Real-time 3D Model-based Tracking Using Edge and Keypoint Features for Robotic Manipulation

Changhyun Choi and Henrik I. Christensen  
Robotics & Intelligent Machines, College of Computing  
Georgia Institute of Technology  
Atlanta, GA 30332, USA  
{cchoi,hic}@cc.gatech.edu

Abstract—We propose a combined approach for 3D real-time object recognition and tracking, which is directly applicable to robotic manipulation. We use keypoints features for the initial pose estimation. This pose estimate serves as an initial estimate for edge-based tracking. The combination of these two complementary methods provides an efficient and robust tracking solution. The main contributions of this paper includes: 1) While most of the RAPiD style tracking methods have used simplified CAD models or at least manually well designed models, our system can handle any form of polygon mesh model. To achieve the generality of object shapes, salient edges are automatically identified during an offline stage. Dull edges usually invisible in images are maintained as well for the cases when they constitute the object boundaries. 2) Our system provides a fully automatic recognition and tracking solution, unlike most of the previous edge-based tracking that require a manual pose initialization scheme. Since the edge-based tracking sometimes drift because of edge ambiguity, the proposed system monitors the tracking results and occasionally re-initialize when the tracking results are inconsistent. Experimental results demonstrate our system’s efficiency as well as robustness.

I. INTRODUCTION

As robots moves from industrial to daily environments, the most important problem robots face is to recognize objects and estimate 6-DOF pose parameters in less constrained environments. For the last decade, computer vision, robotics, and augmented reality have all addressed this as a model-based tracking issue. Most of the work has been based on 3D CAD models or keypoint metric models. The former models correspond to edges in an image, which can be efficiently computed, while the latter models match with keypoints in an image which are suitable for robust wide baseline matching. A strategy for using keypoint for pose initialization and differential methods for pose tracking is presented.

II. RELATED WORK

For the 6-DOF pose tracking, robotics and augmented reality areas have employed a number of different approaches. One of the easiest way is through use of fiducial markers. Artificial markers are attached to the object or environment as camera targets. Although the method provides an easy and robust solution for real-time pose estimation, attaching markers has been regarded as a major limitation. Hence, researchers have focused on tracking using natural features. For several decades methods, which employ natural features, have been proposed: edge-based, optical flow-based, template-based, and keypoint-based. Each method has its own pros and cons, but surveying every methods in this paper is out of scope. For an in-depth study of the different methods, we refer the interested reader to the survey [1].

Among the various methods, we focus on two methods: edge-based and keypoint-based. The edge features are easy to compute and computationally cheap. Since the edge is usually computed by image gradients, it is moderately invariant to illumination and viewpoint. The keypoint features are also capable of being invariant to illumination, orientation, scale, and partially viewpoint. But the keypoints requires relatively computationally expensive descriptors which maintain local texture or orientation information around stable points to be distinctive.

In edge-based methods, a 3D CAD model is usually employed to estimate the full pose using a monocular camera. Harris [2] established RAPiD (Real-time Attitude and Position Determination) which was one of the first marker-less 3D model-based real-time tracking system. It tracks an object by comparing projected CAD model edges to edges detected in a gray-scale image. To project the model close...
to the real object, the system use the previous pose estimate as a priori. Since it use an 1-D search along the normal direction of sample points for the closest edge locations, it rapidly calculate errors which must be minimized to solve for the 6-DOF motion parameters. The motion parameters are subsequently estimated between frames. Drummond and Cipolla [3] solved a similar problem, but enhanced robustness by using the iterative re-weighted least squares with a M-estimator. To perform hidden line removal, they used a BSP (Binary Space Partition) tree. Marchand and Chaumette [4] proposed an augmented reality framework, which relies on points and lines, and that has been applied to the visual servoing [5]. Comport et al. [6] compared and evaluated the two different systems, but they concluded both are fundamentally equivalent.

In *keypoint-based* methods, a sparse 3D metric model is used. Like CAD models, the keypoint models are built offline. With a set of images in each has a view of an object from a slightly different viewpoint, the non-linear optimization algorithm, such as Levenberg-Marquardt, return a refined 3D model of keypoints. Since this model maintains 3D coordinates of each keypoint, the pose estimation is easily performed by using the correspondence between the 3D points of the model and the 2D keypoints in an input image. Using this model, Gordon and Lowe [7] proposed an augmented reality system that calculates pose with scale invariant features [8]. Collet et al. [9] applied a similar method to robot manipulation where they combined RANSAC [10] with a clustering algorithm to locate multiple instances. Vacchetti et al. [11] used standard corner features to match the current image and the reference frames, so called keyframes. Unlike the efforts using non-linear optimization, they obtained 3D coordinates of 2D corner points by back-projecting them onto the object CAD model.

Since the edge and the keypoint methods are complementary to each other, several have reported combined approaches [12], [13]. Vacchetti et al. [14] incorporated the edge-based method with their corner point-based method to make the system more robust and jitter free. As part of the edge-based tracking, they used multiple hypotheses to handle erroneous edge correspondence, but it is equivalent to the nearest hypothesis of RAPiD-like approaches. Rosten and Drummond [15] similarly combined corner points with lines, but they only used corner points to estimate motion parameters between frames.

We also adopt a combined approach in which *keypoint-based* matching and *edge-based* tracking are employed. As depicted in Fig. 1, our system is composed of a Global Pose Estimation (GPE) and a Local Pose Estimation (LPE). Unlike [14] and [15] which use keypoints to estimate motion between frames, we only use the keypoints for estimating the initial pose in GPE. After estimating the initial pose an edge-based tracking scheme is utilized in the LPE.

In the remainder of the paper, we first explain the GPE in Section III. Section IV describes the LPE including the salient edge selection from polygon mesh models and the edge-based tracking formulation. Quantitative and qualitative results using the system are presented in Section V.

III. GLOBAL POSE ESTIMATION USING KEYPONITS

In this section, we present the Global Pose Estimation (GPE) in which we use SURF keypoints [16] to match the current image with keyframes. The model keyframe is a set of images that contains a target object. The keyframes are saved offline. To estimate pose, the 3D coordinate of each keypoint is computed by back-projecting to the CAD model.

A. Keypframe Model Acquisition

To estimate an initial pose, our system requires keyframes, which are reference images. Since the keyframes will be compared with the input image, the keyframes should contain appearance of the object similar to the one in the input image. But it is practically impossible to maintain every image to cover all possible appearances of the object due to variability across illumination, scale, orientation and viewpoint. In a real application, a smaller number of keyframes is preferred. Ideally there would only be one keyframe per aspect for the object. For the maximum coverage of a keyframe, a keypoint descriptor that describes local appearance around corner-like points is used. If the local descriptor is discriminative then matching keypoints between two images can be performed despite variations in orientation, scale, and illumination. However, the local appearance is only semi-invariant to viewpoint change. For robust pose initialization we are required to maintain multiple keyframes to cover multiple view aspects.

Capturing keyframes is performed offline. Since keyframes will be used for pose estimation in which there is a need for generation of 2D-3D correspondences, we need to know the 3D coordinates of each keypoint. To calculate 3D coordinates, we use 3D CAD models with the current pose estimate. In this phase, the current pose is estimated by the LPE as will be explained in Section IV. With a CAD model and the current pose, we can compute the 3D coordinates of each keypoint by back-projecting the 2D keypoint to
the corresponding facet of the CAD model. For fast facet identification, we use ‘Facet-ID’ trick which encodes i-th facet of the target object’s model in an unique color in order to identify the membership of each 2D keypoint by looking up the image buffer that OpenGL renders [11]. The 3D coordinates of the keypoints are then saved into a file for later use in keypoint matching.

B. Matching keypoints

After obtaining keyframes offline, keypoint matching is performed between an input frame and keyframes. A simple strategy for the matching might use naïve exhaustive search. However, such a search has \(O(n^2)\) complexity. Using an approximated method the complexity can be reduced. As an approximated search, we use the Best-Bin-First (BBF) algorithm [17] which can be performed in \(O(n \log n)\). While [18] and [8] used a fixed number of nearest-neighbors, we set the number of nearest-neighbors as the number of keyframe + 1. We use the ratio test described by [8], and the ratio threshold we used was 0.7. Once the putative correspondences has been determined, they are further refined using RANSAC [10]. In each RANSAC iteration, we estimate a homography matrix and eliminate outliers from the homography matrix. Since most of objects which exist in our daily environment are manufactured, their CAD models might be available, and such models provide helpful information for robotic manipulation. Although there are various formats in CAD models, most of them can be represented in a polygon mesh. A polygon mesh is usually composed of vertices, edges, faces, polygons and surfaces. In the LPE, we use edge features in images coming from a monocular camera to estimate the pose difference between two consecutive frames. So we should determine which edges in the model of a targeted object would be visible in images. Here we make an assumption that sharp edges are more likely to be salient. To identify sharp edges, we use the face normal vectors available in the model.

IV. Local Pose Estimation Using Edges

In this section, we explain the Local Pose Estimation (LPE) in which edges are utilized for object tracking.

A. Automatic Salient Model Edges Selection

Since most of objects which exist in our daily environment are manufactured, their CAD models might be available, and such models provide helpful information for robotic manipulation. Although there are various formats in CAD models, most of them can be represented in a polygon mesh. A polygon mesh is usually composed of vertices, edges, faces, polygons and surfaces. In the LPE, we use edge features in images coming from a monocular camera to estimate the pose difference between two consecutive frames. So we should determine which edges in the model of a targeted object would be visible in images. Here we make an assumption that sharp edges are more likely to be salient. To identify sharp edges, we use the face normal vectors from the model. As illustrated in Fig. 5, if the face normal vectors of two adjacent faces are close to perpendicular, the edge shared by the two faces is regarded a sharp edge. If two face normal vectors are close to parallel, the edge is regarded a dull edge. For the decision, we use a simple thresholding scheme with the value of the inner product of two normal vectors. More formally, we can define an indicator function with respect to
the edges in the model by:

\[ I(\text{edge}_i) = \begin{cases} 1 & \text{if } |\mathbf{n}_1^i \cdot \mathbf{n}_2^i| \leq \tau_s \\ 0 & \text{otherwise} \end{cases} \]

where \( \mathbf{n}_1^i \) and \( \mathbf{n}_2^i \) are the face normal unit vectors of the two adjacent faces which share the i-th edge, \( \text{edge}_i \). We found the threshold \( \tau_s = 0.3 \) is a reasonable value. This salient edge selection is performed fully automatically offline. In general, the salient edges are only considered in edge-based tracking, but when the dull edges constitute the object’s boundary they are also considered. Testing boundary of the dull edges are performed at run-time using back-face culling.

**B. Mathematical and Camera Projection Model**

Since our approach is based on the formulation from Drummond and Cipolla [3], we adopt the Lie Algebra formulation. In the LPE, our goal is to estimate the posterior pose \( E_{t+1} \) from the prior pose \( E_t \) given the inter-frame motion \( M \):

\[ E_{t+1} = E_t M \]

where \( E_{t+1}, E_t, \) and \( M \) are 6-dimensional Lie Group of rigid body motion in \( SE(3) \). At time \( t + 1 \), we know the prior pose \( E_t \) from the GPE or the previous LPE. Hence we are interested in determining the motion \( M \) to estimate the posterior pose. \( M \) can be represented in the exponential map of generators \( G_i \) as follows:

\[ M = \exp(\mu) = e^{\sum_{i=1}^{6} \mu_i G_i} \quad (1) \]

where \( \mu \in \mathbb{R}^6 \) is the motion velocities corresponding to the 6-DOF instantaneous displacement and the \( G_i \) are the group generators matrices:

\[ G_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, G_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, G_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \]

\[ G_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, G_5 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, G_6 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \]

As a camera model, we use the standard pin-hole model given by:

\[ p = \text{Proj}(P^M; E, K) = K \begin{pmatrix} x^c \\ y^c \\ 1 \end{pmatrix} \]

where \( p = (u \ v)^T \) is 2D image coordinates corresponding to the 3D model coordinates \( P^M = (x^M \ y^M \ z^M)^T \) and the matrix \( K \) represent the camera’s intrinsic parameters:

\[ K = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \]

where \( f_u \) and \( f_v \) are the focal length in pixel dimensions, and \( u_0 \) and \( v_0 \) represent the position of the principal point. The 3D coordinates in camera coordinates \( P^C = (x^C \ y^C \ z^C)^T \) can be calculated by:

\[ P^C = E P^M \]

where \( E \) is the extrinsic matrix or camera’s pose. For simplicity, we ignore the radial distortion as image rectification is performed during the image acquisition phase.

**C. Model Rendering and Error Calculation**

To estimate the motion \( M \), we need to measure errors between the prior pose and the current pose. As a first step to calculate errors, we project the CAD model to the image plane using the prior pose \( E_t \). Instead of considering the edge itself, we sample points along the projected edges. Since some of sampled points are occluded by the object itself, a visibility test is performed. While [3] used a BSP tree for hidden line removal, OpenGL occlusion query is an easy and efficient alternative. Each visible point is then matched to the edges in the input image. The edge image is obtained by using a Canny Edge Detector [19]. We find the nearest edge by using a 1-D search along the direction perpendicular to the projected edge. The error vector \( e \) is obtained by stacking all of the errors of each sample point as follows:

\[ e = (e_1 \ e_2 \ldots \ e_N)^T \]
where \( e_i \) is the Euclidean distance from i-th sample point to the nearest edge and \( N \) is the number of valid sample points (i.e., sample points correspond to the nearest edge). Fig. 7 illustrates the error calculation, and \( e_i \) is the length of the i-th red arrow.

D. Update Pose with IRLS

After calculating the error vector \( e \), the problem is reduced to:

\[
\hat{\mu} = \arg \min_{\mu} \sum_{i=1}^{N} ||e_i||^2
\]

\[
= \arg \min_{\mu} \sum_{i=1}^{N} ||p_i - \text{Proj}(\mathbf{P}^M_i; E_t \exp(\mu), K)||^2
\]

where \( p_i \) is the 2D image coordinates of the nearest edge which is corresponding to the projected 2D point of the i-th 3D model coordinates \( \mathbf{P}^M_i = (x_i^M, y_i^M, z_i^M)^T \) and \( N \) is the number of valid sample points.

To calculate \( \mu \) which minimizes the error \( e \), a Jacobian matrix \( J \in \mathbb{R}^{N \times 6} \) can be obtained by computing partial derivatives at the current pose:

\[
J_{ij} = \frac{\partial e_i}{\partial \mu_j}
\]

\[
= n_i^T \frac{\partial}{\partial \mu_j} \left( u_i \right)
\]

\[
= n_i^T \frac{\partial}{\partial \mu_j} \left( \text{Proj}(\mathbf{P}^M_i; E_t \exp(\mu), K) \right)
\]

where \( n_i \) is the unit normal vector of the i-th sample point.

We can split \( \text{Proj()} \) in Eq. 2 into two parts as follows:

\[
\begin{pmatrix}
    u_i \\
    v_i
\end{pmatrix}
= \begin{pmatrix}
    f_u & 0 & u_0 \\
    0 & f_v & v_0
\end{pmatrix}
\begin{pmatrix}
    \tilde{u}_i \\
    \tilde{v}_i
\end{pmatrix}
\]

\[
\begin{pmatrix}
    \tilde{u}_i \\
    \tilde{v}_i
\end{pmatrix}
= \begin{pmatrix}
    x^C_i \\
    v^C_i
\end{pmatrix}
\]

Their corresponding Jacobian matrices can be obtained:

\[
J_K = \begin{pmatrix}
    \frac{\partial u_i}{\partial x^C_i} & \frac{\partial u_i}{\partial y^C_i} & 0 & 0 & \frac{\partial u_i}{\partial z^C_i} & 0 \\
    \frac{\partial u_i}{\partial z^C_i} & \frac{\partial v_i}{\partial x^C_i} & 0 & 0 & \frac{\partial v_i}{\partial y^C_i} & 0
\end{pmatrix}
\]

\[
J_P = \begin{pmatrix}
    \frac{\partial \tilde{u}_i}{\partial x^C_i} & \frac{\partial \tilde{u}_i}{\partial y^C_i} & \frac{\partial \tilde{u}_i}{\partial z^C_i} & \frac{\partial \tilde{v}_i}{\partial x^C_i} & \frac{\partial \tilde{v}_i}{\partial y^C_i} & \frac{\partial \tilde{v}_i}{\partial z^C_i}
\end{pmatrix}
= \begin{pmatrix}
    \frac{1}{z^C_i} & 0 & \frac{-x^C_i}{(z^C_i)^2} & 0 & \frac{1}{z^C_i} & \frac{-y^C_i}{(z^C_i)^2}
\end{pmatrix}
\]

Since \( \frac{\partial}{\partial \mu_j}(\exp(\mu)) = G_j \) at \( \mu = 0 \) by Eq. 1, we can get:

\[
\frac{\partial \mathbf{P}^C_i}{\partial \mu_j} = \frac{\partial}{\partial \mu_j}(E_t \exp(\mu) \mathbf{P}^M_i)
= E_t G_j \mathbf{P}^M_i
\]

Therefore the i-th row and j-th column element of the Jacobian matrix \( J \) is:

\[
J_{ij} = \frac{\partial e_i}{\partial \mu_j} = n_i^T J_K \begin{pmatrix}
    J_P & 0
\end{pmatrix} E_t G_j \mathbf{P}^M_i
\]

We can solve the following equation to calculate the motion velocities:

\[
J \mu = e
\]
Rather than using the usual pseudo-inverse of \( J \), we solve the above equation with Iterative Re-weight Least Square (IRLS) and M-estimator:

\[
\hat{\mu} = (J^T W J)^{-1} J^T W e
\]

where \( W \) is a diagonal matrix determined by a M-estimator. The \( i \)-th diagonal element in \( W \) is \( w_i = \frac{1}{c + e_i} \) where \( c \) is a constant.

V. EXPERIMENTAL RESULTS

![Figure 8: 6-DOF pose plots of the book object in the general tracking test. While our approach (GPE+LPE) has convergence to ground truth, the GPE only mode suffers from jitter and occasionally fails to estimate pose.](image)

![Figure 9: Normalized residue plots of the book object in the general tracking test. The jitter and tracking failures result in a high residual.](image)

In this section, we validate our visual recognition and tracking algorithm with several experiments. To show the generality of our system, we performed experiments with 4 objects: teabox, book, cup and car door. Note that these objects have different complexity and characteristics. The first three objects are especially interesting in for service robotics while the last object is of interest for assembly robots.

Our system is composed of a standard desktop computer and a Point Grey Research’s Flea 1394 camera (640 × 480 resolution). The CAD models of teabox, book and cup were generated by using Blender™ which is an open source 3D modeling tool. The car door model was provided by an automobile company. We converted all of the models to the OBJ format\(^1\) to be used in our C++ implementation.

For the GPE, we prepared keyframe images. As a smaller number of keyframes is desirable, we captured only five keyframes per object. Each keyframe has different appearances of object as shown in Fig. 2.

A. General Tracking Test

The tracking results for the four objects are shown in Fig. 6. The images in left-most column show estimated pose from the GPE and the last of them depicts the pose estimated by the LPE. Note that although the pose estimated by the GPE is not perfect, the subsequent LPE corrects the error and the pose estimates converge to the real pose. For quantitative evaluation, we employed AR markers to gather ground truth pose data. As shown in Fig. 10, we manually measured the transformation \( M_T_O \) which is the description of the object frame \{O\} relative to the marker frame \{M\}. So the ground

\(^1\)OBJ format is developed by Wavefront Technologies and has been widely accepted for 3D graphics. That format can be easily handled by using the GLUT library.
is required to generate a robust tracking system. Here we cause of edge’s ambiguity. So monitoring and re-initializing
keypoint-based texture. This implies that when an object is textureless, the
of their appearances which stem from the lack of surface
and the car door objects. The errors are due to limitations
in terms of accuracy. Note the significant errors for the cup
book object, our approach outperforms the GPE only mode
our approach (i.e GPE+LPE). Except the roll angle of the
the GPE only mode and the lower ones are the results of
Table I. For each object, the upper ones are the results of
of the tracking test for the four objects are presented in
shortcomings result in the high residues in both translation
and rotation (Fig. 9). The RMS (Root Mean Square) errors
of the tracking test for the four objects are presented in
Fig. 11: 6-DOF pose plots of the book object in the re-initialization test. The shaded spans mean that the object and the marker are
invisible because of fast camera movement or occlusion. Our approach (GPE+LPE) takes more frame to re-initialize than the AR marker,
but it maintains track of the object.

![Fig. 11: 6-DOF pose plots of the book object in the re-initialization test. The shaded spans mean that the object and the marker are invisible because of fast camera movement or occlusion. Our approach (GPE+LPE) takes more frame to re-initialize than the AR marker, but it maintains track of the object.](image)

![Fig. 12: Normalized residual plots of the book object in the re-initialization test. The three peaks are due to a time difference between the AR marker and our approach.](image)

truth pose $C^O_T$ can be obtained as follows:

$$C^O_T = C^M_T M^O_T$$

where $C^M_T$ is the pose estimated by AR markers. The
estimated pose by our system $C^O_T$ is compared with the
ground truth $C^O_T$ as shown in Fig. 8 and Fig. 11. The
plot shows the estimated poses of the book object from the
GPE only mode and the GPE+LPE mode. Since the GPE
relies on the keypoint matching, the quality of the keypoint
correspondences directly affect the pose results. Hence the
GPE only mode produces significant jitter. Sometimes it fails
to estimate pose when the number of correspondences is
insufficient (in our experiments, we only considered 12 or
more correspondences after the RANSAC iterations). These
shortcomings result in the high residues in both translation
and rotation (Fig. 9). The RMS (Root Mean Square) errors
of the tracking test for the four objects are presented in
Table I. For each object, the upper ones are the results of
the GPE only mode and the lower ones are the results of
our approach (i.e GPE+LPE). Except the roll angle of the
book object, our approach outperforms the GPE only mode
in terms of accuracy. Note the significant errors for the cup
and the car door objects. The errors are due to limitations
of their appearances which stem from the lack of surface
texture. This implies that when an object is textureless, the
keypoint-based method might encounter challenges.

### B. Re-initialization Test

During the LPE, it might converge to a local minima be-
cause of edge’s ambiguity. So monitoring and re-initializing
is required to generate a robust tracking system. Here we
use a simple heuristic based on the difference in position of
the object between frames and the number of valid sample
points. When the tracked object drifts, it frequently moves
rapidly while the general motions of the object or the camera
does not because the frequency in the image acquisition is
high$^2$. The number of valid sample points also gives a clue
to the quality of the pose. Since a good status in LPE implies
that most of the sampled points are matched to image edges,
we can reason that lots of invalid sample points indicate a
risks of tracking failure. In this experiment, we use a criteria
when at least one of the xyz coordinates of the object moves
more than 10 cm between frames or the number of valid
sample points is lower than the half of the total visible sample
points, the algorithm switch from the LPE to the GPE and re-

![Fig. 12: Normalized residual plots of the book object in the re-initialization test. The three peaks are due to a time difference between the AR marker and our approach.](image)

\[ \text{TABLE I: RMS ERRORS.} \]

<table>
<thead>
<tr>
<th></th>
<th>RMS Errors (in meter and degree)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Teabox</td>
<td>0.0076</td>
</tr>
<tr>
<td></td>
<td>0.0033</td>
</tr>
<tr>
<td>Book</td>
<td>0.0043</td>
</tr>
<tr>
<td></td>
<td>0.0026</td>
</tr>
<tr>
<td>Cup</td>
<td>0.0603</td>
</tr>
<tr>
<td></td>
<td>0.0083</td>
</tr>
<tr>
<td>Car door</td>
<td>0.0502</td>
</tr>
<tr>
<td></td>
<td>0.0211</td>
</tr>
</tbody>
</table>

$^2$In our implementation, the frame rate is close to 30Hz which means the period is about 33 msec.
initializes. For the test, we intentionally moved the camera to a new scene that does not have the tracked object, shook the camera rapidly to test on blurred images, and occluded with a paper. Fig. 11 shows the full pose of the book object in the re-initialization test. There are three trials of re-initialization and the invisible spans are shaded in each plot. Since corner features are more easily identified than SURF keypoints in blurred images, the AR marker (i.e. Ground Truth) returns slightly faster than the GPE. This time difference leads to peaks in residue plots (Fig. 12).

C. Computation Times

Fig. 13: Computation times of the two experimentations of the book object.

For robotic manipulation, higher frame rates are an important requirement. Fig. 13 shows the computation times plot of the two experiments. Since the GPE is executed during re-initializations, our approach (i.e GPE+LPE) takes nearly the same times as the GPE only mode. The reason why the GPE takes less time during re-initialization spans is that the insufficient number of matching skips RANSAC and the pose estimation. The average computation times of the aforementioned experiment is shown in Table II.

<table>
<thead>
<tr>
<th>Object</th>
<th>GPE (msec)</th>
<th>GPE+LPE (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teabox</td>
<td>83.3984</td>
<td>32.6095</td>
</tr>
<tr>
<td>Book</td>
<td>85.3586</td>
<td>32.6478</td>
</tr>
<tr>
<td>Book (re-init)</td>
<td>84.5773</td>
<td>39.9883</td>
</tr>
<tr>
<td>Cup</td>
<td>83.6209</td>
<td>43.8241</td>
</tr>
<tr>
<td>Car door</td>
<td>94.3990</td>
<td>42.4021</td>
</tr>
</tbody>
</table>

VI. CONCLUSIONS

We presented a hybrid approach for 3D model-based object tracking. The keypoint-based global pose estimation enabled the proposed system to initialize the tracking system. The edge-based local pose estimation achieves efficient pose tracking. By monitoring the pose results, our system can automatically re-initialize when the tracked results are inconsistent. Since our approach can handle general polygon mesh models, we expect the proposed system can be widely employed for robot manipulation of complex objects.

VII. ACKNOWLEDGMENTS

This work was fully funded and developed under a Collaborative Research Project between Georgia Tech and the General Motors R&D, Manufacturing Systems Research Laboratory on Interaction and Learning for Autonomous Assembly Robots. General Motors support is gratefully acknowledged.

REFERENCES