A model-based tracking framework for textureless 3D rigid curved objects

Marina Atsumi Oikawa*, Takafumi Taketomi*, Goshiro Yamamoto*, Makoto Fujisawa†, Toshiyuki Amano‡, Jun Miyazaki* and Hirokazu Kato *
*Nara Institute of Science and Technology, Japan
Email: {marina-o, takafumi-t, goshiro, miyazaki, kato}@is.naist.jp
†University of Tsukuba, Japan
Email: fujis@slis.tsukuba.ac.jp
‡Yamagata University, Japan
E-mail: amano@yz.yamagata-u.ac.jp

Abstract—**This paper addresses the problem of tracking textureless rigid curved objects. A common approach uses polygonal meshes to represent curved objects inside an edge-based tracking system. However, in order to accurately recover their shape, high quality meshes are required, creating a trade-off between computational efficiency and tracking accuracy. To solve this issue, we suggest the use of quadrics calculated for each patch in the mesh to give local approximations of the object contour. This representation reduces considerably the level of detail of the polygonal mesh while maintaining tracking accuracy. The novelty of our research lies in using curves to represent the quadrics' projection in the current viewpoint for distance evaluation instead of comparing directly the edges from the mesh and detected edges in the video image. In our tracking framework, we also include a method to calculate the measurable Degrees of Freedom (DoF) of the target object. This is used to recover the pose parameters when the object has less than 6DoF. Experimental results compare our approach to the traditional method of using sparse and dense meshes. Finally, we present a potential Augmented Reality application of the proposed method.**

I. Introduction

Tracking the 3D pose of a known object is a common task in computer vision and many approaches aiming to achieve real-time tracking have been developed to attend different applications and scenarios. Considering tracking for rigid objects, there is a need to continuously recover 6 DoF parameters representing the object position and orientation relative to the camera while it moves around the scene [1].

The method described in this paper uses a model-based approach that considers the object edges during tracking, similar to [2], [3]. A CAD model of the target object is used for matching with the edge information found in the video image. This matching is done by looking for strong gradients in the image using an initial estimation of the pose and performing edge normal search of projected edges in the image. The final pose is obtained after an optimization process.

However, unlike the aforementioned approaches, which are applied mainly to polyhedral objects with flat faces, our method targets rigid curved objects. In this case, the tracking becomes more challenging because curved objects do not present static edges; instead, they change according to the viewpoint, representing the *apparent contour* of the object.

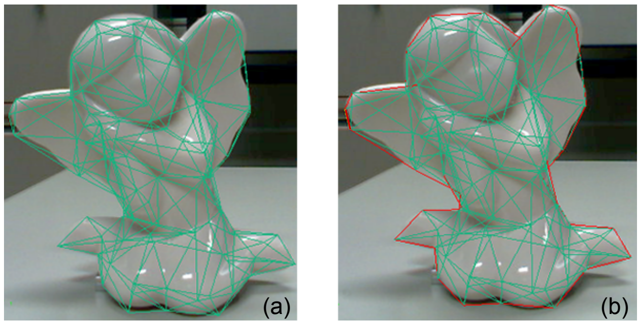


Fig. 1. In (a), a sparse mesh is rendered on the target object resulting in (b) a coarse representation of the object contour (red lines).

The apparent contour is a curve formed by the projection of the *contour generator*, that is, parts of the surface that are tangent to the viewing ray [4]. It is the main feature found in curved surfaces and useful when the object does not have any texture information available. However, its use with standard edge based tracking methods is not trivial as shown by previous works in section II.

Furthermore, when dealing with curved objects, high quality meshes are required to accurately represent the object's shape. This creates a trade-off between the computational efficiency and tracking accuracy: reducing the number of patches in the mesh to improve the system efficiency affects the object contour representation.

In Figure 1(a), a sparse polygonal mesh representing the angel figurine is rendered on top of the real object. The object contour has a coarse representation as highlighted in Figure 1(b), which affects accuracy since this increases the error between projected and detected edge points.

A. Our approach

To use sparse polygonal meshes for edge-based tracking of curved objects, we have developed a model representation named *quadrics patch representation*, where a general quadric equation is calculated for each patch in the mesh and used to represent the local shape of the object.

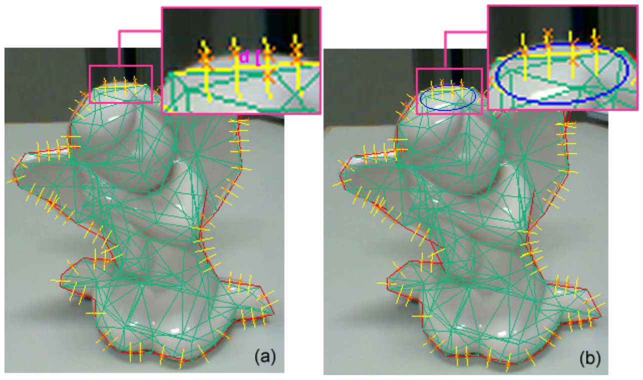


Fig. 2. (a) When the edges from the mesh are used alone with sparse meshes, a large distance *d* between the sample points on the model and the detected edge points is evaluated. (b) However, by approximating each patch by a quadric surface, its projection represented by conic curves (blue line) approximates better the object outline - the conic curve shape is very close to the object outline in the highlighted part.

Different from standard edge-based tracking systems, our approach does not evaluate the distance between the points assigned on the mesh edges to the detected points in the video image; instead, the distance is evaluated by using the curves representing the quadrics projection of the patches located on the object contour. For instance, in Figure 2(b), instead of using the model edge (in yellow) as reference, the curve which approximates the local shape of the object contour (in blue) is used to analyze the current patch. The error is clearly smaller when compared to Figure 2(a).

Quadrics were chosen because they have simple contour generators. Their apparent contour is represented by conic curves and can be easily obtained by using the theory provided by differential geometry [4]. When dealing with sparse meshes, using the conic curves instead of the original edges from the mesh makes the tracking more robust because more correct point correspondences can be found. Additionally, accuracy is also improved because conic curves gives a better approximation of the object's local shape. This also allows applying the proposed framework to diverse object shapes and since this calculation is done during the offline stage, it does not affect the computational time in the optimization step.

II. Related work

It has been shown that the apparent contour itself and its deformations contain enough geometrical information to recover the shape and camera motion of curved surfaces [4], since it is considered the dominant image feature when few or no texture is available. Some attempts aiming to estimate the structure and motion of apparent contours include the use of epipolar parameterization [4]; monocular camera but constrained to orthographic, weak-perspective and affine projection [5]; conic-stereo vision [6] or trinocular stereo [7]. In some cases, these approaches are appealing because a model of the target object is not required, though approximating the object shape with a polygonal mesh, for instance, simplify the tracking.

Furthermore, since the apparent contour changes according to the viewpoint, in order to incorporate them in standard edge based tracking methods, some adjustments are required and in some cases restrictions are imposed on the target object. In [8], a unified approach that can handle fixed and apparent contour edges within the same framework is suggested but it is limited to a certain range of motions.

Other than polygonal meshes, curved surfaces can also be represented by curved primitives or implicit surfaces [9]. Although curved primitives are simple and efficient, they are limited to a small class of shapes [10]. Implicit surfaces are more general and efficient but the models are usually very complex to be constructed by hand. In [9], the apparent contour is calculated by solving an ordinary differential equation and used to match with image edges. However, using this approach with complex objects increases the computational time due to the number of evaluations of the implicit function and its derivatives at each point.

III. Tracking framework

A. Overview

Our tracking framework uses a sparse polygonal mesh of the target object inside a standard edge-based tracking system. An overview is shown in Figure 3. The object coordinates in the world coordinate frame are represented by $\mathbf{X}_w = (x_w, y_w, z_w)$ and the pose parameters vector by $s = (r_x, r_y, r_z, t_x, t_y, t_z)^T$. Considering perspective projection, the rigid body transformation between the world and the camera coordinate frame is related by a rotation $\mathbf{R}$ and a translation $\mathbf{t}$. This transformation matrix is combined to the matrix $\mathbf{K}$, containing the camera internal parameters. The result is the matrix equation $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ used to project the current view of the object in the image plane at each iteration.

During the offline stage (A) in Figure 3, the matrix $\mathbf{K}$ is obtained by calibrating the camera using the OpenCV library. The data available in the polygonal mesh representing the object shape are used as input to create an implicit model representing local approximations of the entire object, that is, quadrics that best fit each patch. In our experiments (section V), the polygonal mesh of complex shapes was obtained by using the Range 7 3D laser scanner. Simple shapes, such as the torus, were modeled using the open source 3D modeling software Blender[1].

In the online stage (B), steps (i) to (v) are similar to a standard edge based tracking [2]. The object pose is manually initialized (ii) and a visibility test is performed to find which patches from the mesh are visible to the camera (iii). Sample points are assigned to the visible edges (iv) and used to find nearby edge points (v). For step (vi), the main changes implemented to allow the use of the apparent contour represented by conic curves are shown in Figure 4: a cost function (Equation 5) is calculated based on the distance between detected points and the conic curves on the contour. If the error value returned by this cost function is smaller than a threshold, the system

[1]http://www.blender.org/, version 2.49

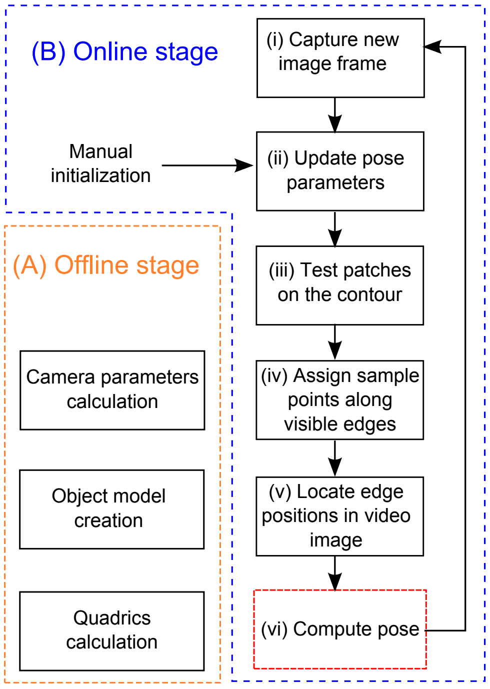

(B) Online stage
(i) Capture new image frame
Manual initialization
(ii) Update pose parameters
(A) Offline stage
(iii) Test patches on the contour
Camera parameters calculation
(iv) Assign sample points along visible edges
Object model creation
(v) Locate edge positions in video image
Quadrics calculation
(vi) Compute pose

Fig. 3. An overview of our tracking framework: in (A), the offline stage with the camera parameters calculation, model creation and quadrics calculation and in (B) the online stage, with the optimization loop highlighted in red.

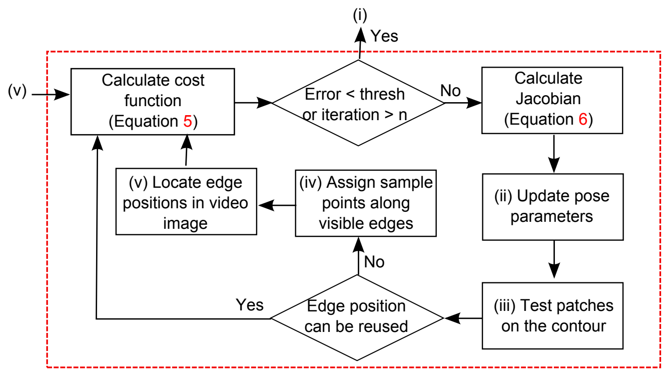

(i)
Yes
(v)
Calculate cost function (Equation 5)
Error < thresh or iteration > n
No
Calculate Jacobian (Equation 6)
(v) Locate edge positions in video image
(iv) Assign sample points along visible edges
(ii) Update pose parameters
No
Yes
Edge position can be reused
(iii) Test patches on the contour

Fig. 4. More details about the operations processed in step (vi): the error value returned by the cost function is evaluated and if it less than the threshold, the pose can be updated. Otherwise, the pose parameters are refined. Since the apparent contour changes according to the viewpoint, the edge positions are tested again to verify if the correspondence between visible edges and detected points have changed. If necessary, steps (iv) and (v) are repeated.

loops back to step (i); otherwise, the pose parameters are refined. Since a small movement of the apparent contour can change the correct correspondence between the detected edge points and the patches on the contour, after updating the pose parameters, the edge positions are tested again. If the edge positions can be reused, the tracking loop restarts; otherwise, steps (iv) and (v) are repeated.

B. Quadrics calculation

As mentioned previously, using sparse polygonal meshes to represent curved objects result in a coarse approximation of the object contour, therefore affecting tracking accuracy.

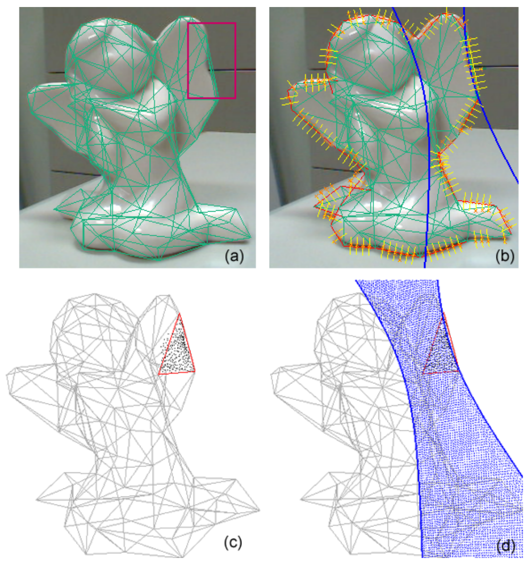


Fig. 5. (a) Coarse shape approximation after model remeshing. (b) A conic which approximates the object shape more accurately for the area highlighted in (a). In (c), the internal vertices are represented by the black dots bounded by the red triangle (current patch). (d) Quadric fitting obtained.

In addition, the remeshing process may sometimes overlook important details of the object shape. For instance, in Figure 5(a), the red box shows the region where a concave part is replaced by a straight line. These issues motivated us to develop a new representation for the patches in the mesh, considering a function simple to calculate, with better local approximations of the object, as can be seen in Figure 5(b).

Curved primitives have been used in previous approaches by dividing the object in parts and creating a suitable model for each one. For instance, truncated quadrics are used in [10]. Although it is an efficient method, its use is restricted to a small class of shapes. In our suggested representation, curved primitives represented by quadrics are also used, but our approach is not affected by the same issue because the smallest component in which the object can be divided was used, that is, the patches from the mesh.

In our *quadrics patch representation*, each patch on the mesh has a quadric equation associated. The 3D object model is constructed using triangle patches connected on the positions $V_{pki} = (x_i, y_i, z_i) \in p_k$ $(k - th$ patch), where $i = \{1, 2, 3\}$ and k represents the number of patches. When a patch p_k is located on the object contour, its projection is calculated according to the viewpoint, but only part of it is used to form the apparent contour - the length varies according to the length of the edge of p_k that is located on the contour.

In general, quadric surfaces are algebraic surfaces of degree 2 in $\mathbb{R}^3$ defined implicitly by:

$$\begin{aligned} f(x,y,z) \quad = \quad & a_1x^2 + a_2y^2 + a_3z^2 + 2a_4xy + 2a_5yz + \\ & 2a_6xz + 2b_1x + 2b_2y + 2b_3z + c = 0 \qquad (1) \end{aligned}$$

where $(a_1, a_2, a_3, a_4, a_5, a_6, b_1, b_2, b_3, c)$ are real numbers representing the quadric parameters, not all being zero.

These 10 quadric parameters are calculated for every patch by using the corresponding vertices positions V_{pk}. However, since the model is made of triangle patches, the existent data in only one patch is not enough to find a solution for this equation. To solve this problem, the *internal vertices* of each patch is used. They are defined as vertices that originally belonged to the dense mesh, but were deleted during the remeshing process. In this work, remeshing is obtained by the software QSlim provided in [11] and a boundary test is conducted to all deleted points to find which patch in the sparse mesh they belong to, as shown in Figure 5(c).

C. Quadrics evaluation

To calculate the quadrics parameters, the value $c = 1.0$ is fixed and the other nine parameters $(a_1, a_2, a_3, a_4, a_5, a_6, b_1, b_2, b_3)$ are calculated by geometric fitting. This fitting uses the internal vertices to find the quadric that best fits each patch. Hence, the sparser the mesh is, more data will be available for the calculation. If less than nine points are available, calculation of the quadrics is not possible and if this patch falls on the object contour, this edge cannot be used for tracking. For this reason, our method works better for sparse meshes than for dense meshes (see evaluation graphs in section V-A1).

Although the quadrics fitting is good for most of the patches in the mesh, in some cases the obtained fitting is not the most suitable one. Hence, to improve the overall performance of our method, the quadric fitting error is also evaluated - if it exceeds a threshold, the patch is discarded and the corresponding edge contour is not considered for tracking. For instance, Figure 6 shows the conics approximation for the patches on the contour for the current object view: the images highlighted in red are the ones discarded during tracking due to the bad fitting.

D. Contour Generation

Once the model is projected using an initial estimation of the pose, tests are performed to identify which patches from the polygonal mesh are tangent to the current viewing ray. Subsequently, for each patch, the corresponding conic is calculated, generating the object apparent contour as a collection of different conic segments (Figure 6).

Considering a point $\mathbf{X}_w$ lying on a quadric Q, the conic curves belonging to the contour can be obtained by conveniently writing Equation 1 in matrix form using homogeneous coordinates:

$$f(x, y, z) = \mathbf{X}_w^T Q \mathbf{X}_w = 0 \tag{2}$$

where Q is a 4x4 symmetric matrix:

$$Q = \left[\begin{array}{ccc|c} a_1 & a_4 & a_6 & b_1 \\ a_4 & a_2 & a_5 & b_2 \\ a_6 & a_5 & a_3 & b_3 \\ \hline b_1 & b_2 & b_3 & c \end{array}\right] = \begin{bmatrix} Q_3 & q \\ q^T & c \end{bmatrix} \tag{3}$$

Each quadric is converted to the camera coordinate frame and then to image coordinates to obtain the apparent contour,

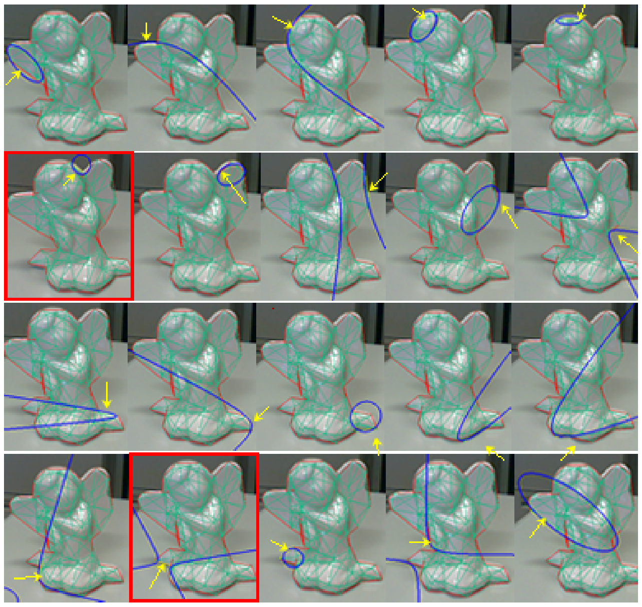


Fig. 6. Conics Tracking: conic curves (in blue) on the patches that belong to the object contour. The images highlighted by the red square show patches with a bad conic fitting for the current pose.

which is the intersection of the cone of tangent rays with the image plane at $z = f$, i.e., the points $\mathbf{x} = (x, y, f)^T$ [4]:

$$\mathbf{x}^T (qq^T - cQ_3)\mathbf{x} = 0 \tag{4}$$

where $qq^T - cQ_3$ represents the conic parameters.

E. Pose Parameters Computation

The pose parameters are calculated as a minimization problem and Levenberg-Marquardt Algorithm (LMA) [1] is used to compute the registration that minimizes the distance between the projected and the observed features. In our approach, the following cost function $e_c(s)$ is minimized:

$$e_c(s) = \frac{1}{n} \sum_{i=1}^{n} d_e(X_i, \varphi(c(Q_i, s)))^2 \tag{5}$$

where Q_i represents the quadric parameters in world coordinates, $c(Q_i, s)$ calculates the quadric parameters in camera coordinates and $\varphi(c(Q_i, s)))$is the projection of the quadric in camera coordinates in the 2D image plane resulting in a conic curve. Lastly, s represents the pose parameters vector and the function d_e represents the distance between projected features and the detected points X_i in image coordinates.

The pose s is updated by applying LMA on a Jacobian matrix $\boldsymbol{J_{es}}$ of a function describing the influence of each element of s on each element of d_e. In addition, the influence of outliers is removed by using M-Estimators. The Tukey estimator [1] is used to weight the error returned by d_e:

$$w_i = \begin{cases} \frac{k^2}{6}\left\{1 - \left[1 - \left(\frac{d_e}{k}\right)^2\right]^3\right\}, if\ |d_e| \leqslant k \\ \frac{k^2}{6}, otherwise \end{cases} \tag{6}$$

where k represents the threshold value separating the inliers and outliers points. The value of k is the double of the n^{th} value of the vector d_e in ascendant order, where n represents the number of inliers. The calculated Jacobians also need to be weighted at each iteration step, given by the derivative of Equation 6:

$$weight = \begin{cases} d_e\left[1-\left(\frac{d_e}{k}\right)^2\right]^2, if\ |d_e| \leqslant k \\ 0, otherwise \end{cases} \tag{7}$$

F. Distance evaluation

When dealing with static edges, d_e can be calculated by using the formula of the distance from a point to a line (Figure 7(a)). However, this calculation is not trivial for the apparent contour of curved surfaces because there is no closed form to calculate the distance between points to a generic implicit curve.

Some iterative approaches such as [12] have been suggested to evaluate this distance, but it makes the original minimization given by Equation 5 impractical. In [13], an analytical approximation for the Euclidean distance of a point to an implicit curve is also suggested. However, this first order approximated distance did not provide good results in our approach - the approximation had satisfactory results only when the detected point was very close to the conic.

Furthermore, in some cases the distance was erroneously calculated because the whole conic was being considered and not just the corresponding conic segment located exactly on the patch contour. In Figure 7(b), even though p_1 is closer to the detected point p_0, this distance cannot be used because it is further from the contour edge. Therefore, the development of a different strategy to evaluate this distance was necessary.

In our approach, reference points, namely $p'_1 = (x'_1, y'_1)$ and $p'_2 = (x'_2, y'_2)$, are placed on the visible contour edge and used to find two other points on the apparent contour, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, from which the distance is calculated (Figure 8). These reference points are calculated initially considering the two points of the edge from the current patch located on the contour and they are updated according to the position of the detected point. Sometimes the detected point is closer to one reference point than another, so the reference point that is further away is updated to a closer position and the orthogonal projection of the detected point can be correctly calculated. This update is necessary because their position influences the position of the points on the conic passing through the current patch.

Denoting the points p_1 and p_2 as p_i (as well as p'_1 and p'_2 as p'_i), with $i = 1, 2$ and considering $p_0 = (x_0, y_0)$ is the detected point in the video image, the point p_i on the conic curve and belonging to the line l_i passing through p_0 and p'_i can be written as:

$$\begin{cases} x_i = (x_0 - x'_i)t_i + x'_i \\ y_i = (y_0 - y'_i)t_i + y'_i \end{cases} \tag{8}$$

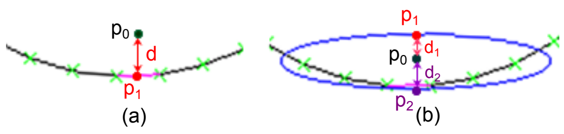


Fig. 7. (a) Distance evaluation using the edge from the mesh. (b) When dealing with conic curves, it is necessary to find the correct conic segment from which the distance will be calculated. In this case, although $d_1 < d_2$, the correct distance to be used is d_2.

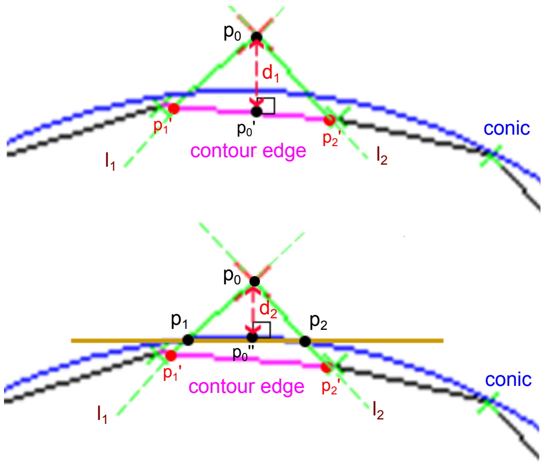


Fig. 8. (a) Standard approaches calculate the distance between p_0 and the point p'_0 on the contour edge. (b) In our approach, a new point position p''_0 is found on the conic passing through this patch to calculate the distance.

In the equation above, the values of the parameters t_i can be found by replacing Equation 8 in a general conic equation of the form:

$$f(x,y) = c_1x^2 + c_2y^2 + c_3xy + c_4x + c_5y + c_6 = 0 \tag{9}$$

where $(c_1, ..., c_6)$ are real constants and c_1, c_2, c_3 are not all zero. This resulted in the following equation:

$$\begin{aligned} &(c_1(x_0 - x'_i)^2 + c_2(y_0 - y'_i)^2 + \\ &c_3(x_0 - x'_i)(y_0 - y'_i))t_i^2 + \\ &(2c_1x'_i(x_0 - x'_i) + 2c_2y'_i(y_0 - y'_i) + \\ &c_3(x_0y'_i + x'_iy_0 - 2x'_iy'_i) + \\ &c_4(x_0 - x'_i) + c_5(y_0 - y'_i))t_i + \\ &(c_1x'^2_i + c_2y'^2_i + c_3x'_iy'_i + c_4x'_i + c_5y'_i + c_6) = 0 \end{aligned} \tag{10}$$

Equation 10 is a quadratic equation in the form $\mathbf{a}_ix^2 + \mathbf{b}_ix + \mathbf{c}_i = 0$ with components:

$$\begin{aligned} \mathbf{a}_i &= c_1(x_0 - x'_i)^2 + c_2(y_0 - y'_i)^2 + \\ &\quad c_3(x_0 - x'_i)(y_0 - y'_i) \\ \mathbf{b}_i &= 2c_1x'_i(x_0 - x'_i) + 2c_2y'_i(y_0 - y'_i) + \\ &\quad c_3(x_0y'_i + x'_iy_0 - 2x'_iy'_i) + \\ &\quad c_4(x_0 - x'_i) + c_5(y_0 - y'_i) \\ \mathbf{c}_i &= c_1x'^2_i + c_2y'^2_i + c_3x'_iy'_i + c_4x'_i + c_5y'_i + c_6 \end{aligned} \tag{11}$$

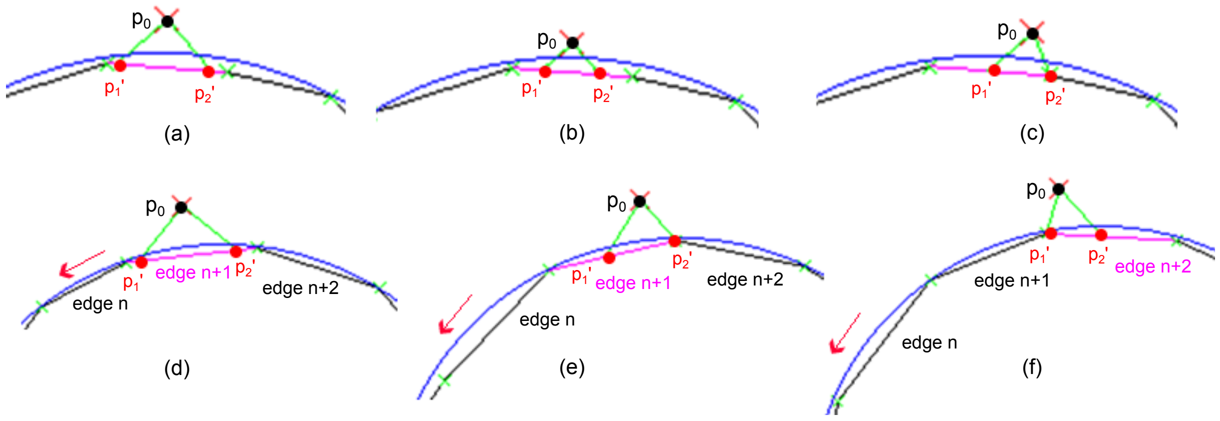


Fig. 9. (a) to (c) illustrates different relationships between the detected points and the mesh edge as well as the conic. (d) to (f) shows a slight movement of the object to the left resulting in a new correspondence between detected point and contour edge.

which can be calculated by finding the roots of the respective quadratic equations:

$$t_i = \frac{-\mathbf{b}_i \pm \sqrt{\mathbf{b}_i^2 - 4\mathbf{a}_i\mathbf{c}_i}}{2\mathbf{a}_i} \tag{12}$$

Equation 12 returns two values representing the two points where the line l_i intersects the conic. The smallest value for p_i is chosen, that is, the point closer to p_0. Finally, the distance from the point p_0 to the line passing through p_1 and p_2 is evaluated, being expressed by:

$$(y_2 - y_1)x + (x_1 - x_2)y + (x_2y_1 - y_2x_1) = 0 \tag{13}$$

whose parameters $(y_2 - y_1) = a$, $(x_1 - x_2) = b$ and $(x_2y_1 - y_2x_1) = c$ will be used to calculate the distance from a point to a line:

$$d_e = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} \tag{14}$$

Figure 9(a)-(c) shows a point p_0 in different positions and the respective relationship established with respect to the contour edge from the mesh as well as the corresponding conic. It is possible to notice that the reference point positions p'_1 and p'_2 also change according to the p_0 position.

Furthermore, since curved surfaces do not present static edges, it is necessary to constantly check the correct correspondence between p_0 and the contour edge. Figure 9(d)-(f) shows a small movement of the object to the left, which will associate p_0 to a new contour edge and hence to a different conic. In this example, *edge n+1* and *edge n+2* have similar conic curves, but depending on the object shape they may be associated with conics of different shapes. This can affect the result if the correct correspondence is not constantly verified.

IV. Tracking objects with less than 6DoF

In general, many rigid objects found in the real world have 6DoF, which means that the Jacobian matrix $\boldsymbol{J_{es}}$ has rank 6 and the estimation of the six parameters defining the object pose is possible.

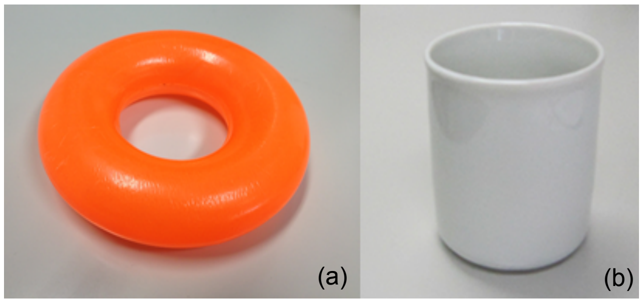


Fig. 10. Example of objects with 5DoF whose accurate pose estimation in one axis is not possible: (a) torus shaped object and (b) a cup without handle.

However, object shapes such as the ones shown in Figure 10 have only 5DoF, making $\boldsymbol{J_{es}}$ rank-deficient and hence the solution to find the pose parameters vector s cannot be determined. To allow our framework to be able to handle these kind of objects, first calculation of the measurable DoF of the target object is performed. This is achieved by counting the non-zero singular values obtained from the Singular Value Decomposition (SVD) [14] of $\boldsymbol{J_{es}}$, similar to [15].

Let n be the number of points found on the object model contour through search on the normal vector direction. Since there is one Jacobian matrix for each of these points, a $n \times 6$ matrix $\boldsymbol{J_{es}}$ is constructed and by using SVD, it can be decomposed in:

$$\boldsymbol{J_{es}} = \boldsymbol{U}\Sigma\boldsymbol{V}^T \tag{15}$$

where $\boldsymbol{U}$ is an orthogonal $n \times 6$ matrix, $\boldsymbol{V^T}$ is the transpose of an orthogonal 6×6 matrix and Σ is a 6×6 diagonal matrix containing the singular values:

$$\Sigma = diag(\sigma_1, ..., \sigma_6); \sigma_1 > \sigma_2 > ... > \sigma_6 \tag{16}$$

This decomposition can be always done, no matter how singular the matrix is, what makes possible to find the solution for the pose parameters even when matrix $\boldsymbol{J_{es}}$ has $rank$<6.

The singular values represent the degree of influence on the image of each change in the pose between frames. The bigger the singular values, the bigger the changes in the image and accurate estimation of the pose parameters is possible. If these values are small, it is more difficult to estimate the changes in the image and, hence, the pose parameters.

When the number of measurable DoF decreases, the singular values of the missing DoF becomes zero. However, due to computational approximations, these values do not become zero; instead, they become very small values. For this reason, a threshold was set to test the singular values and determine the measurable DoF of the target object. Hence, if one DoF is missing, the singular value $\sigma_6 \approx 0$; if two DoF are missing, $\sigma_5 \approx \sigma_6 \approx 0$ and so on.

V. Experimental results

Experiments were carried out in two scenarios to verify the performance of the proposed tracking system: the first with synthetic data generated by a simulator and the second with video images taken in a real environment. The machine used for the tests was a Corei7 3.20Ghz, with 8GB of RAM and NVIDIA GeForce GTX 560Ti. Comparative results are shown for our Conics Tracking (CT) method with a standard Line Tracking approach using Sparse (SLT) and Dense (DLT) meshes.

A. *Quantitative results*

In this section, quantitative results from a controlled experiment are presented. The simulator developed in [16] is used, having as input three models of the target object in different levels of quality:

(a) A high quality mesh of the target object, used as reference for the edge search step. It corresponds to a synthetic representation of the real object.
(b) A medium quality mesh to be used with DLT and having approximately 10% of the total number of patches of the mesh used in (a);
(c) A sparse mesh to be used with SLT and CT.

Depending on the object complexity and the method employed to obtain its polygonal mesh, the number of patches used in each of the items described above varies. If the model is obtained using the 3D laser scanner, the final model usually has more than 100,000 patches. However, this amount of data is neither necessary nor computationally efficient. Hence, to get the model used in item (a), the original model is simply reduced until all redundant data is deleted and the smoothness of the object is not affected.

TABLE I
Objects description

Object	Approximate size in mm (x, y, z)
Torus	(80, 80, 23)
Duck	(80, 110, 100)
Angel	(119, 69, 120)
Bunny	(120, 90, 120)

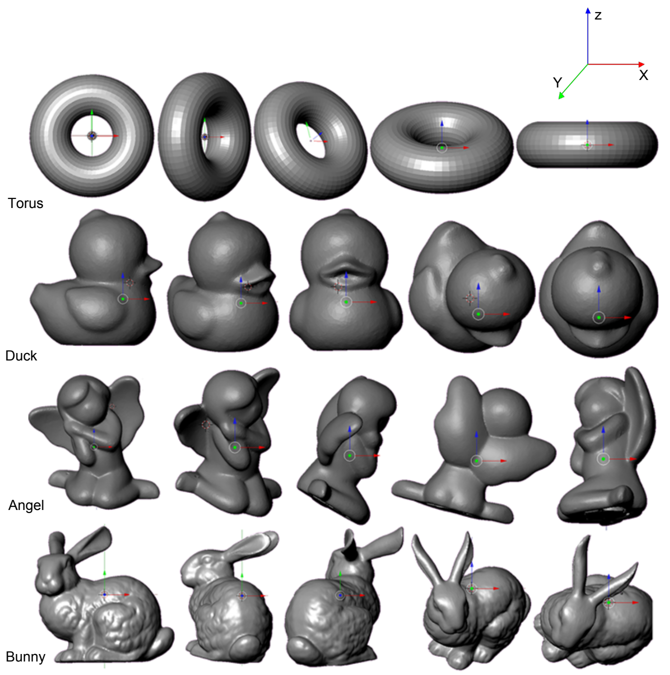


Fig. 11. The 5 poses used in the quantitative evaluation of each object. Each pose was tried 100 times with different noise values.

For item (b), after testing for several objects, it was found that, on reducing the number of patches of the high quality mesh by around 10%, the final model still retained smoothness and had a reasonable number of patches to be processed in realtime. Furthermore, the data of this polygonal mesh is used later to define the internal vertices (section III-B) used in the conics fitting. Hence, the polygonal mesh is tested to ensure that all patches, or more than 90% of the patches, have at least nine internal vertices. The larger the number of internal vertices, the better the conics fitting. In our experiments, usually it is possible to have more than 30 internal vertices for each patch of a complex shape. This is also taken into account if the polygonal mesh is manually modeled using Blender. Finally, for item (c), the number of patches is chosen according to the experiment goal as described in the next sections.

The simulation consisted of a series of 5 trials for each object, each trial having a total of 100 runs. The trials represent the object in a different initial position, manually initialized using the poses shown in Figure 11. The tracking approaches are evaluated by computing the camera displacement between these initial positions and a new pose generated by the simulator by adding Gaussian noise to them.

The mesh size of the four different objects' models (torus, duck, angel and bunny[2]) used in this experiment can be found in Table I. The distance of the objects to the camera was fixed to 350mm and the camera internal parameters were obtained from a Logitech QuickCam Fusion webcam calibrated with the OpenCV library.

[2]3D model of the bunny object is available at http://graphics.stanford.edu/data/3Dscanrep/

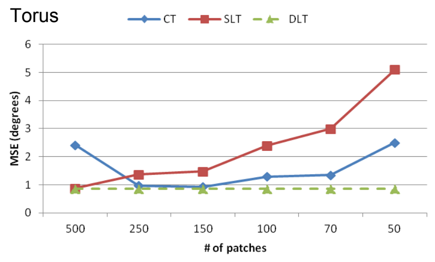
To simulate similar noise that can be found when video images are taken from the real environment, Gaussian noise with mean equal to zero and standard deviation of $\sigma_n = 2.0$ was added to the edge detection process.

Two experiments were performed in this context: one varying the number of patches in the mesh to verify the relationship between the quality of the mesh and the tracking accuracy; the other evaluates the tracking accuracy considering different amounts of noise by changing the simulator parameters and fixing the number of patches.

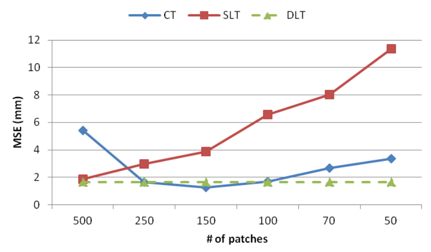
1) Experiment A: In this first experiment, the main goal is to compare the performance of the evaluated methods by varying the number of patches in the polygonal mesh. This range varies according to the object complexity.

Since the torus has a simple shape, a range from 500 to 50 patches was used. For the duck, 750 to 50 patches was chosen, while, for the angel and the bunny, a range from 1,000 to 100 patches was more suitable. The noise values used to generate the poses were generated with standard deviation of $\sigma_{a1} = 1.0°$ and $\sigma_{a2} = 10.0mm$ per frame, for each axis in the rotation (r_x, r_y, r_z) and translation (t_x, t_y, t_z), respectively.

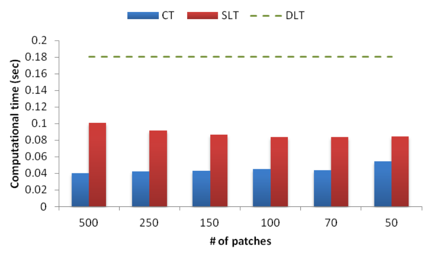
Accuracy (Mean Squared Error (MSE) of the angle and the distance components of the pose vector *s*), *computational time* (average processing time) and *robustness* (success rate in 100 runs) were evaluated. For the last item, the success rate is based on a threshold used to compare the error value returned by the cost function: if it is less than approximately 3.0, the tracking succeeds; otherwise, it is considered a failure.



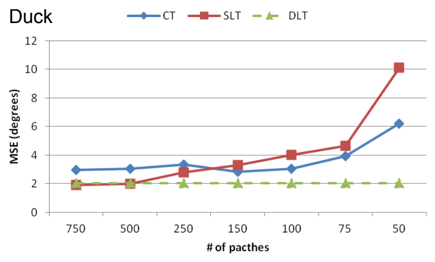
(a)



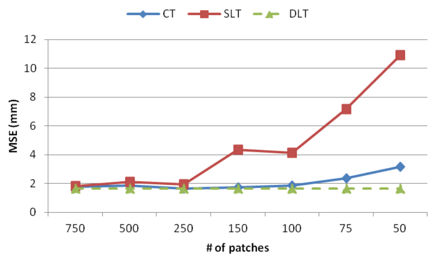
(b)



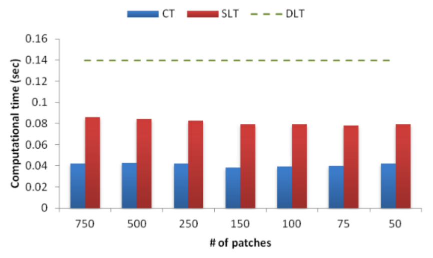
(c)



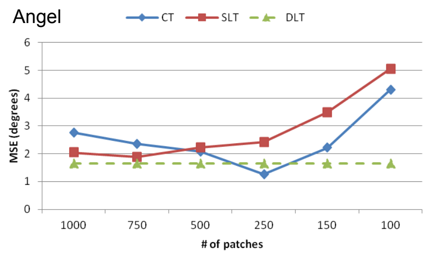
(d)



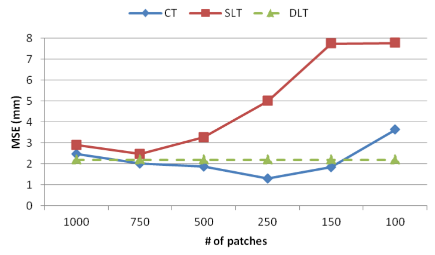
(e)



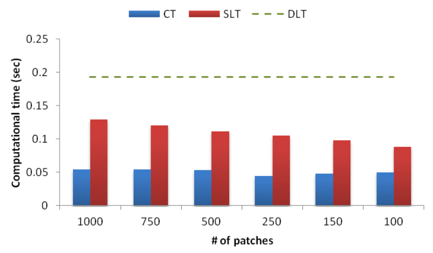
(f)



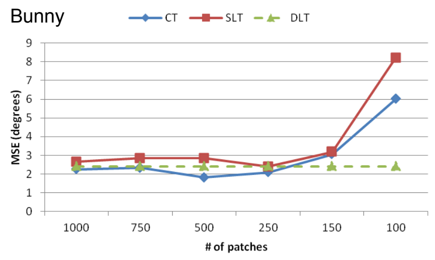
(g)



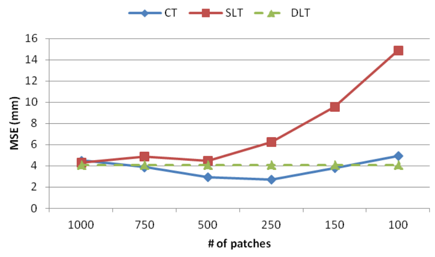
(h)



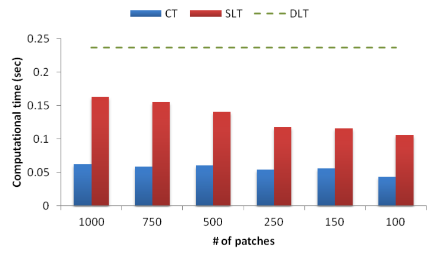
(i)



(j)



(k)



(l)

Fig. 12. Results for accuracy and computational time of experiment A, where each row represents one object: torus, duck, angel and bunny, in this order. **Left**: MSE for the rotation in degrees. **Middle**: MSE for translation in mm. **Right**: Average of the computational time.

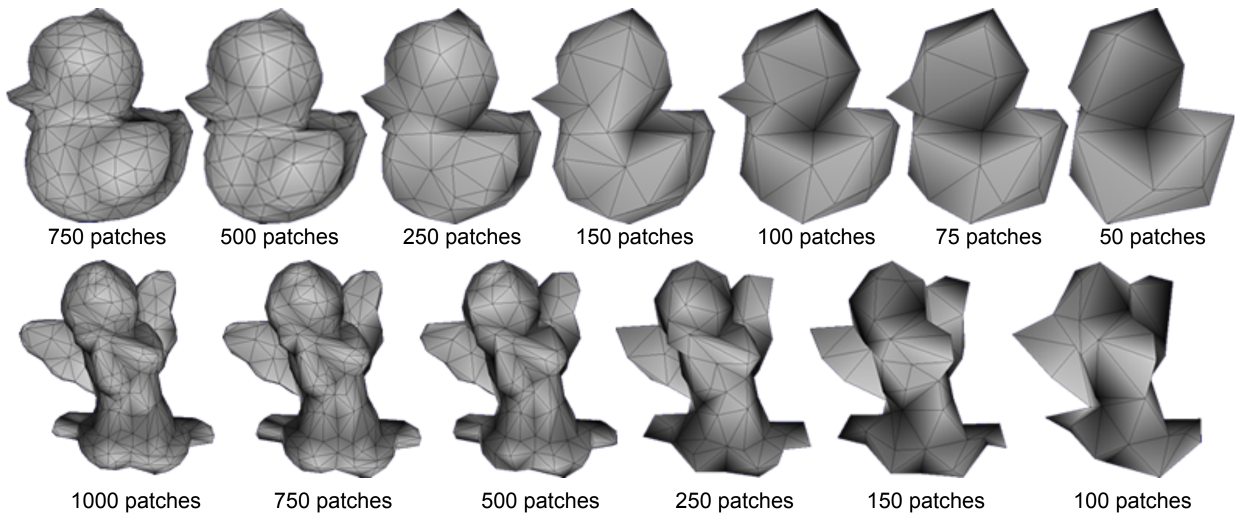


Fig. 13. Polygonal meshes used in the experiments of the duck and the angel objects: when the remeshing is overdone, accuracy of CT becomes lower because of changes in the original shape of the object.

The graphs in Figure 12 show the accuracy results obtained for all objects in the 5 poses according to the number of patches of the polygonal mesh. It is important to note that the variation of the number of patches used in this experiment is valid only to CT and SLT results. All results for DLT considered a fixed number of patches (2.500 patches) and they were plotted on the same graph only for comparison reasons.

Graphs (a)(d)(g)(j) summarize the results for the rotation component. When the number of patches is high, SLT performs better than CT, with MSE values close to the ones calculated by DLT. However, as the number of patches decreases, the SLT performance also decreases and at a certain point, the MSE of CT gets better results. Similar behavior can be seen in graphs (b)(e)(h)(k) in the middle, with the results for the translation components: the MSE for CT decreases as the number of patches decreases. In some cases, CT results are better than the results presented by DLT, such as in graph (h) for the angel with 250 and 150 patches; and in graph (k), for the bunny with 500 and 250 patches.

Furthermore, while the error of SLT keeps increasing with the decrease of the number of patches, the error of CT decreases until it reaches an optimal point where the error starts to increase again (though with values still lower than SLT). This behavior happens because there is a limit for the object's model remeshing: when it is overdone, the changes in the shape no longer resemble the original shape in some parts, affecting the quadrics fitting and hence the tracking accuracy.

To illustrate the remeshing effects on CT, in Figure 13 the beak of the duck is missing when it has 50 patches and part of the wing and the leg of the angel are missing when it has 100 patches. These numbers of patches represent the point in which CT has the highest error, as can be seen in the graphs of MSE in Figure12. Hence, depending on the object's shape complexity, the ideal number of patches to be used with CT changes, being determined by visual evaluation of the remeshing process.

Regarding computational time, the average of the results of each method is shown in Figure 12(c)(f)(i)(l). Among the three methods, CT has the best results.

In the cases where simple shapes are used (torus and duck), the computation time varies comparatively less than other methods with the decrease of the number of patches. However, for complex objects (angel and bunny), the trade-off between computational time and accuracy is clear for SLT: the lower the number of patches the faster the algorithm performs, but at the same time, MSE becomes higher. On the other hand, CT performs well in both accuracy (up to the optimal point) and computation time.

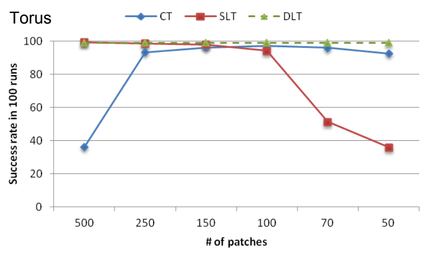
Finally, Figure 14 shows the average of the success rate in 100 runs, with CT performing better than SLT when the number of patches is low. The dashed green line represents the values obtained for DLT with fixed number of patches. For simple shapes, such as the torus and the duck, the values of CT are very close to DLT in most cases, indicating high success rate even for a reduced number of patches. A decrease in the success rate for CT is noticed for the angel and the bunny, but with equivalent or higher values than SLT when the number of patches is low.

2) Experiment B: In this experiment, the polygonal mesh has a fixed number of patches, but the simulator parameters are changed following the conditions below:

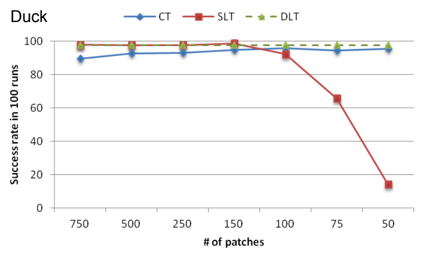
(a) Condition 1: $\sigma_{b11} = 1.0°$ and $\sigma_{b12} = 10.0mm$.
(b) Condition 2: $\sigma_{b21} = 1.5°$ and $\sigma_{b22} = 15.0mm$.
(c) Condition 3: $\sigma_{b31} = 3.0°$ and $\sigma_{b32} = 15.0mm$.

These conditions are used to set the standard deviation of the Gaussian noise applied to the rotation and translation components, respectively.

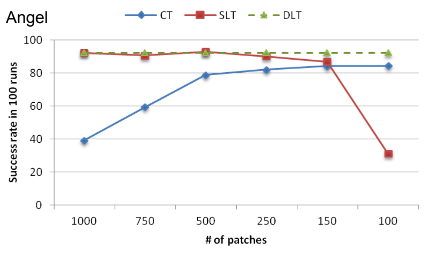
The number of patches to be used for each object was chosen after the analysis of Figure 12 and Figure 14. They basically represent the point where CT had better performance.



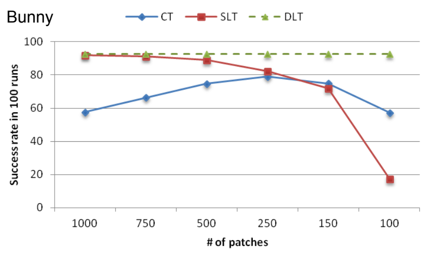
(a)



(b)



(c)



(d)

Fig. 14. Results for the success rate in 100 runs of experiment A. The success rate of SLT drastically decreases when the number of patches is reduced, while in CT, a slight or no decrease is noticeable.

In our experiments, the following number of patches were chosen:

- Torus: 150 patches
- Duck: 100 patches
- Angel: 250 patches
- Bunny: 250 patches

The graphs in Figure 15 show the accuracy and success rate results obtained for all objects using the same 5 poses of the previous experiment. For all conditions, CT presented better accuracy than SLT in both, rotation and translation components (graphs (a)(d)(g)(j) and (b)(e)(h)(k)). This is consistent with Table II, in which the average of the re-projection error for each of the tested conditions was lower for CT.

Regarding the success rate, in most of the cases, SLT performed slightly better than CT. Since bigger amounts of noise means larger displacements from the original pose, it is expected the re-projection error also increases, affecting the success rate. On the other hand, comparing the success rate graphs from Figure 15 with graphs (a), (c) and (d) of Figure 14, the torus (150 patches), angel (250 patches) and bunny (250 patches) also had lower success rate. Hence, the increase in noise lowered the success rate of these objects but it is still consistent to the expected result.

The graphs for computational time are omitted because the number of patches is fixed. The results for each object can be verified in the corresponding graphs (c)(f)(i)(l) of Figure 12.

DLT results are also presented in Table II for comparison purposes. CT results are in some cases better than DLT. This confirms that our proposed tracking system using sparse polygonal meshes and conics can be as accurate as a standard line tracking using dense meshes, with the advantage that it consumes less computational time.

TABLE II
RE-PROJECTION ERROR AVERAGE (MM)

Object	Conditions	CLT	SLT	DLT
Torus	Condition 1	0.876564	1.2456	0.9039172
	Condition 2	0.8785856	1.284532	0.962884
	Condition 3	0.9089266	1.298442	0.971726
Duck	Condition 1	0.9008606	1.954882	0.8850602
	Condition 2	0.9176128	1.944626	0.912994
	Condition 3	0.9526356	1.936006	0.9215902
Angel	Condition 1	0.8737294	1.445286	0.936935
	Condition 2	0.8873462	1.471256	1.0600578
	Condition 3	0.8891558	1.46911	1.0177716
Bunny	Condition 1	0.9894448	1.592458	0.967533
	Condition 2	1.003952	1.701384	1.1316334
	Condition 3	1.078521	1.707916	1.122874

B. *Qualitative results*

The previous experiments provided numerical results to quantify the tracking performance. In this section, qualitative evaluation is implemented using video sequences captured from the real world. It aims to confirm that CT can perform well not only in a synthetic environment, but also in a real environment, where the presence of noise is bigger and the movement of the object or/and the camera is more complex.

A handheld monocular camera Logitech QuickCam Fusion was used and to ensure a fair comparison, all approaches were tested using the same video sequence, recorded beforehand. To visualize the tracking, the polygonal mesh is rendered on the object's surface: when tracking succeeds, the mesh is green; if it fails, the mesh becomes pink. Finally, performance is evaluated by counting the number of frames in which the tracking fails.

Two objects were used in this experiment: Torus and Angel. In the first video sequence, the user holds the torus with his hand and moves it arbitrarily. An example of this sequence is shown in Figure 16, where the upper images show CT results without any failures and the bottom sequence shows SLT failing for some of the poses. The failure in SLT happens after making an horizontal movement with the torus, though tracking is recovered in the next frames.

For the angel's sequence, two conditions were applied:

(a) Angel I: The user moves the object with his hands.
(b) Angel II: The camera moves around the object.

Similar to the torus sequence, Figure 17 shows some CT results in the upper sequence, which do not have any failures. In the bottom sequence, SLT fails in some of the poses.

In the *Angel I* condition, SLT fails after rotating the object and tracking is recovered when the object returns to a pose close to the initial pose. In the *Angel II* condition, SLT fails when the camera gets closer to the object and moves away. However, after this failure, tracking is not recovered until the end of the sequence. These results show the robustness of CT regarding different movements made either by moving the object or by moving the camera.

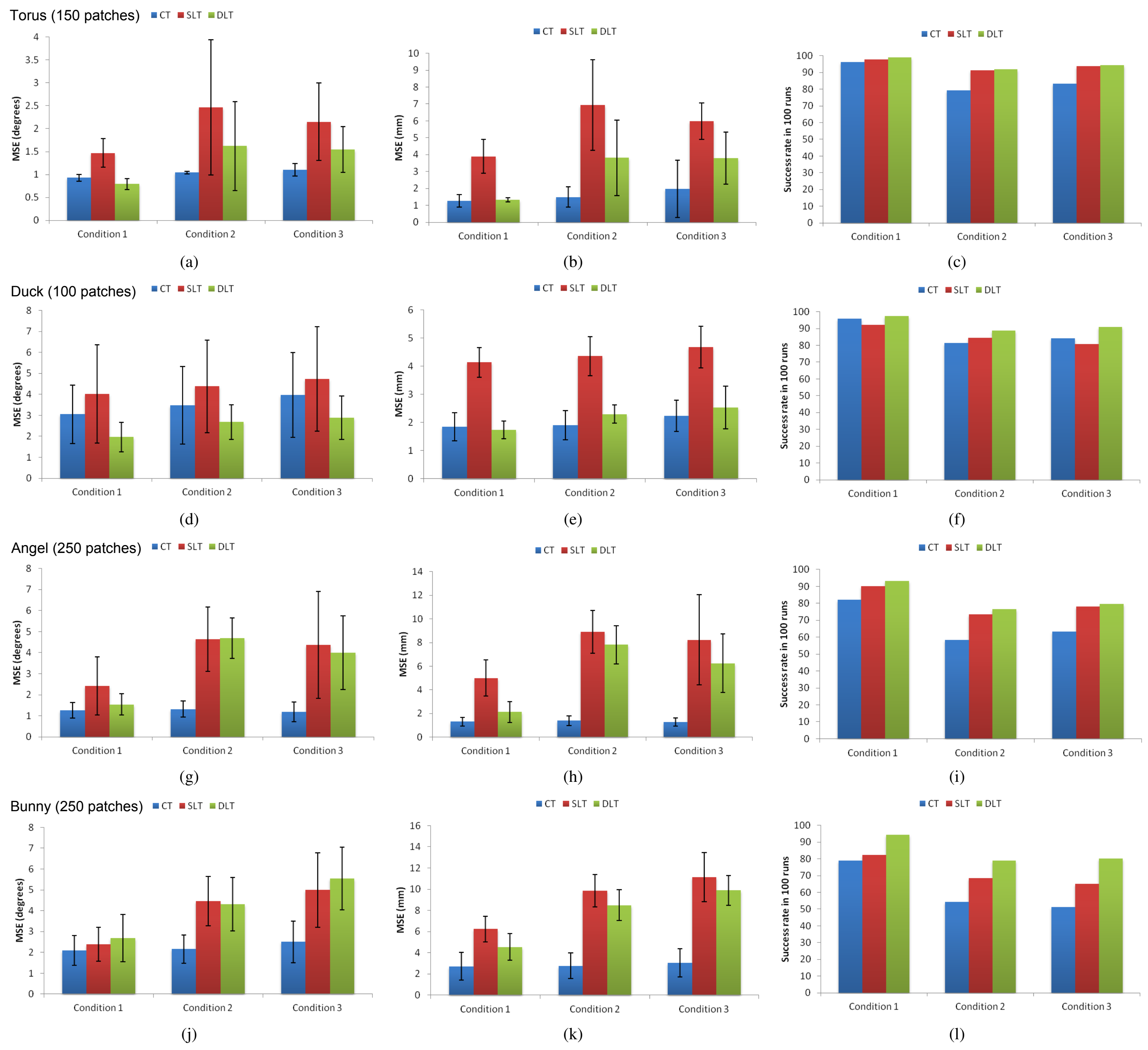


Fig. 15. Results for accuracy and success rate of experiment B. **Left**: MSE for the rotation in degrees. **Middle**: MSE for translation in mm. **Right**: Average of the success rate in 100 runs.

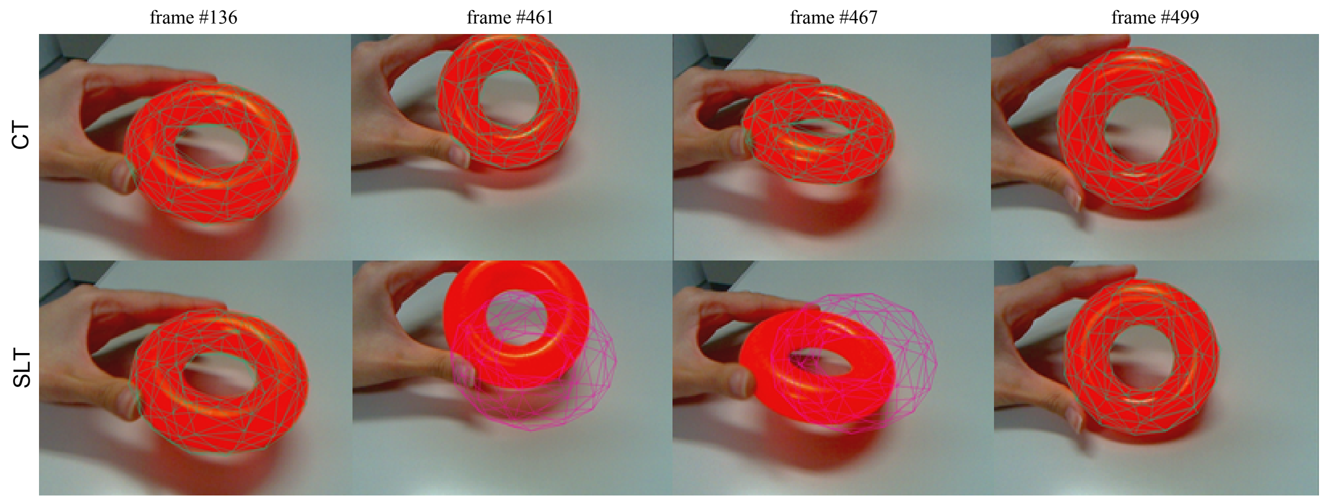


Fig. 16. Qualitative evaluation for the torus. In the upper sequence, CT does not present any failure. In the bottom sequence, SLT fails in some poses.

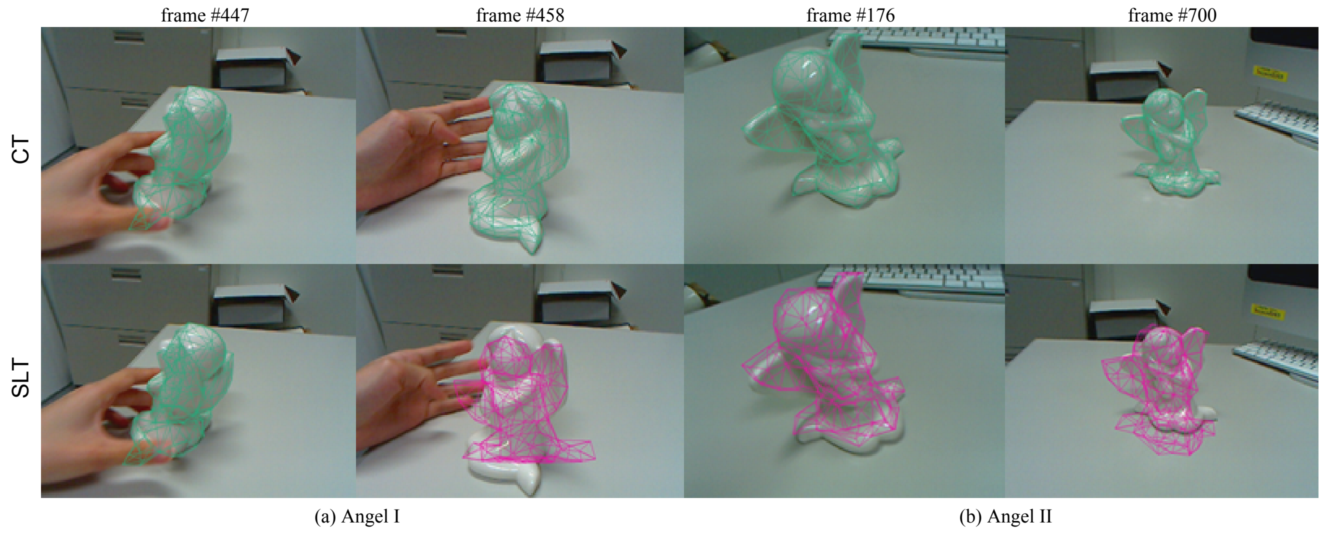


Fig. 17. Qualitative evaluation for the angel. Two conditions are considered: (a) the user moving the object (two images on the left) and (b) the camera moving around the object (two images on the right). In the upper sequence, CT does not present any failure. In the bottom sequence, SLT fails in some poses.

Table III shows more details about the results from these experiments, with the number of frames recorded in each video sequence and the respective number of frames in which each approach failed. They confirmed the results obtained in the quantitative experiments, with CT performing better than SLT when using sparse polygonal meshes. The video sequences showing the comparison between CT and SLT are available at: http://imd.naist.jp/videos/quadrics.html.

TABLE III
NUMBER OF FAILURES

Object	**# of Frames**	**CT failures**	**SLT failures**
Torus	605	- (0%)	69($\approx$ 11.4%)
Angel I	749	- (0%)	89($\approx$ 14.7%)
Angel II	702	- (0%)	532($\approx$ 87.93%)

VI. Augmented Prototyping application

In Augmented Reality (AR), virtual imagery is properly overlaid inside the real world to give the user the feeling that the virtual and real worlds coexist, though this coexistence is strongly influenced by the accuracy of the registration between virtual and real objects is [17]. Different fields have been using AR for a wide variety of purposes: to improve the user experience, such as in the entertainment industry; to give guidance during assembly operations or to include virtual annotations in the real world to give directions as well as information about specific places nearby the user.

In this section, we present an application of our proposed CT method in AR for Rapid Prototyping (RP). With RP, a physical prototype of the desired product can be automatically constructed from Computer-Aided Design (CAD) data by using a 3D printer. By enhancing this prototype with AR, evaluation of aesthetic concepts of the final product (e.g. color

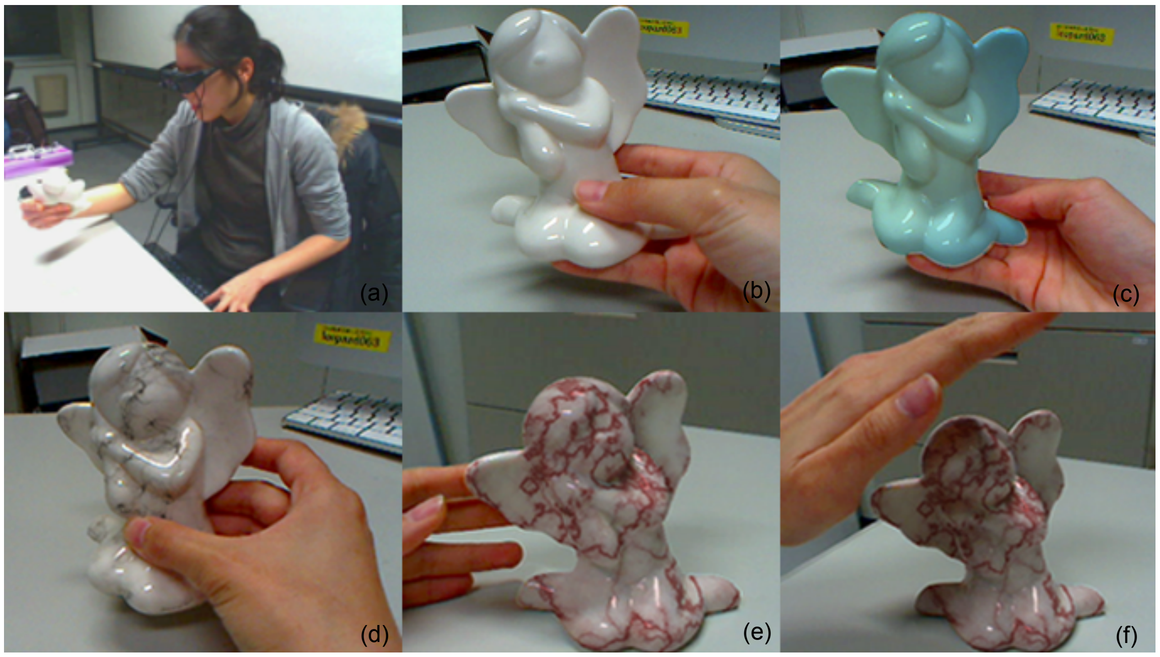


Fig. 18. Image sequence showing the AP application: (a) the ideal scenario, with the user holding the angel figurine and wearing a Head-Mounted Display, from where it is possible to visualize different colors and textures rendered on the object in real-time. In the sequence, (b) the original object, without any texture, followed by (c) an augmented image of the same object with a blue color. Different textures were also tried and the object is shown in different views (d)-(f). In (f), the detail of the shadow of the user's hand is smoothly blended with the object's virtual texture and realistically rendered.

or texture) in real-time and inside the users' environment becomes easier, saving both time and production costs. This usage of augmentation in RP has been previously proposed, being referred as Augmented Prototyping (AP) [18], [19], [20].

In our AP application, curved shapes are the main target and different versions of the planned design can be overlaid on the prototype in real-time and easily interchanged without the need of producing new prototypes. The user can interact with his hands and place it in his own environment as he would do with the final product.

The CAD model used to construct the physical prototype is employed inside the proposed tracking system. Hence, the prototypes do not need to have any kind of texture information and no fiducial markers are required. In Figure 18, an image sequence shows our AP application: by using a Head-Mounted Display, the user can have a natural view of the object and visualize different colors and textures as they are rendered onto the object in real-time. In (f), the detail of the shadow of the user's hand is smoothly blended with the object's virtual texture and realistically rendered.

VII. Conclusion and Future Work

In this paper, we presented a novel method to solve the trade-off between computational efficiency and tracking accuracy when dealing with textureless rigid curved objects.

Quadrics are suggested to represent each patch of the mesh, being a simple representation, easy to calculate and efficient for curved objects. A new method for evaluating the distance between projected and detected features was also developed to attend the particularities of using the quadrics patch representation. Furthermore, a method to calculate the DoF of the target object is also included in our framework, allowing the tracking of objects with less than 6DoF.

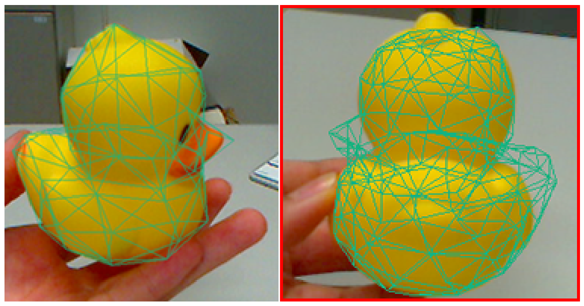


Fig. 19. Highlighted by the red square, a pose incorrectly handled by our framework.

Quantitative results comparing our approach and the traditional method using sparse and dense meshes are presented in two stages. First, by using several polygonal meshes of the same object with different number of patches to verify their affect on accuracy, speed and success rate of each method. Second, by choosing one mesh from the previous experiment and testing it under different conditions, with simulation of small and large movements made with the object.

While the quantitative experiments are performed with a simulator, qualitative results are presented by using video images captured from the real environment. Thus, success and failure conditions can be evaluated in more realistic conditions, confirming the quantitative results previously presented.

Our current implementation showed good results in both synthetic and real environment, but there are still some issues to be solved. Currently our framework is able to handle objects with less than 6DOF by using SVD as shown in section IV. However, it cannot properly handle situations where the object starts with 6DOF but depending on the viewpoint, the DOF changes. For instance, in Figure 19, when the duck is rotated, at some point the shape becomes symmetric and 1DOF is lost. The tracking does not fail, but the model is not correctly rendered on the object. This problem can be solved by incorporating into our framework the approach developed in [15] which, in order to recover the object DoF, performs null space search on the Jacobian matrix relating model pose and detected features.

Finally, an application of our proposed method in AP of rigid curved surfaces is presented, with examples from a real scenario using the angel figurine. Different colors and textures could be easily tested using only one physical prototype.

Other applications in which our method can be applied include augmented reality targeting mobile devices, in which the data size of the 3D model can be critical for the tracking performance. By using sparse meshes, loading 3D models from a remote sever would be faster as well as the tracking itself, with less model data to be analyzed during tracking.

In order to allow a large variety of objects to be tracked by our framework, our method will be extended to deal with curved objects that are not entirely smooth. An example can be the cup shown in Figure 10(b), in which the upper and bottom part of it is not smooth. To obtain more accurate tracking for this type of objects, a patch classification step can be performed during the offline stage to identify which patches in the mesh are smooth and which are not. Then, for non-smooth patches, depending on the edge that is located on the contour, a different Jacobian and matching method could be triggered.

REFERENCES

[1] V. Lepetit and P. Fua, "Monocular model-based 3d tracking of rigid objects: A survey," *Foundations and Trends in Computer Graphics and Vision*, 2005.

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

[3] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette, "Real-time markerless tracking for augmented reality: The virtual visual servoing framework," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 615–628, 2006.

[4] R. Cipolla and P. Giblin, *Visual Motion of Curves and Surfaces*. Cambridge University Press, 2000.

[5] Y. Furukawa, A. Sethi, J. Ponce, and D. Kriegman, "Robust structure and motion from outlines of smooth curved surfaces." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 2, pp. 302–315, 2006.

[6] S. Ma, "Conics-based stereo, motion estimation and pose determination," *International Journal of Computer Vision*, vol. 10, no. 1, pp. 7–25, 1993.

[7] T. Joshi, N. Ahuja, and J. Ponce, "Structure and motion estimation from dynamic silhouettes under perspective projection," *International Journal of Computer Vision*, vol. 31, no. 1, pp. 31–50, 1999.

[8] G. Li, Y. Tsin, and Y. Genc, "Exploiting occluding contours for real-time 3d tracking: A unified approach." in *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[9] E. Rosten and T. Drummond, "Rapid rendering of apparent contours of implicit surfaces for realtime tracking," in *British Machine Vision Conference*, 2003, pp. 719–728.

[10] T. Drummond and R. Cipolla, "Real-time tracking of highly articulated structures in the presence of noisy measurements," in *Proceedings of the Eighth International Conference on Computer Vision*, 2001, pp. 315–320.

[11] M. Garland, "Quadric-based polygonal surface simplification," Ph.D. dissertation, Carnegie Mellon University, 1999.

[12] V. H. Mederos, J. Sarlabous, and P. Sanchez, "A new algorithm to compute the euclidean distance from a point to a conic." *Revista Investigacion Operacional*, vol. 23, no. 2, 2002.

[13] G. Taubin, "Distance approximation for rasterizing implicit curves," *ACM Transactions on Graphics*, vol. 13, no. 1, pp. 3–42, 1994.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: the art of scientific computing*, 3rd ed. Cambridge University Press, 2007.

[15] K. Kumagai, M. A. Oikawa, T. Taketomi, G. Yamamoto, J. Miyazaki, and H. Kato, "Robust model-based tracking considering changes in the measurable dof of the target object," in *Proceedings of the 21st International Conference on Pattern Recognition*, 2012.

[16] M. A. Oikawa, G. Yamamoto, M. Fujisawa, T. Amano, J. Miyazaki, and H. Kato, "Quantitative evaluation method for model-based tracking of 3d rigid curved objects," in *Proceedings of The 2nd International Workshop on AR/MR Registration, Tracking and Benchmarking*, 2011.

[17] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Computer Graphics and Applications*, pp. 34–47, 2001.

[18] J. Verlinden and I. Horvath, "A critical systems position on augmented prototyping systems for industrial design," in *Proceedings of the ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2007.

[19] W. Lee and J. Park, "Augmented foam: A tangible augmented reality for product design," in *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2005, pp. 106–109.

[20] H. Park, H. C. Moon, and J. Y. Lee, "Tangible augmented prototyping of digital handheld products," *Computers in Industry*, vol. 60, pp. 114–125, 2009.