$

where $T$ is the stored template, $I$ is the current image, and $W$ is the search window. This simple approach is useful because it is computationally light but still allows tracking over shorter time intervals. It should be noted that our integration method does not depend on tracking the same feature through the whole sequence of images. The tracked features are rejected on two conditions: i) if the sum of squared differences grows above a predetermined threshold, or ii) if the feature moves out of the estimated frontal plane of the object. Also, we found that an RGB template greatly outperforms the gray scale template.

## IV. Integration of motion with IEKF

The measurements of the two types of features are integrated using an iterated extended Kalman filter (IEKF). It is used instead of the basic extended Kalman filter because it is shown to perform better when the measurement functions are strongly nonlinear.

IEKF estimates the state $\mathbf{x}$ of a system by using a system model $f(\mathbf{x})$, which models the time dependencies of the system such that

$$\mathbf{x}_{i+1} = f(\mathbf{x}_i) + \mathbf{w}_i, \tag{13}$$

where $\mathbf{w}$ denotes the model error modelled as a zero-mean Gaussian random variable. In addition, a measurement model $h(\mathbf{x})$ is used to link the internal state to a set of measurable quantities $\mathbf{y}$ by

$$\mathbf{y}_i = h(\mathbf{x}_i) + \mathbf{v}_i, \tag{14}$$

where $\mathbf{v}$ is the measurement error modelled also as a zero-mean Gaussian random variable. The uncertainties are modeled with covariances. Let $\mathbf{P}$ denote the covariance of the internal state, i.e., $\mathbf{P}_i = E[\mathbf{x}_i \mathbf{x}_i^T]$, $\mathbf{Q}$ be the covariance of the model error, i.e., $\mathbf{Q}_i = E[\mathbf{w}_i \mathbf{w}_i^T]$, and $\mathbf{S}$ be the covariance of the measurement error $\mathbf{S}_i = E[\mathbf{v}_i \mathbf{v}_i^T]$. It is also assumed that the errors are uncorrelated over time, i.e., $E[\mathbf{w}_i \mathbf{w}_k] = 0, E[\mathbf{v}_i \mathbf{v}_k] = 0, \forall i \neq k$. It is also assumed that measurement and model errors are uncorrelated, i.e., $E[\mathbf{w}_i \mathbf{v}_k] = 0, \forall i, k$.

The IEKF estimation consists of two steps. First, in the prediction step, the evolution of the system is predicted using the system model by

$$\mathbf{x}_{i+1|i} = f(\mathbf{x}_i). \tag{15}$$

The covariance is updated according to

$$\mathbf{P}_{i+1|i} = \mathbf{F}_i \mathbf{P}_i \mathbf{F}_i^T + \mathbf{Q}_i, \tag{16}$$

where $\mathbf{F}_i$ is the gradient of $f(\cdot)$ evaluated at $\mathbf{x}_i$, i.e., $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\big|_{\mathbf{x}=\mathbf{x}_i}$.

The update step iterates two calculations: The Kalman gain is first computed according to

$$\mathbf{K}_{i+1} = \mathbf{P}_{i+1|i} \mathbf{H}_{i+1}^{k^T} \left( \mathbf{H}_{i+1}^k \mathbf{P}_{i+1|i} \mathbf{H}_{i+1}^{k^T} + \mathbf{S}_{i+1} \right)^{-1}, \tag{17}$$

where $\mathbf{H}_i^k$ is the gradient of the measurement function $h(\cdot)$ evaluated at $\mathbf{x}_{i+1}^k, \mathbf{x}_{i+1}^0 = \mathbf{x}_{i+1|i}$. Then, the state is updated according to

$$\mathbf{x}_{i+i}^{k+1} = \mathbf{x}_{i+1}^k + \mathbf{K}_{i+1} \left( \mathbf{z}_i - h(\mathbf{x}_{i+1}^k) \right). \tag{18}$$

Eqs. (17) and (18) are iterated until the change in $\mathbf{x}_{i+1}^k$ falls below a threshold. Then, the state is updated to the final estimate, i.e., $\mathbf{x}_{i+1} = \mathbf{x}_{i+1}^N$ and the state covariance is updated by

$$\mathbf{P}_{i+1} = \mathbf{P}_{i+1|i} - \mathbf{K}_{i+1}\mathbf{H}_i^N\mathbf{P}_{i+1|i}. \quad (19)$$

### A. System models

Our model uses the 6-DOF pose vector of the tracked object as the system state, i.e., $\mathbf{x} = (X, Y, Z, \phi, \theta, \psi)^T$. Due to the well-known problems with the non-uniqueness of Euler angles [7], we adopt the approach proposed in [16], where the orientation is represented externally, outside the Kalman filter state, and the angles $\phi$, $\theta$ and $\psi$ only describe incremental changes. Unlike their approach of using a quaternion, we represent the external orientation using a rotation matrix. Thus, at each time step the rotation angles are integrated into matrix $\mathbf{R}_0$ and reset to zero.

We now define two zeroth order models of motion. We use zeroth order models, i.e., we do not have any velocity model, to really investigate only the integration issue. The difference between the models is the center point of the rotation. In the first (Model 1), the object rotates around its own origin, which corresponds to moving object and stationary camera. In the second (Model 2), the rotation is around the camera frame origin, corresponding to moving camera and stationary object. We want to use both models in order to study how much the choice of a model affects the tracking accuracy. Both models are linear, but have difference in the prediction covariance $\mathbf{Q}$. Both models predict the internal state according to $\mathbf{x}_{i+1|i} = \mathbf{x}_i$. Thus, the gradient of the state update is the identity matrix, $\mathbf{F} = \mathbf{I}_6$.

The motion is modelled as a Wiener process, with independent uncorrelated noise sources for both translational and rotational motion. Thus, for Model 1, where the motion is with respect to the object origin, the state prediction covariance is

$$\mathbf{Q}_1(\Delta t) = \begin{pmatrix} \Delta t\,\sigma_p^2\,\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \Delta t\,\sigma_\phi^2\,\mathbf{I}_3 \end{pmatrix} \quad (20)$$

where $\Delta t$ is the length of the time step, and $\sigma_p^2$ and $\sigma_\phi^2$ are the variances of the translational and rotational motion, respectively. In Model 2, the translation is affected by the rotation around the camera center and the system can be written as

$$\begin{aligned} \mathbf{p}_{i+1} &= \mathbf{R}(\mathbf{w}_\phi)\mathbf{p}_i - \mathbf{w}_p \\ \boldsymbol{\phi}_{i+1} &= \boldsymbol{\phi}_i - \mathbf{w}_\phi \end{aligned} \quad (21)$$

where $\mathbf{w}_p$ and $\mathbf{w}_\phi$ are the noise sources for camera translation and rotation. Writing $\mathbf{R}(\mathbf{w}_\phi, \mathbf{p}_i) \equiv \mathbf{R}(\mathbf{w}_\phi)\mathbf{p}_i$, we can then approximate it with a first order Taylor series as

$$\mathbf{R}(\mathbf{w}_\phi, \mathbf{p}) = \mathbf{R}(\mathbf{0}, \mathbf{p}) + \frac{\partial \mathbf{R}(\mathbf{w}_\phi, \mathbf{p})}{\partial \mathbf{w}_\phi}\mathbf{w}_\phi = \mathbf{p} + \mathbf{A}\mathbf{w}_\phi \quad (22)$$

where

$$\mathbf{A} = \begin{pmatrix} 0 & p_z & -p_y \\ -p_z & 0 & p_x \\ p_y & -p_x & 0 \end{pmatrix}. \quad (23)$$

(21) and (22) allow us to write $\mathbf{Q}_2$ as

$$\mathbf{Q}_2 = \begin{pmatrix} \mathbf{Q}_{pp} & \mathbf{Q}_{p\phi} \\ \mathbf{Q}_{\phi p} & \mathbf{Q}_{\phi\phi} \end{pmatrix} \quad (24)$$

where

$$\begin{aligned} \mathbf{Q}_{pp} &= \Delta t\,\sigma_p^2\,\mathbf{I}_3 + \Delta t\,\sigma_\phi^2\,\mathbf{A}\mathbf{A}^T & \mathbf{Q}_{p\phi} &= -\Delta t\,\sigma_\phi^2\,\mathbf{A} \\ \mathbf{Q}_{\phi p} &= -\Delta t\,\sigma_\phi^2\,\mathbf{A}^T & \mathbf{Q}_{\phi\phi} &= \Delta t\,\sigma_\phi^2\,\mathbf{I}_3. \end{aligned}$$

Thus, the translation is affected by the rotation around the camera center in addition to the pure translational component.

### B. Measurement models

We have two measurement models, one for the absolute orientation acquired using the model based tracking, and another for the point measurements. The model for the direct pose measurement $h_0(\cdot)$ is linear: $h_0(\mathbf{x}) = \mathbf{x}$. It is assumed that the measurement errors are independent in different directions, and thus the covariance matrix can be written

$$\mathbf{S}_0 = \begin{pmatrix} \sigma_t^2\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \sigma_\psi^2\mathbf{I}_3 \end{pmatrix}, \quad (25)$$

where $\sigma_t^2$ and $\sigma_\psi^2$ are the variances of the translation and rotation measurements.

For the point measurements, the following simplification is made in contrast to traditional structure-from-motion approaches: Every time a new image plane tracker is initialized, its 3D coordinates in the object frame, $\mathbf{q}_j$, are calculated as the intersection of the line going through the point and the camera frame origin, and the plane defined by the model and the current pose estimate. To keep the filter state vector short and real-time computation possible, we do not incorporate the error in the depth estimate to the Kalman filter, but handle this error as part of the image measurement error.

The point measurements are represented in the camera frame. The image plane points are first transformed into camera frame using weak intrinsic calibration (orthogonal axes, optical origin at the center of the calibration array, manufacturer given focal length). Thus, our camera frame measurement function for point $i$ is

$$\begin{aligned} \begin{pmatrix} X_j & Y_j & Z_j \end{pmatrix}^T &= R(\mathbf{x})\mathbf{q}_j + t(\mathbf{x}) \\ h_j(\mathbf{x}) &= \begin{pmatrix} X_j/Z_j & Y_j/Z_j \end{pmatrix}^T + \mathbf{v}, \end{aligned} \quad (26)$$

where $R(\mathbf{x})$ is the rotation matrix taking into account the current rotation estimate described by the state and the accumulated rotation $\mathbf{R}_0$, and $t(\mathbf{x})$ is the translation component of the state. We assume that the measurement errors are independent with respect to the coordinate axes and between different measurements. Thus, the covariance matrix takes the simple form of $\mathbf{S}_j = \sigma_i^2\mathbf{I}_2$, where $\sigma_i^2$ is the image measurement error. The measurements are assumed to be independent of each other and the full-pose measurements, that is $\mathbf{S}_{ij} = \mathbf{0}, i \neq j$. The gradients of the measurement functions are calculated analytically, but not shown here because of the limited space available.

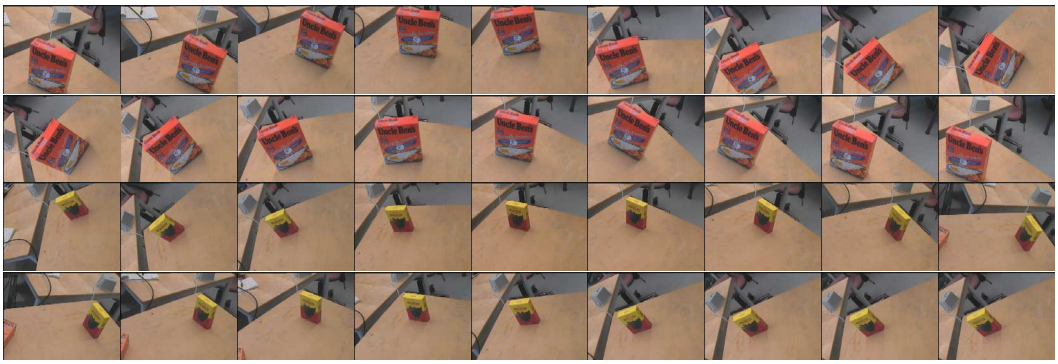

Fig. 4. Test sequences 1 and 2.

## V. Experiments

Experiments were performed on two recorded sequences to allow repeated tests. Both sequences had a different target object and can be seen in Fig. 4. The lengths of the sequences are 173 (Sequence 1) and 157 seconds (Sequence 2). The sequences were recorded by moving a camera mounted on a robot arm. Ground truth was generated by recording the robot trajectory and determining the hand-eye calibration by the method of Tsai and Lenz [17]. It should also be noted that the camera was only coarsely calibrated, i.e., the optical origin was assumed to coincide with the center pixel and the manufacturer given focal length was used.

### A. Initialization

The initialization accuracy was examined by using the initialization to find the pose for each image in Sequence 1. Initialization attempt was considered failed if the translation error exceeded 15 centimeters or if the rotation error was above 15 degrees.

The errors for successful initializations are seen in Fig. 5. Only 1.5 % of the attemps failed, and no two consequtive frames were failures. This means in practice that if a failure can be detected, it is highly probable that the next initialization attempt will be successful. The mean translation error for the successful attempts was approximately 2 cm and rotation error 4 degrees, which demonstrates that the approach is definitely valid for initializing the tracker.

### B. Tracking

Tracking accuracy was studied for both sequences and the two system models described in Sec. IV-A. Typical number of tracked point features for the model-free tracker was 25 for Seq. 1 and 8 for Seq. 2.

Figures 6–9 show the error magnitudes for the two sequences for both system models. For Sequence 1, it can be seen that the camera object model is less prone to sudden changes in rotation error. This is because the model restricts the rotation of the object to co-occur with suitable translations. Both models had similar average errors, the moving object model having slightly lower error of 1.7 cm in translation and 3.8 degrees in rotation compared to 1.9 cm and 3.9 degrees with the moving camera model. For Sequence 2, the errors were 6.5 cm, 12.2 degrees (moving object), and 6.5 cm, 12.1 degrees (moving camera). Here, the rotation error was remarkably higher. This is because in the sequence one of the visible edges can not be detected because the lighting angle caused the pixels on both sides of the edge to have same color. Thus, the edge is mismatched to a neighboring edge which causes the pose to rotate somewhat. This is a problem that cannot be solved with the model-free features, and requires further study.

The use of integration was also inspected by ignoring one of the cues. These results can be seen in Table I for Sequence 1. It can be seen that the use of only model-free features results in a bad accuracy. This is because the pose drifts as the features only describe the relative pose changes without respect to absolute error. This drifting is demonstrated in Fig. 10. Neither of the sequences had a strong background edge coinciding with one of the object edges, which would have made the model tracker fail as in Fig. 1. Thus, the integration did not have a dramatic effect, but still improved the accuracy.

TABLE I
Integration results.

| | model | model-free | both |
|---|---|---|---|
| moving object | 0.018 / 4.1 | 0.25 / 24.8 | 0.017 / 3.8 |
| moving camera | 0.052 / 4.0 | 0.075 / 27.7 | 0.019 / 3.9 |

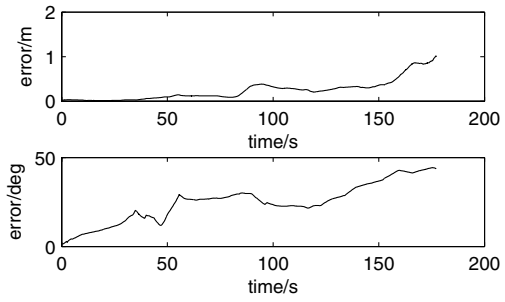


Fig. 10. Drift in model-free tracking.

The effect of the sampling rate was inspected by lowering the sample rate to 1/2, 1/3, et cetera. The errors for

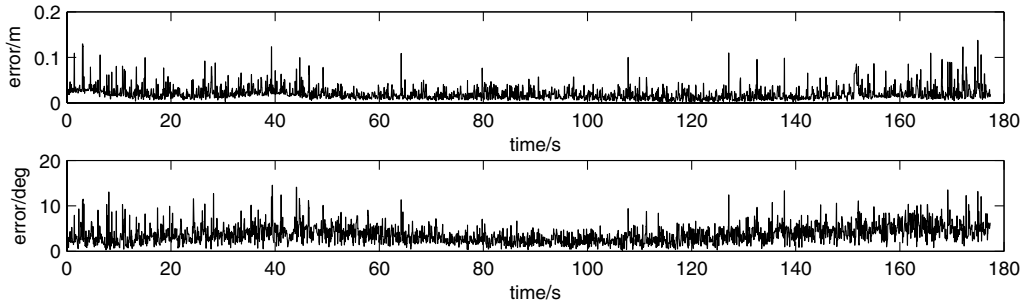


Fig. 5. Initialization accuracy.

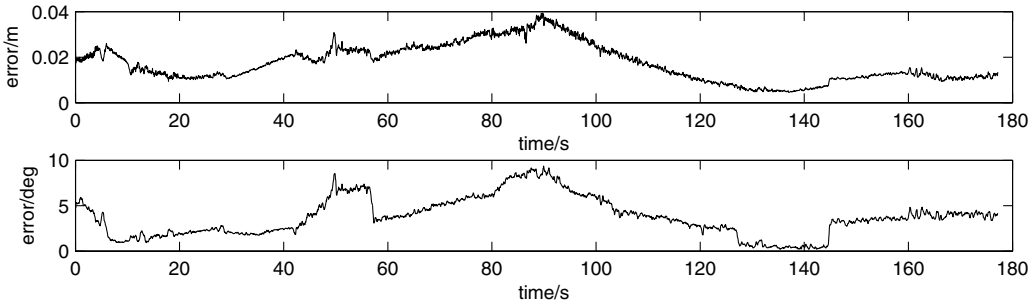


Fig. 6. Moving object model, Sequence 1.

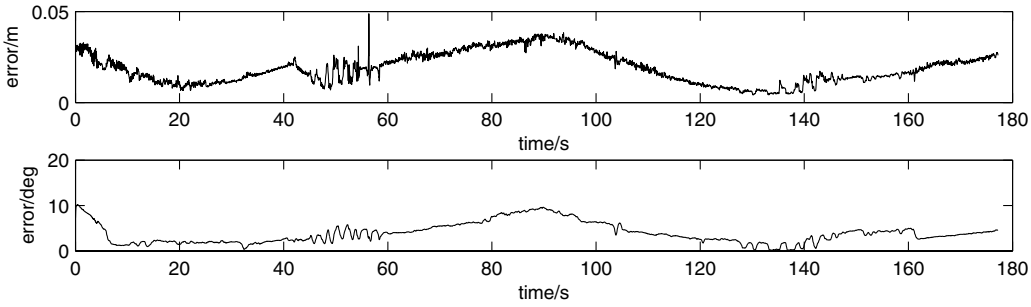


Fig. 7. Moving camera model, Sequence 1.

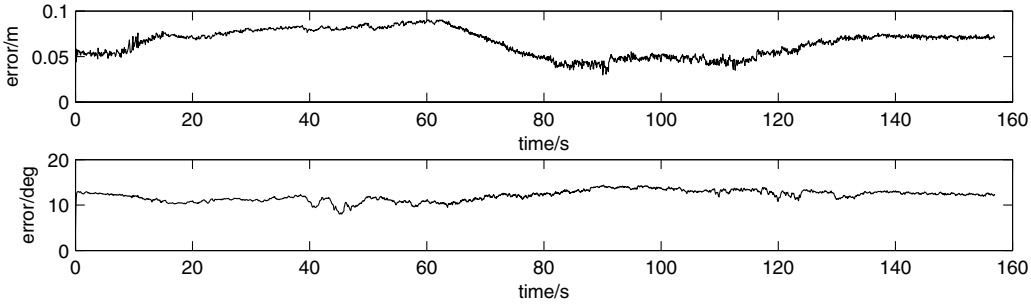


Fig. 8. Moving object model, Sequence 2.

different sampling rates are shown in Table II. The sudden increase in the errors at 1/3 sampling rate happens because at that rate the object motion at one point of the sequence is fast enough so that one of the tracked edges is confused with another. After that point, the tracker has a constant error offset, which can be seen in Fig. 11 which shows the evolution of the error for 1/5 sampling rate.

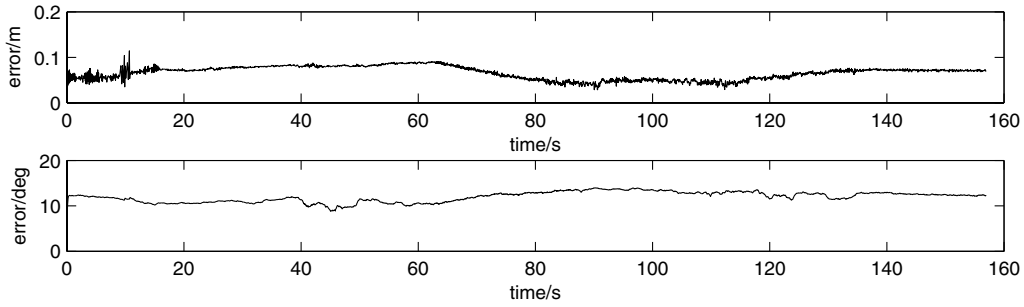


Fig. 9. Moving camera model, Sequence 2.

TABLE II
ERRORS WITH LOWER SAMPLING RATES.

| | moving object | moving camera |
|---|---|---|
| 1/1 | 0.017 / 3.8 | 0.019 / 3.9 |
| 1/2 | 0.019 / 4.4 | 0.020 / 4.8 |
| 1/3 | 0.038 / 13.8 | 0.037 / 13.4 |
| 1/4 | 0.038 / 13.8 | 0.038 / 13.6 |
| 1/5 | 0.039 / 13.9 | 0.041 / 17.6 |

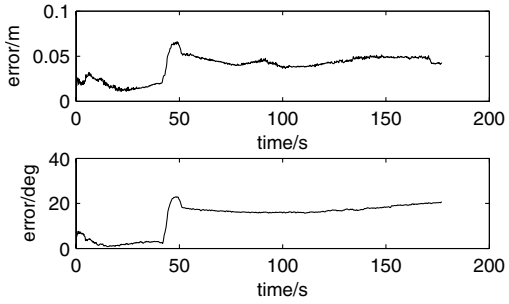


Fig. 11. Error at 1/5 sampling rate.

## VI. DISCUSSION

In this paper, we have presented a method to integrate information from model-based and model-free cues for 3D object tracking. The method is based on an Iterated Extended Kalman filter, and two different system models have been presented for the motion. System run-time may differ during different iterations, which is taken into account in the changes of the state covariance. Two types of image features are used to compensate for each others weaknesses. We believe that the integration of model-free cues can benefit a wide variety of tracking approaches.

An approach parallel to the use of pose tracking is the direct use of image measurements in visual control, for example, in image based visual servoing. However, we believe that the use of multiple visual cues is very important to avoid the weaknesses of each individual cue. In that context it is important to note that the work presented also applies to visual control strategies other than the basic position based servoing, e.g., [18]. We have not considered the issue of outlier measurements, which has received attention in tracking studies, but which clearly needs to be further examined in the context of model-free cues. In addition, we plan to continue the study of system models to find out if constant velocity models have advantage over the presented models.

## REFERENCES

[1] C. Harris, "Tracking with rigid models," in *Active Vision*, A. Blake and A. Yuille, Eds. MIT Press, 1992, ch. 4, pp. 59–73.
[2] P. Wunsch and G. Hirzinger, "Real-time visual tracking of 3-d objects with dynamic handling of occlusion," in *IEEE Int. Conf. on Robotics and Automation, ICRA'97*, Albuquerque, New Mexico, USA, Apr. 1997, pp. 2868–2873.
[3] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Trans. PAMI*, vol. 24, no. 7, pp. 932–946, 2002.
[4] S. Ekvall and D. Kragic, "Interactive grasp learning based on human demonstration," in *IEEE International Conference on Robotics and Automation, ICRA'04*, 2004.
[5] C. Rasmussen and G. Hager, "Probabilistic data association methods for tracking complex visual objects," *IEEE Trans. PAMI*, vol. 23, no. 6, pp. 560–576, 2001.
[6] D. Kragic and H. I. Christensen, "Cue integration for visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 1, pp. 18–27, Feb. 2001.
[7] G. Taylor and L. Kleeman, "Fusion of multimodal visual cues for model-based object tracking," in *Australiasian Conf. on Robotics and Automation*, Brisbane, Australia, 2003.
[8] F. Jurie and M. Dhome, "Real time tracking of 3D objects: an efficient and robust approach," *Pattern Recognition*, vol. 35, pp. 317–328, 2002.
[9] E. D. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine Vision and Applications*, vol. 1, pp. 223–240, 1988.
[10] D. G. Lowe, "Robust model-based motion tracking through the integration of search and estimation," *Int. J. of Comp. Vis.*, vol. 8, no. 2, pp. 113–122, 1992.
[11] M. Vincze, M. Ayromlou, and W. Kubinger, "An integrating framework for robust real-time 3D object tracking," in *Int. Conf. on Comp. Vis. Syst., ICVS'99*, 1999, pp. 135–150.
[12] D. Koller, K. Daniilidis, and H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *International Journal of Computer Vision*, 1993.
[13] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comp. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[14] D. Kragic, "Visual Servoing for Manipulation: Robustness and Integration Issues," Ph.D. dissertation, CVAP, Royal Institute of Technology, Stockholm, Sweden, 2001.
[15] C. J. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vision Conference*, Manchester, UK, 1988, pp. 147–151.
[16] G. Welch and G. Bishop, "SCAAT: Incremental tracking with incomplete information," in *Proc. Computer graphics and interactive techniques*, Los Angeles, CA, USA, August 3–8 1997, pp. 333–344.
[17] R. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 3, pp. 345–358, 1989.
[18] V. Kyrki, D. Kragic, and H. I. Christensen, "New shortest-path approaches to visual servoing," in *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, Sendai, Japan, Sept. 2004, to be published.