



## 奈良先端科学技術大学院大学 学術リポジトリ

Nara Institute of Science and Technology Academic Repository: naistar

<b>Title</b>	Local quadrics surface approximation for real-time tracking of textureless 3D rigid curved objects
<b>Author (s)</b>	Marina Atsumi Oikawa, Takafumi Taketomi, Goshiro Yamamoto, Makoto Fujisawa, Toshiyuki Amano, Jun Miyazaki, Hirokazu Kato
<b>Citation</b>	2012 14th Symposium on Virtual and Augmented Reality, Rio Janiero, Brazil 28-31 May 2012
<b>Issue Date</b>	2012-5
<b>Resource Version</b>	author
<b>Rights</b>	© 2012 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
<b>DOI</b>	10.1109/SVR.2012.3
<b>URL</b>	<a href="http://hdl.handle.net/10061/13342">http://hdl.handle.net/10061/13342</a>

# Local quadrics surface approximation for real-time tracking of textureless 3D rigid curved objects

Marina Atsumi Oikawa, Takafumi Taketomi,  
Goshiro Yamamoto  
Nara Institute of Science and Technology, Japan  
{marina-o, takafumi-t, goshiro}@is.naist.jp

Toshiyuki Amano  
Yamagata University, Japan  
amano@yz.yamagata-u.ac.jp

Makoto Fujisawa  
University of Tsukuba, Japan  
fujis@slis.tsukuba.ac.jp

Jun Miyazaki, Hirokazu Kato  
Nara Institute of Science and Technology, Japan  
{miyazaki, kato}@is.naist.jp

**Abstract**—This paper addresses the problem of tracking textureless rigid curved objects. A common approach uses polygonal meshes to represent curved objects and use them 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 for each patch in the mesh to give local approximations of the object shape. The novelty of our research lies in using curves that represent the quadrics projection in the current viewpoint for distance evaluation instead of using the standard method which compares edges from mesh and detected edges in the video image. This representation allows to considerably reduce the level of detail of the polygonal mesh and led us to the development of a novel method for evaluating the distance between projected and detected features. The experiments results show the comparison between our approach and the traditional method using sparse and dense meshes. They are presented using both synthetic and real image data.

**Keywords**—model-based tracking, camera pose estimation, quadrics, rigid curved objects.

## 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. For rigid objects, this means to continuously recover 6 DOF parameters representing the object position and orientation relative to the camera while it moves around the scene [1].

This paper focuses on model-based tracking that considers the object edges while doing tracking, similar to the approaches presented in [2], [3]. A CAD model of the target object is used for matching with the edge information found on 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 finally obtained after an optimization process.

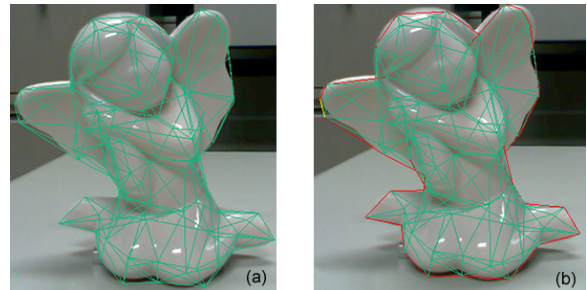


Figure 1. In (a), a sparse mesh is rendered on the target object resulting in (b) a coarse representation of the object contour (red lines).

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 high quality meshes are required to accurately recover the object 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 (Figure 1(a)) results in a coarse approximation of the object contour (Figure 1(b)). Hence, the error between projected and detected edge points increases, thus affecting accuracy. Furthermore, curved objects do not present static edges; instead, their main feature is the *apparent contour* - a curve formed by the projection of the *contour generator*, that is, parts of the surface that are tangent to the viewing ray [5].

### 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. 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(a), instead of using the model edge (in yellow) as reference, the curve which approximates the shape of the object contour (in blue) is used to analyze the current patch. The error is clearly smaller when compared to Figure 2(b).

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 [5]. 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 approximate better the object local shape. This also makes possible to apply 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 [5], being considered the dominant image feature when few or none texture information is available. Some attempts aiming to estimate the structure and motion of apparent contours include the use of epipolar parameterization [5]; monocular camera but constrained to orthographic, weak-perspective and affine projection models [6]; conic-stereo vision [7] or trinocular stereo [8]. In some cases, these approaches are attractive because a model of the target object is not required, though approximating the object shape with a polygonal mesh, for instance, simplify the tracking. On the other hand, since the apparent contour changes according to the viewpoint, in order to allow their use with standard edge based tracking methods, some adjustments are required and in some cases restrictions are imposed on the target object.

In [9], an 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 [10]. Although curved primitives are simple and efficient, they are limited to a small class of shapes [11]. Implicit surfaces are more general and efficient but the models are usually very complex to be constructed by hand. In [10], 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 computational time due to the number of evaluations of the implicit function and its derivatives at each point.

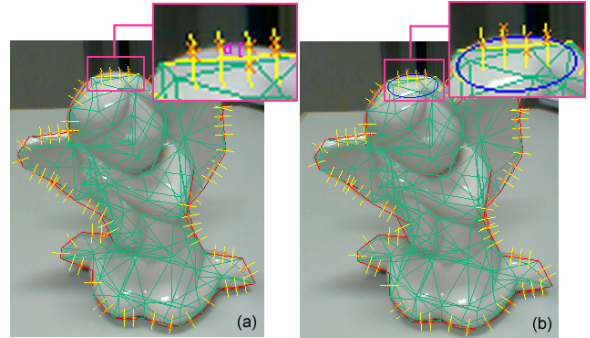


Figure 2. (a) When the edges from the mesh are used alone with sparse meshes, a big error between the sample points on the model and the detected edge points is evaluated. (b) However, by approximating each patch by a quadric surface, the projection represented by conic curves (blue line) makes a better approximation of the object outline - the conic curve coincides to the object outline on the highlighted part.

## III. TRACKING FRAMEWORK

### A. Overview

Our tracking framework uses a sparse polygonal mesh of the target object inside a standard edge-based tracking system. During the offline stage, the data available in the mesh are used as input to create an implicit model representing local approximations of the entire object, that is, quadrics that best fit each patch.

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 is combined to the camera internal parameters  $\mathbf{K}$ , resulting in a 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.

An overview of our tracking framework is showed in Figure 3. Steps (i) to (iv) are similar to a standard edge based tracking [2]. The object pose is manually initialized (i) and a visibility test is performed to find which patches from the mesh are visible to the camera (ii). Sample points are assigned to the visible edges (iii) and used to find nearby edge points (iv). For step (v), the main changes implemented to allow the use of the apparent contour represented by conic curves are shown in more details in Figure 4.

A cost function (Equation 5) is calculated based on the distance between detected points and the conic curves on the contour. Since a small movement of the apparent contour can change the correct correspondence between the detected edge points and the patches on the contour, this correspondence is tested inside the optimization step and if necessary, steps (iii) and (iv) are repeated.

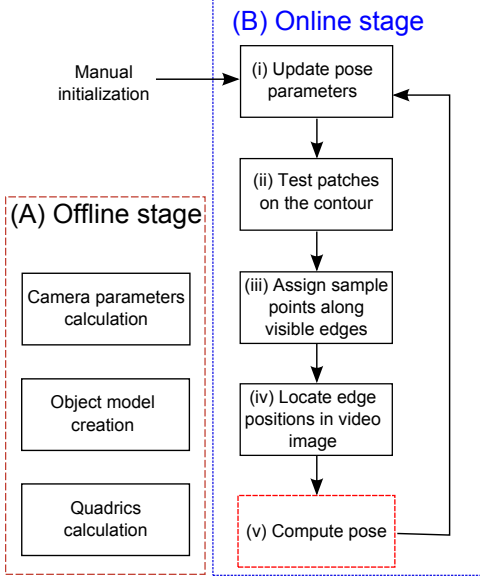


Figure 3. An overview of our tracking framework.

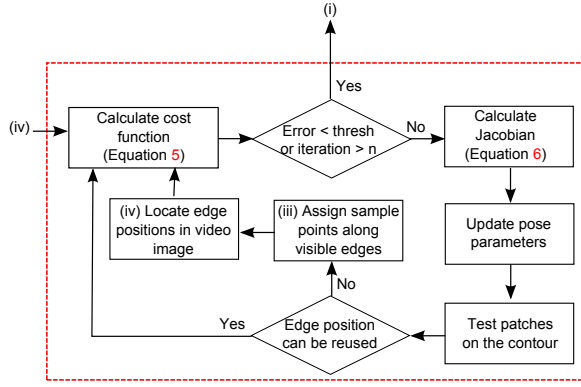


Figure 4. Step (v) explained in more details.

### B. Quadratics patch representation

As mentioned previously, using sparse polygonal meshes to represent curved objects will result in a coarse approximation of the object contour, therefore affecting tracking accuracy. In addition, the remeshing process may sometimes overlook important details of the object shape. 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 that would be simple to calculate and give 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 [11]. Although it is an efficient method, its use is restricted to a small class of shapes. In our suggested representation,

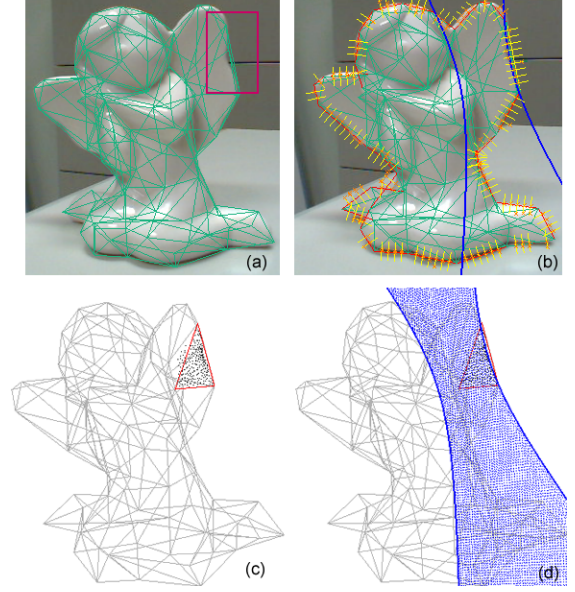


Figure 5. (a) Coarse shape approximation after model remeshing. (b) A conic which approximates the object shape more accurately for the area highlighted in (a). In (c), the patch internal vertices are represented by the black dots bounded by the red triangle (current patch). (d) Quadric fitting obtained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

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  $V_{pk} = (x_i, y_i, z_i) \in p_k$  ( $k$ -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:

$$f(x, y, z) = a_1x^2 + a_2y^2 + a_3z^2 + 2a_4xy + 2a_5yz + 2a_6xz + 2b_1x + 2b_2y + 2b_3z + c = 0 \quad (1)$$

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, using *internal vertices* of each patch is suggested, which 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 [12] and a bounding test is conducted to all deleted points to find which patch in the sparse mesh they belong to, as shown in Figure 5(c). A geometric fitting using the internal vertices is applied to find the quadric that best fits each patch (Figure 5(d)). Additionally, to improve the overall performance of our method, this quadric fitting error is also evaluated - if the error exceeds a threshold, the patch is discarded and the corresponding edge contour is not considered for tracking.

### C. 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 12).

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:

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

where  $Q$  is a 4x4 symmetric matrix:

$$Q = \begin{bmatrix} a_1 & a_4 & a_6 & b_1 \\ a_4 & a_2 & a_5 & b_2 \\ a_6 & a_5 & a_3 & b_3 \\ b_1 & b_2 & b_3 & c \end{bmatrix} = \begin{bmatrix} Q_3 & q \\ q^T & c \end{bmatrix} \quad (3)$$

Each quadric is converted to the camera coordinate frame and then to image coordinates to obtain the apparent contour, which is the intersection of the cone of tangent rays with the image plane at  $z = f$ , i.e., the points  $\mathbf{x} = (x, y, f)^T$  [5]:

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

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

### D. 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:

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

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  $J$  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  $d_e$ :

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

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:

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

### E. 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 6(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 [13] have been suggested to evaluate this distance, but it makes the original minimization given by Equation 5 impractical. In [14], an analytical approximation to 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 it was considering the whole conic and not just the corresponding conic segment located exactly on the patch contour. In Figure 6(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.

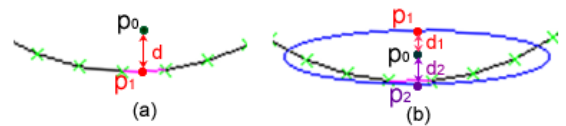


Figure 6. (a) Distance evaluation using the edge from the mesh. (b) When dealing with conic curves, it is necessary to find the correct conic segment from which the distance will be calculated. In this case, although  $d_1 < d_2$ , the correct distance to be used is  $d_2$ .

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 7). 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 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:

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

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:

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

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

$$\begin{aligned} & (c_1(x_0 - x'_i)^2 + c_2(y_0 - y'_i)^2 + \\ & c_3(x_0 - x'_i)(y_0 - y'_i))t_i^2 + \\ & (2c_1x'_i(x_0 - x'_i) + 2c_2y'_i(y_0 - y'_i) + \\ & c_3(x_0y'_i + x'_iy_0 - 2x'_iy'_i) + \\ & c_4(x_0 - x'_i) + c_5(y_0 - y'_i))t_i + \\ & (c_1x_i'^2 + c_2y_i'^2 + c_3x'_iy'_i + c_4x'_i + c_5y'_i + c_6) = 0 \end{aligned} \quad (10)$$

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

$$\begin{aligned} \mathbf{a}_i &= c_1(x_0 - x'_i)^2 + c_2(y_0 - y'_i)^2 + \\ & c_3(x_0 - x'_i)(y_0 - y'_i) \\ \mathbf{b}_i &= 2c_1x'_i(x_0 - x'_i) + 2c_2y'_i(y_0 - y'_i) + \\ & c_3(x_0y'_i + x'_iy_0 - 2x'_iy'_i) + \\ & c_4(x_0 - x'_i) + c_5(y_0 - y'_i) \\ \mathbf{c}_i &= c_1x_i'^2 + c_2y_i'^2 + c_3x'_iy'_i + c_4x'_i + c_5y'_i + c_6 \end{aligned} \quad (11)$$

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

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

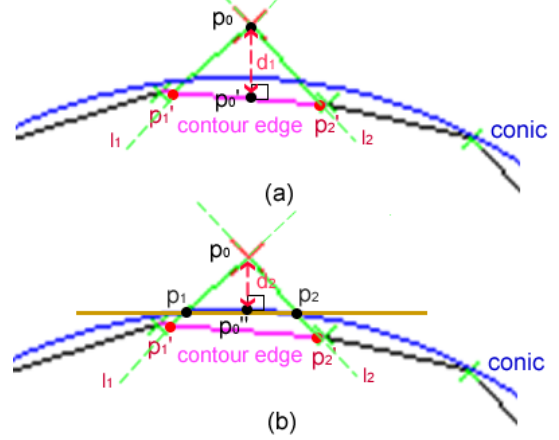


Figure 7. (a) Standard approaches calculate the distance between  $p_0$  and the point  $p'_0$  on the contour edge. (b) In our approach, a new point position  $p'_0$  is found on the conic passing through this patch to calculate the distance.

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:

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

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

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

Figure 8(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 8(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. EXPERIMENTAL RESULTS

Experiments were carried out in two scenarios to verify the performance of the proposed tracking system: with synthetic data and in a real environment using video images. Results are shown comparing our Conics Tracking (CT) method with a standard Line Tracking approach using Sparse (SLT) and Dense (DLT) meshes.

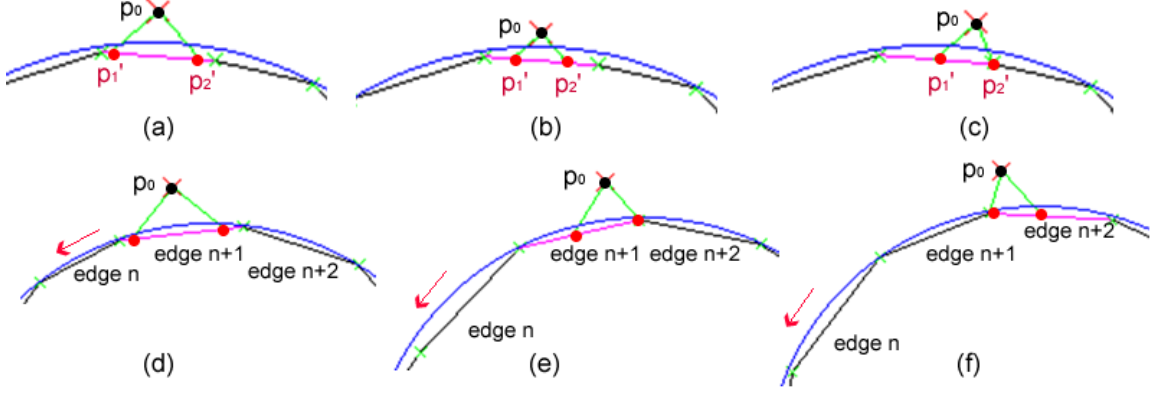


Figure 8. (a) to (c) illustrates different relationships between the detected points and the mesh edge as well as the conic. (d) to (f) shows a slight movement of the object to the left resulting in a new correspondence between detected point and contour edge.

### A. Quantitative results

In this section, quantitative results from a controlled experiment are presented. A simulator similar to the one used in [15] was developed, taking as input three models of the target object in different levels of quality: a very dense mesh, a dense mesh and a sparse mesh. While the actual evaluation compares the tracking using the dense and the sparse mesh, the very dense mesh only creates a synthetic representation of the real object.

The experiment consisted of a series of 10 trials for two different objects (angel and duck) without texture. The number of patches chosen was 10,000 patches for the very dense mesh, 5,000 patches for the dense mesh and 250 patches for the sparse mesh. Each trial had a total of 300 runs and represented the object in a different initial position, manually initialized. In the beginning of each run, the test model was moved back to its initial position and a new pose produced by the simulator using Gaussian noise was used. Evaluation was performed by computing the camera displacement between those two poses at each run.

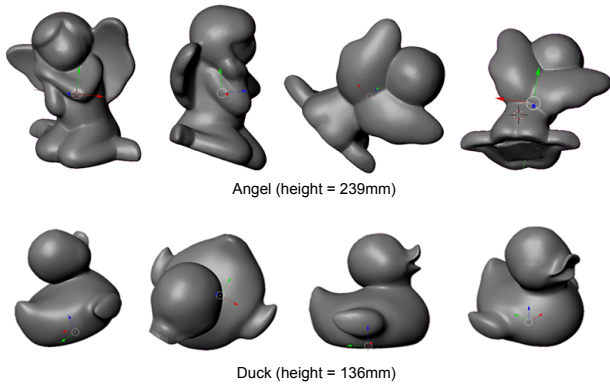


Figure 9. 4 out of 10 poses used in the quantitative evaluation of each object. Each pose was tried 300 times with different noise values. Below each model, the approximate height of the model used in the simulator.

The noise values were calculated with standard deviation of 3.0 degrees and 5.0mm per frame, for each axis in the rotation ( $r_x, r_y, r_z$ ) and translation ( $t_x, t_y, t_z$ ) components, respectively. It was considered a frame rate of 30fps and a hypothetical situation of the user holding the object and moving it about 150mm per second. The distance of the object to the camera was fixed to 350mm and the camera internal parameters were obtained from a Logitech QuickCam Fusion webcam calibrated with OpenCV. Some examples of poses that were used and the approximate size of the models are shown in Figure 9.

Three main conditions were evaluated: *computational time* (average processing time), *accuracy* (mean squared error (MSE) and the standard deviation of the estimated error values for the angle and the distance components of the pose vector) and *robustness* (success rate in 300 runs). For the last condition, if the error returned by the cost function was less than 1.0, than the tracking succeeded; otherwise, it was considered a failure.

The graphs in Figure 10 show the average of the results obtained for the 10 poses. Figure 10(a) shows better results for DLT as expected and although the success rate average of CT was lower than the values obtained in SLT, the difference was very small. On the other hand, CT presented better results than SLT for computational time and accuracy for the translation component (Figure 10(b) and (d) respectively), except for the computational time of the duck object, in which a small difference is also noticed. For the rotation component (Figure 10(c)), CT was less accurate, but with results very close to SLT. Table I shows in more details the accuracy values of the evaluated methods.

### B. Qualitative results

Qualitative evaluation was performed with a handheld monocular camera and to ensure a fair comparison, all approaches used the same video sequences. To visualize the tracking, a 3D model of the target object was rendered on

	Angel			Duck		
	Robustness	Rotation	Translation	Robustness	Rotation	Translation
CT	299	0.6923609	0.9027721	300	1.9483961	1.5428698
SLT	300	1.1852025	2.443657	300	1.869557	2.4045051
DLT	299.9	0.2284348	0.8110891	300	1.3330626	1.4758108

Table I  
SUCCESS RATE AVERAGE AND MSE RESULTS FOR THE ROTATION AND TRANSLATION COMPONENTS.

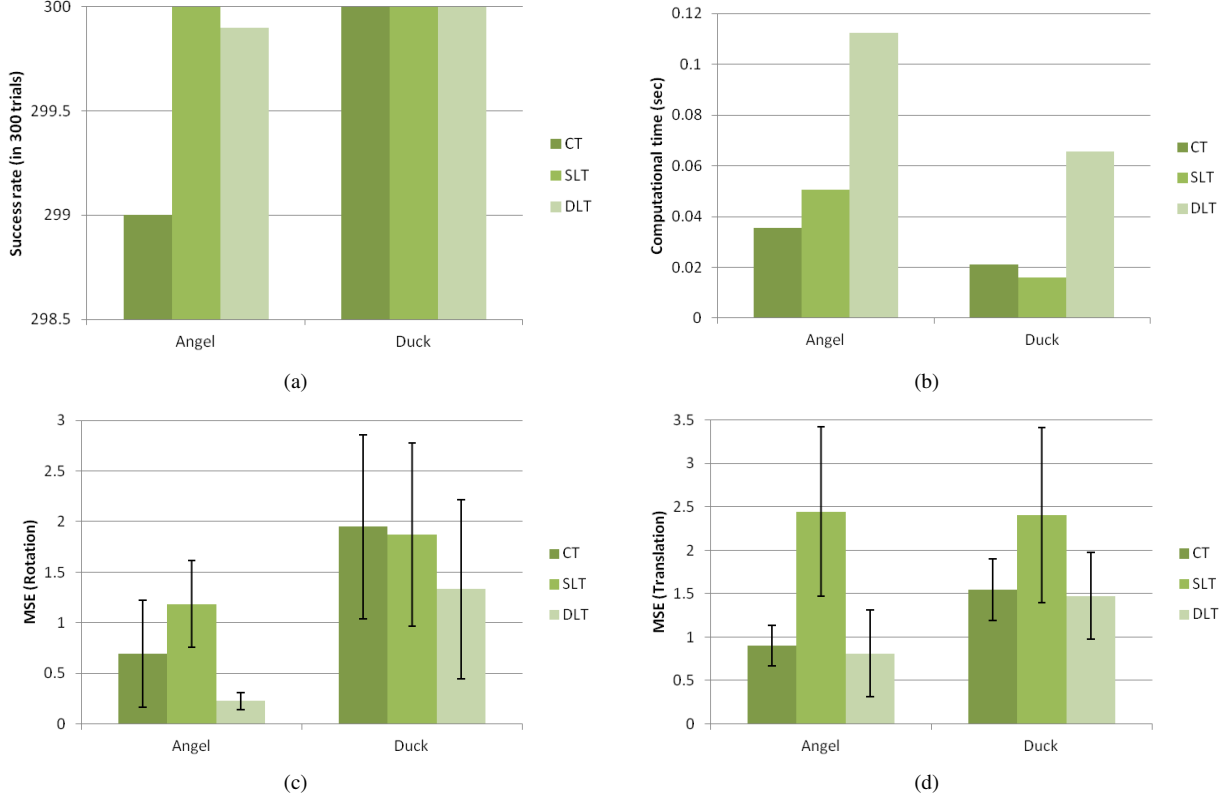


Figure 10. Graphs for (a) success rate, (b) computational time, (c) accuracy in the rotation and (d) translation components.

top of the real object. In the first sequence, the user held the objects in his hand and moved it in different positions and orientations. In the second sequence, instead of moving the object, the camera moves around the object. In both sequences, occlusion by the hands and other objects around the scene were also considered (Figure 11).

### C. Discussion

The current implementation showed good results in both synthetic and real environment, but there are still some issues to be solved. Although the quadrics fitting is good for most of the patches in the mesh, for some poses the obtained fitting is not the most suitable one. In our approach, these patches are not considered during the tracking which may be affecting robustness. In Figure 12, the conics used for tracking are shown with the patches presenting bad fitting highlighted by the red square.

Furthermore, although our framework is able to handle objects with less than 6DOF by using Singular Value Decomposition, it cannot properly handle situations where the object starts with 6DOF but depending on the viewpoint, the DOF changes. In Figure 11, when the duck is rotated, at some point the shape becomes symmetric and 1DOF is lost. The tracking does not fail, but it is possible to see that the model is not correctly rendered on the object. This problem will be solved in the future by incorporating into our framework the approach suggested in [16].

### V. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel method to deal with the trade-off between computational efficiency and tracking accuracy when dealing with non-textured rigid curved objects. Quadrics are suggested to represent each patch of the mesh, being a simple representation, easy to calculate and



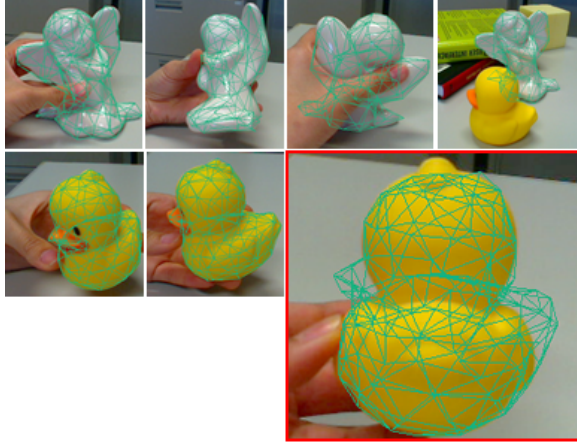


Figure 11. Qualitative evaluation. Highlighted by the red square, a pose incorrectly handled by our framework.

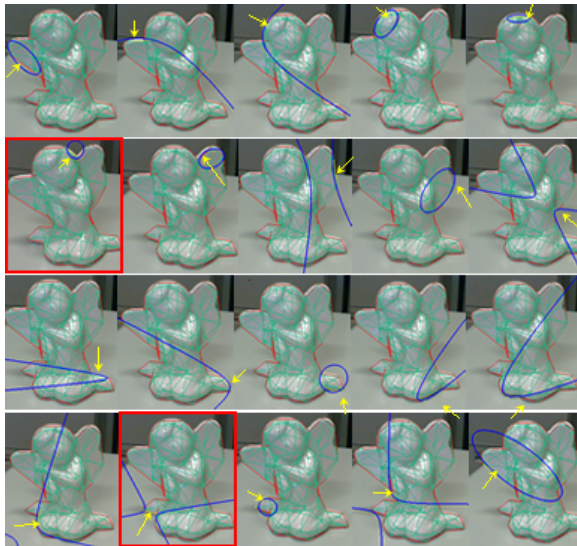


Figure 12. Conics Tracking: conic curves (in blue) on the patches that belong to the object contour. The images highlighted by the red square show patches with a bad conic fitting for the current pose.

efficient when dealing with curved objects having different shapes. A new method for evaluating the distance between projected and detected features was also developed to attend the particularities of using this quadrics patch representation.

Some 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 a critical factor for the tracking performance. Our approach can be also used in rapid prototyping systems, to render textures and check different results in the produced model.

#### 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] 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.
- [5] R. Cipolla and P. Giblin, *Visual Motion of Curves and Surfaces*. Cambridge University Press, 2000.
- [6] 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.
- [7] S. Ma, "Conics-based stereo, motion estimation and pose determination," *International Journal of Computer Vision*, vol. 10, no. 1, pp. 7–25, 1993.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] T. Drummond and R. Cipolla, "Real-time tracking of highly articulated structures in the presence of noisy measurements," in *Proceedings of International Conference on Computer Vision*, 2001, pp. 315–320.
- [12] M. Garland, "Quadric-based polygonal surface simplification," Ph.D. dissertation, Carnegie Mellon University, 1999.
- [13] 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.
- [14] G. Taubin, "Distance approximation for rasterizing implicit curves," *ACM Transactions on Graphics*, vol. 13, no. 1, pp. 3–42, 1994.
- [15] 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.
- [16] K. Kumagai, G. Yamamoto, M. Fujisawa, T. Amano, J. Miyazaki, and H. Kato, "Improvement of the robustness for model-based tracking based on the measurable dof for tracking targets," in *The 16th Annual Conference of the Virtual Reality Society of Japan*, 2011.