NAIST-IS-DD1061027

Doctoral Dissertation

A model-based tracking framework for non-textured 3D rigid curved objects using sparse polygonal meshes

Marina Atsumi Oikawa

February 7, 2013

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Marina Atsumi Oikawa

Thesis Committee:

Professor Hirokazu Kato	(Supervisor)
Professor Naokazu Yokoya	(Co-supervisor)
Associate Professor Jun Miyazaki	(Co-supervisor)
Associate Professor Masayuki Kanbara	

A model-based tracking framework for non-textured 3D rigid curved objects using sparse polygonal meshes*

Marina Atsumi Oikawa

Abstract

Tracking the 3D pose of a known object is a common task in computer vision and approaches aiming to achieve real-time tracking are in constant development to attend different applications and scenarios. Augmented reality, robotic manipulation and gesture recognition are just some examples where accurate and robust tracking in real time is essential for a successful application.

The main issue addressed in this thesis is the problem of determining the pose of non-textured 3D rigid curved objects. A common approach for non-textured objects tracking uses an edge-based method combined with a wireframe model representing the object shape. However, in order to accurately recover the shape of curved objects, high quality meshes are required, creating a trade-off between computational efficiency and tracking accuracy. Previous approaches usually impose some restriction, either in the object shape or motion. To solve this problem, a novel model-based framework for tracking curved objects using sparse polygonal meshes is proposed, with a model representation that can avoid the trade-off previously mentioned and which is able to deal with both simple and complex shapes. It also includes the necessary modifications to efficiently use the apparent contour of curved objects in model-based tracking. Additionally, a method to deal with changes in the observable Degrees of Freedom (DOF) of the target object is also proposed, being able to recover one missing DOF.

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1061027, February 7, 2013.

The effectiveness of the proposed framework is confirmed with quantitative and qualitative experiments, comparing our approach and the traditional method using sparse and dense meshes. Results are presented using both synthetic and real video image data. For further evaluation, an augmented reality application for rapid prototyping was developed and evaluated through a user study, also achieving positive results.

Keywords:

Model-based tracking, non-textured rigid curved objects, sparse polygonal meshes, quadrics, apparent contour, augmented reality.

粗なポリゴンメッシュを用いたテクスチャの無い３次元曲面物体のモデルベーストラッキングフレームワーク*

Marina Atsumi Oikawa

内容梗概

既知の物体の３次元位置姿勢のトラッキングはコンピュータービジョンの分野における一般的な課題であり，異なる応用の要求を満たす実時間トラッキングを成し遂げることを目的とした様々な手法が提案されている．この応用例としては拡張現実感，ロボティクスやジェスチャー認識等が挙げられ，これらに応用することを考えた際，正確かつ頑健な実時間トラッキングであることが重要である．

本論文で扱う主な問題は，テクスチャの無い３次元曲面剛体物体の３次元位置姿勢の推定である．テクスチャの無い物体の一般的なトラッキング手法は，物体の形状を表現するワイヤーフレームモデルと組み合わせたエッジベースの手法である．しかし，曲面物体の輪郭線の見た目は静的なものではなく視点により大きく変化するため，エッジベースのトラッキングを曲面剛体物体のトラッキングに適用することは容易ではない．さらに正確に曲面剛体物体の形状を構築するためには高密度なポリゴンメッシュが必要とされ，計算効率とトラッキング精度がトレードオフの関係にある．これらの問題に対して先行研究では，通常，物体の形状か動きのいずれかにある制限を課すことで解決を図っている．この制限の例として，単純な曲面物体（球体や円柱等）または円運動にしか適用できない等が挙げられる．これらの欠点を考慮して，本研究では，粗なポリゴンメッシュを用いたテクスチャの無い３次元曲面剛体物体のための新たなモデルベーストラッキング法を提案する．この手法の実現のため，３次曲面パッチの表現手法を開発した．これはオフラインにおいて，メッシュ内の各パッチについて一般的な二次多項式を計算し，物体の輪郭の滑らかな局所近似を与える表現である．トラッキング中にはメッシュのエッジを計算に使用するのではなく，この二次曲面の射影を表す曲線を使用する．この表現を用いることで，モデル形状が簡易であっても複雑で

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文, NAIST-IS-DD1061027, 2013年２月７日.

あっても，トラッキング精度を保ちながら，計算効率良く処理することができる．さらに，観測され得る自由度が 6 自由度未満である物体に対し，間違った位置・姿勢推定を行うことを避けるための，対象物体の観測可能な自由を計算する方法についても提案する．この手法は，視点位置により観察可能な自由度の数が変化するような物体に対して，観察不可能だったある 1 自由度の復元を行う場合にも適用可能である．

提案フレームワークの効果は，粗なポリゴンメッシュと密なポリゴンメッシュを用いた従来の方法との比較実験を通して，定量的，定性的に証明を行った．また，実験はシミュレータと実環境の両方により行った．さらなる評価のために開発したラピッドプロトタイピングのための拡張現実感システムを用いた被験者実験においては肯定的な結果が得られた．

キーワード

モデルベーストラッキング、テクスチャの無い３次元曲面物体、粗なポリゴンメッシュ、二次多項式、見かけの輪郭線、拡張現実感.

to the person who inspires me to chase my dreams:
my mother, Cecilia

Contents

List of Figures

List of Tables

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
AR	Augmented Reality
AP	Augmented Prototyping
CAD	Computer Aided Design
CT	Conics Tracking
DLT	Dense Line Tracking
DOF	Degrees of Freedom
HMD	Head-Mounted Display
HSV	Hue, Saturation, Value
QEM	Quadric Error Metrics
RP	Rapid Prototyping
SLT	Sparse Line Tracking
SVD	Singular Value Decomposition
VP	Virtual Prototyping
VR	Virtual Reality

CHAPTER 1

Introduction

1.1. Background and Motivation

Augmented Reality (AR) is gradually becoming part of people's life through different applications. By overlaying virtual imagery with the vision of the real environment, AR adds a new perspective to what users can see and how they interact with the real world.

AR has the potential to be applied in various fields. A popular example is in the entertainment industry, with games such as the *EyePet*™ [1] and *The Eye of Judgment*™ [2] for PlayStation®3 or the AR cards used in Nintendo 3DS™[3]. These games have customized cards that when visualized by a webcam or the game console's camera, triggers an algorithm that can render virtual characters in the display as if they really exist in the real world. Some interactions with these characters are also allowed, which makes the experience more realistic.

Other examples of AR applications include guidance during assembly [4] or maintenance operations [5, 6], with augmented instructions (virtual arrows, text or animated agents) indicating which step should be performed by the user. In this scenario, usually the user wears a HMD (Head-Mounted Display) to visualize the augmented information on top of the real object. Hence, there is no need

to spend time looking for the same information in manuals or other paper based documentations. Following a similar principle, applications have been also developed in education [7, 8], medical visualization [9, 10] and car navigation [11, 12], just to mention some.

With the recent popularization and increase of processing power of smartphones and tablet computers, devices that can be used for AR are becoming smaller, lighter and cheaper. Therefore, their implementation as tools to help users' daily life are also becoming common. For instance, by obtaining information about the users' current location, useful annotations about nearby places can be overlaid in the image captured by the device's camera and visualized in AR browsers such as *Wikitude* [13] or *Junaio* [14].

However, to realize the applications mentioned above, there are several technical challenges. Three important properties are listed by Azuma [15] as essential for AR systems: Integration of real and virtual worlds, interaction in real time, and registration in 3D. The first property is basically what defines AR systems: A seamless integration between real and virtual words, in such a way that users can have the perfect illusion that the two worlds coexist, as illustrated by the examples in Figure 1.1. The other two properties reinforce this illusion of coexistence by providing a real-time response to the user and by correctly positioning the virtual objects with respect to the objects in the real world and to the position of the camera. This correct alignment is crucial for a successful augmentation and is directly related to another important definition: *tracking*.

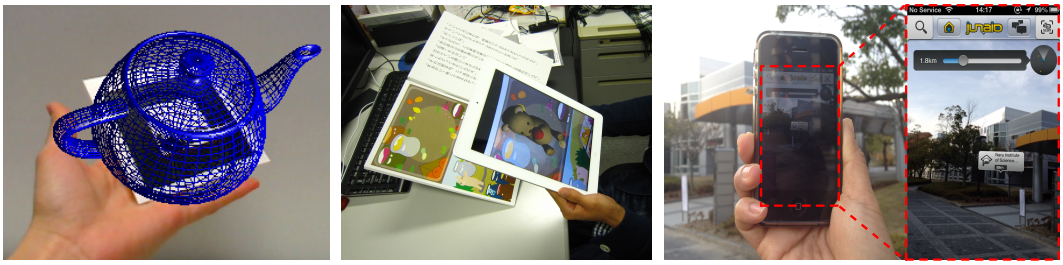


Figure 1.1. Some AR examples: (a) A virtual teapot rendered on top of a fiducial marker. (b) Augmented pop-up book, in which a virtual bear can be seen on top of the real book in the image captured by the iPad's camera. (c) Augmented browser [14] with additional information about places nearby the user.

Tracking the 3D pose of a known object is a common task in computer vision and it aims to continuously recover six Degrees of Freedom (DOF) representing the object position and orientation relative to the camera while it moves around the scene [16]. Different approaches for achieving real-time tracking have been developed, considering the application type, target object shape and features (textured/non-textured), environment (outdoor/indoor), among other conditions. In general, they can be vision-based, sensor-based or hybrid.

In vision-based approaches, no special equipment is necessary, only a camera to capture the image from the real world. This input image is then analyzed to identify possible features that can be used for tracking, which can be intentionally placed in the environment (e.g. fiducial markers such as the ones used by ARToolKit [17]), prepared beforehand based on the target object shape [18, 19] or naturally exist in the scene [20].

Sensor-based approaches use devices such as GPS for obtaining position data and magnetic or inertial sensors for orientation. They are commonly employed in outdoor applications, but they can be expensive, prone to error accumulation and sensitive to perturbations from the environment (e.g. magnetic fields). This led to hybrid approaches, that combine both, vision and sensor-based methods to achieve robustness [21, 22, 23, 24], but increase the complexity of the system.

In this thesis, a novel vision-based approach is proposed for tracking *non-textured rigid curved objects*. In the target scenario, the object shape is known beforehand, it is smooth, and no texture information exists on its surface, as shown in Figure 1.2.

Figure 1.2. Examples of non-textured curved objects. Without texture, the most distinctive feature that can be used for tracking these objects is their outline.

Some examples where curved surfaces tracking can be useful include: Assembling and maintenance operations, where virtual annotations can be linked to each target component to assist the user during the task; industrial applications, such as augmented prototyping (Chapter 6), where virtual textures are used to evaluate the visual appearance of physical prototypes; tracking of human body parts (e.g. hands), which can be used as user interfaces, among others. In these applications, a markerless approach is desirable, since visualization of the augmented information on the target object from different views require placement of multiple markers, which can be difficult if the target object is small or has a complex shape. Furthermore, for augmented prototyping applications, if markers are placed on the object's surface, they become intrusive elements, obstructing the correct blending of the environment information (e.g. illumination) with the object's material properties. As a result, the augmentation does not look realistic.

1.2. Research problem

Approaches for rigid objects tracking have been widely explored, for both marker and markerless environments. A more detailed survey can be found in [16, 25, 26]. However, few works have been dealing specifically with non-textured curved objects - in most of the cases, the problem of tracking these kind of surfaces is only solved for simple shapes, which can be approximated by curved primitives [27] or by simply treating them as polyhedral objects [28].

Since there are no markers or texture information on the object surface, a distinctive feature commonly used is the object's outline or *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 [29]. This apparent contour can be used with model-based approaches that consider the object edges during tracking [18, 19]: A wireframe model of the target object is prepared beforehand and used for matching with the edge information found in the video image. The final pose is then obtained after an optimization process.

However, since the apparent contour of curved objects changes according to the viewpoint, to use it in standard edge based tracking methods, some adjustments are required and in some cases restrictions are imposed on the target object shape or motion, as explored in more details in Chapter 2.

Furthermore, when dealing with curved objects, high quality meshes are required to accurately represent the object's shape. This creates a trade-off between *computational efficiency* and *tracking accuracy*: Reducing the number of patches in the mesh to improve the system efficiency directly affects the object contour representation. One example is illustrated in Figure 1.3, where a polygonal mesh representing the angel figurine is rendered on top of the real object: Using a (b) dense polygonal mesh, it is possible to obtain a good contour approximation, but at the cost of low efficiency due to the amount of patches that needs to be processed. On the other hand, using a sparse polygonal mesh improves efficiency, but the object contour has a coarse representation as highlighted in (c), which affects tracking accuracy since the error between projected and detected edge points increases.

Another problem that needs to be considered during tracking is related to the number of observable DOF of the target object. Consider, for instance, the cup without handle and the mug cup shown in Figure 1.2. In the first case, the object has only five DOF, which will cause problems during the optimization loop: The Jacobian matrix J_{es} relating the distance between projected and detected edges and the pose parameters become rank-deficient and, hence, the pose cannot be correctly calculated. In the second case, when the handle of the mug is visible to

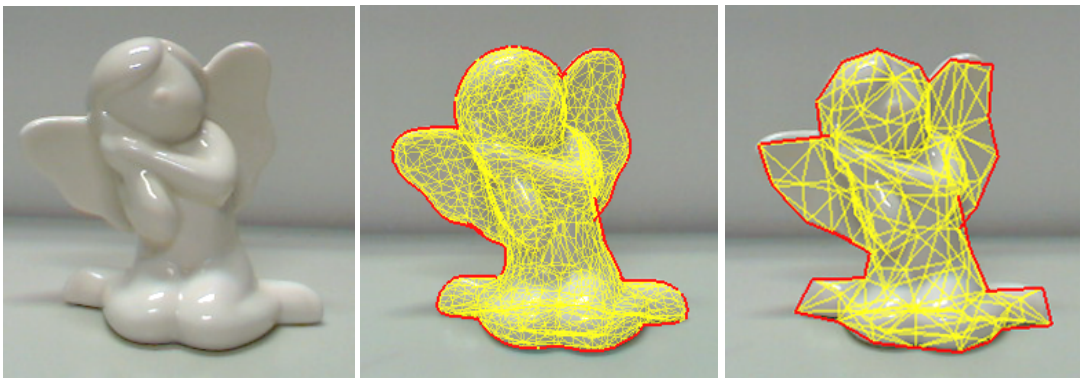


Figure 1.3. Contour representation of dense and sparse polygonal meshes. (a) The target object. (b) A dense polygonal mesh in yellow is overlaid on the object, with a good approximation of the contour highlighted in red. (c) Using a sparse polygonal mesh results in a coarse representation of the object's contour.

the camera, six DOF can be estimated. However, if during tracking the handle becomes completely occluded, a rotation invariance in one of the axis is detected and the mug loses one DOF. When the handle becomes visible again, six pose parameters cannot be correctly recovered (Figure 1.4). This problem may occur even for objects without any kind of ambiguity, such as the teapot. If the camera moves in such a way that only a small part of the teapot is visible (e.g. the sphere on the lid or the spout), an ambiguity is detected and recovery of the object's pose becomes difficult.

Lastly, as previously mentioned, augmented prototyping is one possible application of non-textured rigid curved objects tracking. In this case, evaluation of aesthetic aspects can be easily achieved by enhancing the physical prototype with AR. However, this augmentation is not straightforward, as illustrated in Figure 1.5: A strategy is necessary to render only the part of the texture that is currently visible to the camera. It is also necessary to provide realistic blending of this texture with the object's material properties and the environment illumination. Lastly, correct texture rendering when the user's hand or other objects in the scene occlude part of the target object also need to be handled.

Motivated by these issues, this thesis describes a novel model-based tracking framework that can accurately determine the pose of non-textured 3D rigid curved objects, mainly targeting the following points: A model representation that can avoid the trade-off between tracking accuracy and computational efficiency when using sparse polygonal meshes, and which is able to deal with both simple and

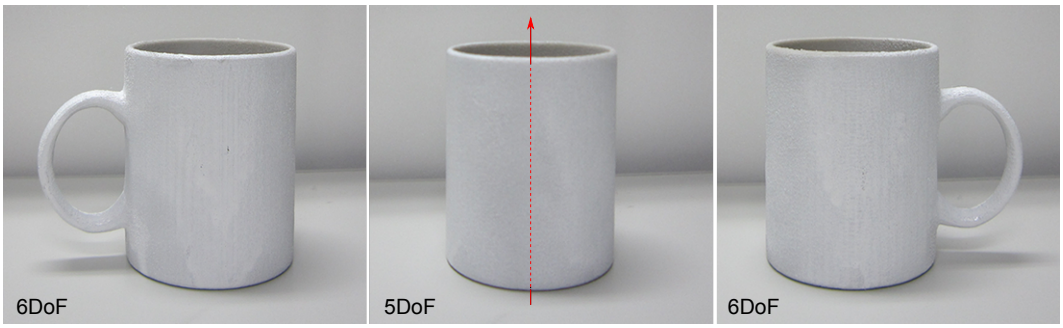


Figure 1.4. Changes in the number of observable DOF of the mug: (a) 6DOF can be estimated. (b) With the handle occluded, only 5DOF can be estimated. (c) If the handle becomes visible again, it is not possible to recover six DOF parameters.

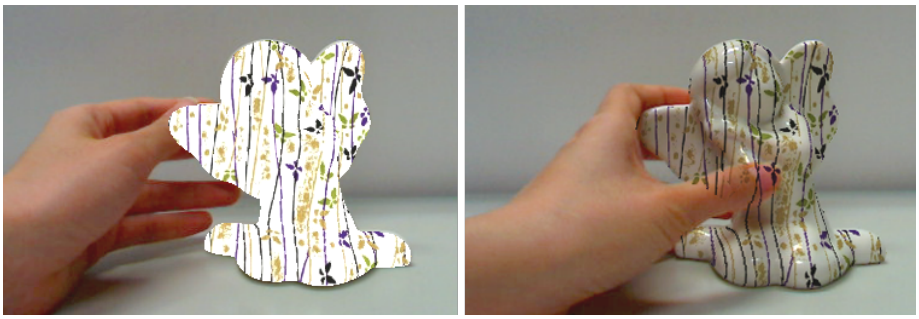


Figure 1.5. Augmented Prototyping application. (a) Unrealistic rendering of the virtual texture. (b) Correct texture rendering, but without handling occlusion.

complex shapes; the necessary modifications to allow the use of the apparent contour in model-based tracking; how to deal with changes in the observable DOF of the target object and the applicability of the proposed framework.

1.3. Research goal and approach

To use sparse polygonal meshes for edge-based tracking of curved objects, a new model representation named *quadrics patch representation* was developed, where a general quadric equation is calculated for each patch in the mesh and used to give local approximations of the object contour. Quadrics are used because they have simple contour generators. Their apparent contour is represented by conic curves and can be obtained by using differential geometry [29]. When dealing with sparse meshes, using conic curves instead of the original edges from the mesh makes the tracking more accurate because conic curves better approximate the object's local shape. Hence, more correct point correspondences can be found.

Unlike standard edge-based tracking systems, our approach does not evaluate the distance between the detected points in the video image to the points assigned on the mesh edges; instead, the distance is evaluated to the projection of the quadrics belonging to the patches located on the object contour for the current view. For instance, in Figure 1.6 (b), instead of using the model edge (in yellow), the curve which approximates the local shape of the object contour (in blue) is used to evaluate the distance between detected and projected edges. The error is clearly smaller when compared to Figure 1.6 (a).

It is important to highlight that several quadrics are calculated for the whole object, each one associated with one patch (or face) of the model. Thus, each patch also has one equivalent 2D curve. This makes it possible to apply the proposed framework to complex object shapes without affecting the computational time during tracking, since this calculation is done during the offline stage.

To handle objects having less than six DOF, Singular Value Decomposition (SVD) [30] in the Jacobian matrix J_{es} is used. Therefore, simple shapes such as the cup without handle on Figure 1.2, or torus-shaped objects and a sphere can also be tracked. Moreover, based on the work developed by Kumagai et al. [31], the proposed framework was extended to allow tracking when the observable DOF of the target object changes during the tracking sequence.

For realistic rendering of the virtual texture, OpenGL is used with a 2-path algorithm to combine the texture's color with the object's color and render the final texture only for the most front surface patches. Texture occlusion is solved by segmenting the prototype's color using HSV (Hue, Saturation, Value). The final result is an approach that can correctly render the virtual texture on the object surface and can be visualized from any viewpoint. Furthermore, it considers the illumination of the environment, with correct rendering even for objects with specular surfaces.

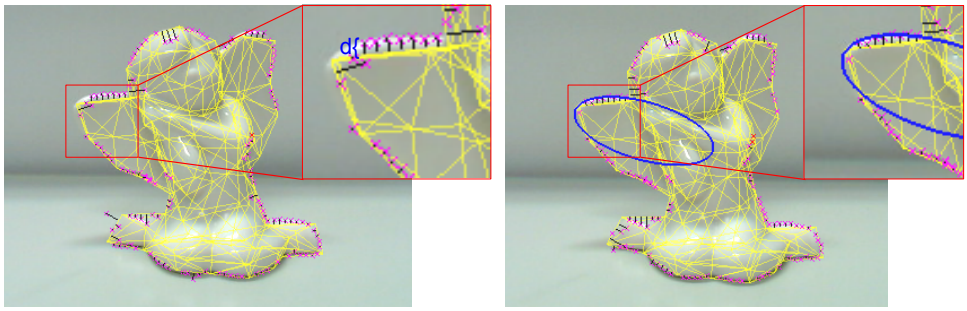


Figure 1.6. Comparison between (a) the standard edge-based tracking and (b) the proposed method: A large distance d between the sample points on the model and the detected edge points is evaluated for the first case. In the second case, the detected edges position is almost coincident with the conic curve (blue line).

1.4. Contributions

The main contributions of this thesis are presented below:

- **A new model representation for tracking rigid curved surfaces using sparse polygonal meshes**.

  By using the proposed *quadrics patch representation*, the level of detail of the polygonal mesh used for tracking can be considerably reduced without affecting accuracy. This is possible because the re-projection error is not calculated using the distance between the points assigned on the mesh edges to the detected points in the video image; instead, curves representing the quadrics projection of the patches located on the object contour are used. A local approach is explored, allowing to apply this representation for simple and complex shapes. A standard model-based tracking framework was modified to handle these changes and validation of the proposed method is presented through quantitative and qualitative experiments.

- **A method for dealing with changes in the measurable DOF of the target object**.

  In general, many rigid objects found in the real world have six DOF. However, simple shapes may have one or more unobservable DOF or depending on the viewpoint, an object can present some kind of ambiguity that leads to a wrong pose parameters estimation. Thus, in the proposed tracking framework, a method to calculate the number of observable DOF of the target object was included. When necessary, a recovery process is triggered, which is able to recover one missing DOF.

- **An Augmented Prototyping application**.

  By using the proposed tracking framework, an augmented prototyping application was developed. With AR, evaluation of the design and aesthetic features of the physical prototypes in real-time became easier, saving time and production costs. Compared to previous approaches using AR to assist design evaluation, the proposed method has an additional advantage of considering the environment illumination effects on the virtual textures. Results from a pilot user study comparing the use of a 3D modeling software and the proposed application are also presented.

1.5. Software

The software developed in this thesis was implemented and evaluated under the *Windows* Operating System (but not being restricted to it) using the C language. Additionally, the following libraries and software were used:

- **ARToolKit** - http://www.hitl.washington.edu/artoolkit/ - a library to develop AR applications using square markers. It provides several functionalities, including methods for video input/output and matrix calculation that facilitated the implementation of our framework.
- **Blender** - http://www.blender.org/ - a 3D modeling software used to edit the 3D polygonal meshes used in the experiments. It was also used to create the virtual textures and in the evaluation of the augmented prototyping application.
- **Metro** - http://vcg.iei.pi.cnr.it - a tool designed to calculate the difference between two polygonal meshes. It was used to evaluate the polygonal simplification method to be used in our approach.
- **OpenCV** - http://opencv.org/ - a library that provides methods for image processing and computer vision. It was mainly used for the camera calibration and to facilitate the creation of the videos used in the qualitative experiments.
- **OpenGL** - http://www.opengl.org/ - a library used for rendering 2D and 3D graphics, being also employed in the edges visibility test.
- **OpenVRML** - http://www.openvrml.org/ - a library used to help handling the VRML files containing the virtual textures rendered on the physical prototypes.
- **Pthreads** - http://sourceware.org/pthreads-win32/ - a library used to implement multithreading in our framework.
- **QSlim** - http://mgarland.org/software/qslim.html - a software used for simplifying the polygonal meshes.

1.6. Outline of the thesis

This thesis consists of seven chapters. Chapter 2 provides an overview on rigid objects tracking and a description of how the problem of tracking curved surfaces have been tackled by previous approaches. It also introduces some notations and definitions used throughout the thesis.

Chapter 3 describes the necessary steps for constructing the proposed quadrics representation, with a detailed description of the tracking framework developed in this thesis.

Chapter 4 focuses on improvements in the proposed framework, with a method to deal with objects having variable observable DOF and to recover one missing DOF when necessary.

Experimental results are presented in Chapter 5 using both, synthetic data and video images from the real world, with comparative results between standard approaches and the proposed approach.

In Chapter 6, an application of the proposed framework is developed for augmented prototyping. A user study was conducted to confirm its applicability in a realistic scenario and results are discussed.

Chapter 7 summarizes this thesis, with discussions about the contributions, limitations and possible future works. Subsequently, the appendices describe the video files that were created during evaluation of the proposed framework and the mathematical tools used for its implementation.

CHAPTER 2

Background and Related Work

In the first part of this chapter, an overview of rigid objects tracking and related research areas is provided. Subsequently, curved objects tracking is explored, focusing on how this problem has been previously tackled and how it relates to the proposed approach. Some mathematical background and definitions that appears throughout the thesis are also presented.

2.1. Rigid objects tracking

Rigid objects are defined as objects with no deformations or articulations, whose pose is constrained to a small number of parameters. In this research, only rigid objects are considered, aiming at 2D-3D pose estimation. Hence, 6DOF parameters defining the camera position and orientation need to be estimated.

For 6DOF estimation, there are several approaches suggested in the literature. A popular and easy one consists of placing fiducial markers in the target object or in the environment. These fiducial markers are designed with some kind of pattern to allow their identification. Once they are correctly identified, a verification step is triggered to find out which marker or markers are visible to the camera and what is the virtual information related to it that should be presented to the user.

Early works on marker-based tracking were developed by Rekimoto [32], using 2D matrix codes, and Kato & Billinghurst [17], using square planar markers. Applications targeting handheld devices were also successfully implemented [33, 34], with both robustness and computational efficiency. However, as the potentiality of applications in this area started to increase as well as the improvements of computational processing power, so did the demand from different types of users and scenarios. Thus, markerless strategies started to be developed, considering features that could be naturally found in the target environment (e.g. edges, lines, or planar areas). Although some approaches using natural features still require markers only for initialization [35, 36, 37, 38], there are many robust systems that completely rely on features found in the environment, with or without prior knowledge about it, as presented in more details in Section 2.1.2.

2.1.1 Fiducial marker tracking

Fiducial markers have been widely used in AR applications and it is a common starting point for many researchers and enthusiasts of this area. They provide visual cues that can be easily detected, requiring low-cost equipments to setup small AR applications.

Many types of markers have been developed along the years, with different shapes and properties, as exemplified in Figure 2.1. ARToolKit [17] introduced square markers with black borders, used for detection. Each marker has a pattern inside it, used to distinguish it from other markers. They are trained beforehand and when visible to the camera, a corresponding virtual object appears attached to them, as previously presented in Figure 1.1(a).

Figure 2.1. Different types of fiducial markers: (a) ARToolKit [17], (b) CyberCode [39], (c) Intersense [22], (d) ARTag [40], (e) reacTIVision [41], (f) other type of circular marker [42].

The CyberCode was developed by Rekimoto [39] based on a 2D-barcode technology. A guide bar and four corners are searched on the image to allow the system to decode the bitmap pattern in the tag. Similar to CyberCode, Fiala [40] developed a type of square border markers that uses a matrix-pattern to encode an ID, having a 6x6 interior grid of cells representing the logical symbols 1 or 0. Each cell carries one bit of digital data and a 36-bits word can be extracted from one marker after its boundary is determined. They were developed to reduce false markers detection and identification.

Circular markers were initially explored by Naimark & Foxlin [22] aiming at wide-area tracking: with the proposed 2D barcode fiducial design, a high number of different codes could be generated. More recently, Koehler et al. [42] have been also exploring circular markers, but to improve robustness against occlusion, since the camera pose can be calculated from the whole marker contour, instead of only the four corners, as usually done by square markers.

Although many proposed markers have square or circular shapes, another way to identify them consists of using topological pattern matching. In this case, each marker has a unique topology that can be matched against a dictionary of subtrees represented as strings, as implemented in the reacTIVision system [41]. Some problems include difficulty to handle occlusions and the topology needs to be complex enough to avoid false positive detections, but they are a popular approach for tangible user interfaces [43, 44, 45].

Figure 2.2. (a) An augmented storytelling book using ARToolKit square markers: The pattern (b) can be designed to have close relationship to the virtual content displayed to the user on the (c) augmented image.

These researches are just some examples of how fiducial markers represent an important element in the history of AR development. They have been extensively explored in many applications and paved the path to bring the technology out of research labs to the common users. Many types of markers exist and a comparative study in terms of usability, efficiency, accuracy, and reliability was made by Zhang et al. [46]. Details about detection and identification methods developed for markers is also discussed in Koehler et al. [44].

In general, marker-based tracking is a low-cost and robust solution for real-time tracking, specially for indoor applications. In some systems, the marker can naturally become part of the target scenario, as shown in Figure 2.2, in which the design of the pattern is directly related to the contents of the book. However, for some applications, they may be intrusive and the offline preparation required can be critical for large-scale environments: Registering hundreds of markers manually is an arduous task. Hence, new approaches started to be developed relying on features that naturally exist in the target scene.

2.1.2 Natural features tracking

In visual markerless tracking systems, no fiducial markers are placed in the environment and another type of prior knowledge may be available to assist tracking. Common representations include a CAD model of the target object, a set of feature points extracted from the target object (Figure 2.3) or some dominant geometric structure present in the environment, e.g. planar areas.

More recently, *tracking-while-mapping* approaches also started to be explored. These approaches can work without any model of the environment and the camera pose is estimated by using information that is obtained once the environment starts to be explored. A brief description of approaches using both strategies is described in the next sections.

2.1.2.1 Model-based tracking

In model-based tracking, some prior knowledge of the environment is available for tracking. Features on these models are compared to features found in the scene and used to estimate the camera pose. Approaches in this category in general use *edges, optical flow, template-matching*, or *interest-points* [16].

In edge-based methods, a CAD model of the target object is usually used for matching with edges found in the video image. Given an initial estimation of the pose, the model is projected in the image, with control points sampled along the model edges. Search along the normal of each control point is performed to find the highest gradient. The camera pose is then obtained by minimizing the displacement between projected and corresponding detected edges.

These approaches are commonly used for non-textured objects, but since pose in the next frame is calculated based on the previous frame, it does not handle well high inter-frame motion. False matches also occur when the object is highly textured or it is inserted in a scenario with cluttered background, due to matching with wrong edges (more details about this method is given in Section 2.1.3).

Optical-flow based methods explore temporal information extracted from the relative movement of the object projection onto the image in order to track it [47]. Some existent works target mainly face [48] or head tracking [49, 50].

For template-matching methods, as the name implies, features are tracked by comparing the content of each image with a sample template and minimizing the difference between them. This can be useful for tracking complex patterns that are difficult to be modeled using local features. Hager & Belhumeur [51] implemented it for object tracking under planar affine motions and 3D template-matching has been proposed by Jurie & Dhome [52, 53]. Drawbacks of these methods include sensitiveness to partial occlusions and illumination changes.

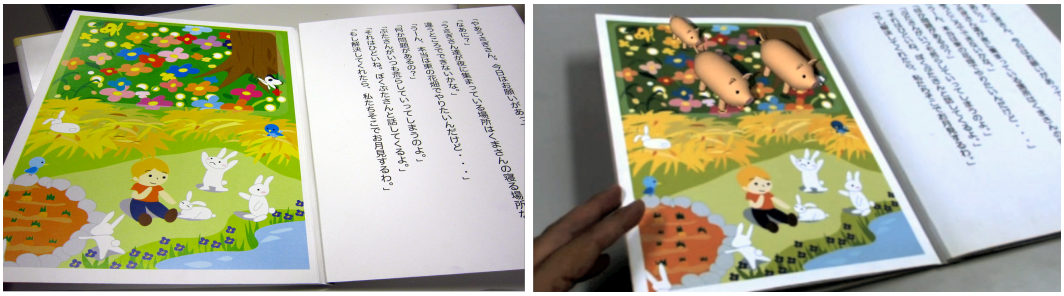


Figure 2.3. An augmented storytelling book using natural features tracking: (a) Features extracted from the image on the left are used for rendering (b) virtual pigs on the book's page.

Different from template-matching methods, which uses global region tracking, interest-points rely on local features. First, a feature detection algorithm is performed (e.g. Harris [54], SUSAN [55] or FAST [56]) to find interest-points in the images, such as corners or blobs. Then, matching with the trained database can be done by using similarity measures such as Normalized Cross-Correlation (NCC) or Sum of Squared Difference (SSD) on image patches.

Other approaches have implemented their own detector and descriptor, creating *feature descriptors* - a vector containing detailed appearance information of distinctive parts of the target image and its neighborhood. The main goal is to achieve robustness against different factors such as noise, detection error, scale and rotation invariance, or photometric distortions. For the matching, distance between vectors, such as Euclidean distance, can be used. Examples of such descriptors include SIFT [57] and SURF [58, 59]. Some advantages of these methods include robustness to cluttered backgrounds, outliers and partial occlusions, but they can be computationally expensive, sensitive to motion blur and the objects need to be textured, with no repetitive patterns.

2.1.2.2 Combination of approaches

To achieve robustness and completeness, combination between the approaches mentioned in the previous section have been also suggested. For instance, since optical-flow based methods are sensitive to error accumulation, Haag & Nagel's [60] approach combines edge elements with optical flow to track moving vehicles in a video scene.

Contour matching with optical-flow is implemented by Brox et al. [61] to achieve a general tracking method and overcome the shortcomings presented by each of the approaches when they are used separately.

A popular and successful combination uses edges and feature points, as an attempt to compensate the weakness of each method by combining their strength. To enable tracking of texture and untextured objects in the same framework, Vacchetti et al. [62] uses multiple hypotheses to handle erroneous edge correspondence, reducing drift as well as sensitiveness to cluttered backgrounds.

Vacchetti et al. [63] also introduced the idea of using *keyframes*, which are reference images created during an offline stage, representing the scene from many different viewpoints (or only one if the scene is not complex). A 3D model of

the target object is required and the tracker starts with a small user-supplied set of keyframes and, if necessary, new keyframes are introduced automatically. During tracking, online information (calculated from previous frames) and offline information (from the keyframes) are combined to prevent drift and jittering.

In Rosten & Drummond's [64] approach, the FAST feature detector is used to extract feature points, which are used to initialize an edge tracker. Feature points are back-projected on a CAD model and correspondence between this point cloud and detected features are established by minimizing the SSD between them. An on-line learning of a dynamic function that maps the SSD to the inlier probability is also used to improve feature tracking.

Using a similar idea, Choi & Christensen [65], create keyframes in the offline stage using keypoint features, which are used for comparison with the input image. Hence, instead of using keypoints to estimate motion between frames similar to the aforementioned approaches [62, 64], they are only used to create an initial estimation for the edge-based tracker. For the pose estimation, an edge-based tracking system is used with iterative re-weighted least-squares.

2.1.2.3 Real-time Structure-From-Motion tracking

In the previous section, camera pose estimation is achieved by relying on some prior knowledge about the target object or the scene. Although more complex, it is also possible to obtain these parameters in completely unknown environments by retrieving information from them in real-time.

In computer vision, Structure-From-Motion (SFM) aims to estimate both 3D geometry (structure) and camera pose (motion) using a sparse set of correspondences between image features [66]. Often implemented offline with bundle adjustment, it is a method commonly used for automatic reconstruction of 3D objects and scenes from video sequences and collections of images [67, 68].

Based on SFM, approaches started to be developed aiming for 3D real-time camera tracking in complex and unknown environments. Koch et al. [69] suggested an optimized SFM using robust statistics, but with tracking being facilitated by the use of fish-eye lens (to capture a large field of view of the scene) and 3DOF inertial rotation sensors.

Targeting the same goal, real-time tracking in unconstrained environments, Davison [70] developed an approach using a single moving camera and Simultaneous Localization and Mapping (SLAM), a concept usually applied for robotic exploration. In SLAM, both the pose and the environment structure is acquired on-the-fly, generating a sparse map of the environment. This approach resulted in the *MonoSLAM* [71], that basically creates a probabilistic sparse map of natural landmarks. Instead of performing image processing and feature matching, landmarks are searched in image regions constrained by estimate uncertainty. However, for real-time operations, the number of landmarks is limited and the approach is only applicable in small rooms. To achieve frame-rate operation with many landmarks, Eade & Drummond [72] developed a monocular SLAM using a FastSLAM-style particle filter [73].

In the AR context, a well known approach using SLAM was developed by Klein & Murray [20], where the main difference, compared to previous methods, was in the implementation of the mapping and tracking as two different processes. In their approach, named Parallel Tracking and Mapping (PTAM), tracking is not linked anymore to mapping, bringing several advantages: the tracking method can be chosen disregarding the mapping approach; and for the mapping, analysis in every single frame is not needed, just in a small number of keyframes.

In PTAM, the 3D point map is initialized using a stereo image of a small tracking area and as the camera moves away from the initial pose, new keyframes and map features are added to the system. Feature tracking is performed by using FAST corners with image patches and the position of the existent 3D feature points is improved using a global optimization approach. This approach proved to be robust enough for tracking a hand-held camera in small AR workspaces, being also implemented later for mobile phones [74].

To deal with non-textured objects in this SLAM context, Mohamed et al. [75] developed a hybrid approach using a 3D model of one static object in the scene, whose edges are used to improve de localization of keyframe-based SLAM. The segments of the model are used to constrain the camera trajectory by adding their reprojection-errors in the bundle adjustment. As a result, a compound cost function that includes the edge information provided by the model and the multi-view relationships from the unknown parts of the environment is constructed, allowing tracking of a non-textured object in a cluttered environment. Although

effective, the elements in the scene need to remain static.

Instead of generating and tracking a sparse map of features from the environment, Newcombe et al. [76] proposed a Dense Tracking and Mapping (DTAM) algorithm, which uses a 3D dense surface model composed of depth maps for dense monocular camera tracking. With the availability of cheap depth sensors such as Kinect, Newcombe et al. have also proposed KinectFusion [77]. A truncated signed distance function is used to constantly update the surface representation and accurate camera pose tracking is obtained by aligning all depth points with the complete scene model. Tracking and mapping runs in parallel using GPGPU. However, similar to PTAM, it only considers static scenes.

Dealing with dynamic scenes makes the problem more challenging, though an attempt has been implemented by Bleser et al. [78]. In this case, a CAD model of one object in the scene is used for initialization, from which line models are created from a given pose and registered on the image to find the initial pose. During tracking, the model is discarded and the scene is reconstructed using linear least squares triangulation, being improved recursively over time using Extended Kalman Filter (EKF).

Sections 2.1.2.1 and 2.1.2.3 summarize some popular tracking approaches, considering some or none a priori knowledge from the environment. For more details and other examples, surveys can be found in [16, 79, 25, 47]. Since this research targets non-textured objects, the most suitable approach to be used is the edge-based tracking, which is explored in more details below.

2.1.3 Edge-based methods in details

Edges are commonly used for being features easy to extract and relatively stable under different types of transformations. In this category, there are two types of implementations [16]: (a) approaches that first extract image contours and later fit them to the model outlines [80, 81, 82] or (b) approaches that first render the object model using an initial estimation of the pose and then look for strong gradients around it for matching [54, 83, 18, 19]. In this section, focus is given on the second case, since it is the method used in our proposed framework.

One of the first approaches for real-time tracking using edges was the RAPID (Real-time Attitude and Position Determination) system [84]. In this method, the pose from the previous frame and a Kalman filter estimator [[85] are combined to predict the new position of the object in the current frame.

Control points are placed along the edges of the 3D model and through a one-dimensional search, they are used for finding and measuring the distance to high contrast image edges that matches them. The object's pose is then updated by minimizing the sum of squared edges normal distances (Figure 2.4).

Other approaches were developed later to improve robustness. Armstrong & Zisserman [83] reduced the influence of outliers by using RANSAC [86]. Marchand et al. [87] implemented tracking based on the estimation of a 2D global affine transformation between two successive images, with the object pose being formulated as an energy minimization process.

In Drummond & Cipolla's [18] approach, the rigid body motion is represented using Lie Algebra formalism and robustness is achieved by using an iterative re-weighted least-square. A robust tracking framework is presented, which was further extended to allow the use of multiple cameras and tracking of complex structures, with articulated components or multiple constraints.

A non-linear pose estimation based on a virtual visual servoing approach is proposed by Comport et al. [19]. Different primitives can be handled by this framework, including straight lines, circles, cylinders, and spheres. A local moving edges tracker is used in order to provide real-time tracking of points normal to the object contours and robustness is obtained by using M-estimators.

Due to its simplicity and since no texture information are available for the objects targeted in this work, our framework is also edge-based. However, most of the approaches developed so far targets polyhedral objects, having flat faces or only targeting simple curved shapes. In the next section, a further analysis on rigid curved objects tracking is given with some of their important definitions and properties.

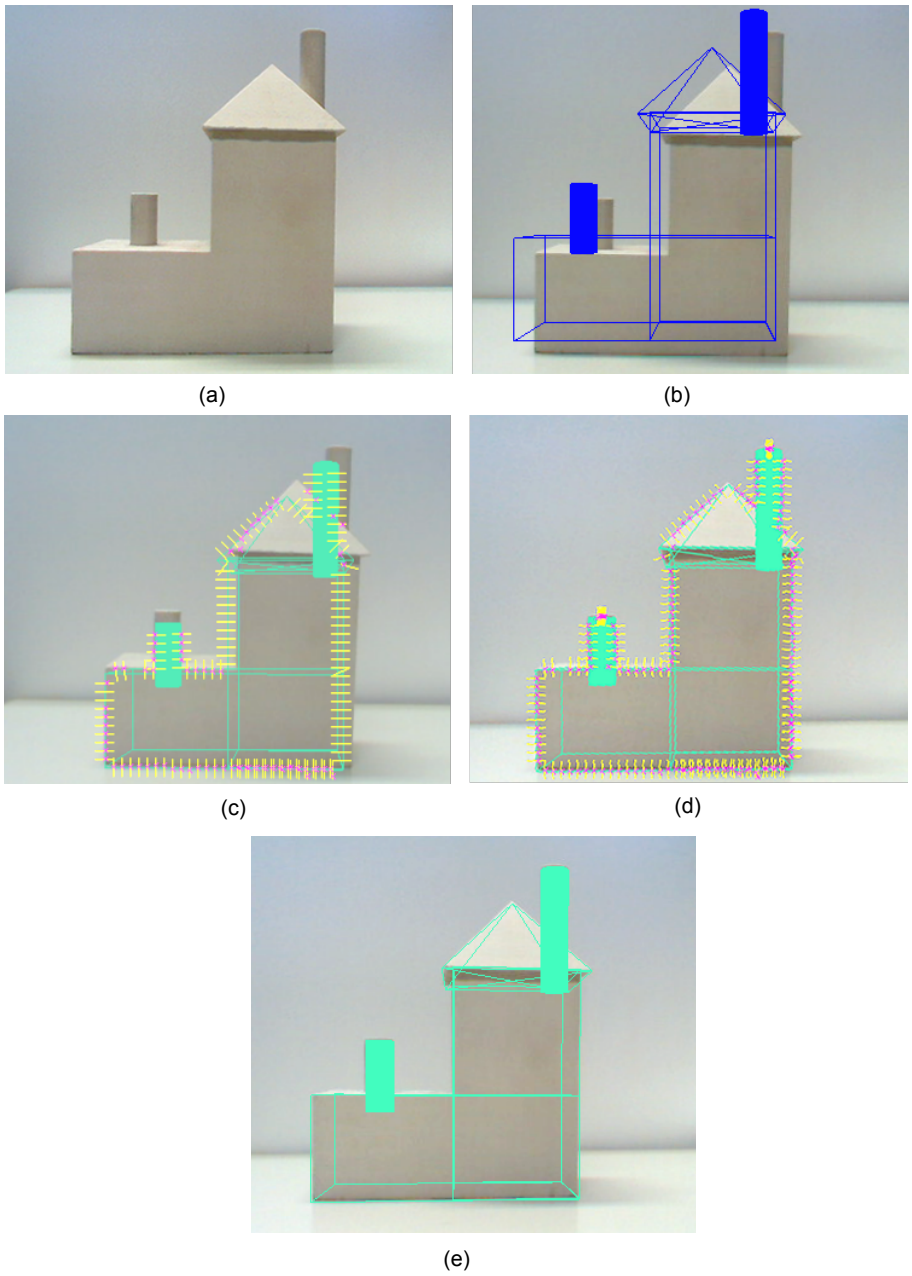


Figure 2.4. Overview of an edge-based tracking system: (a) The target object; (b) the model is rendered on the video image; (c) control points are placed along the model edges and used to find nearby edge points through normal search; (d) model is aligned to the object; (e) the pose is updated.

2.2. Rigid curved objects tracking

Dealing with curved objects using edge-based tracking systems is not straightforward, mainly because the edges are not static; instead, they change according to the viewpoint. This feature is called *apparent contour* and represents the object's silhouette, being considered the dominant image feature when few or no texture is available on the object's surface.

2.2.1 Contour generator and apparent contour

Considering perspective projection, the *contour generator* Γ of a curved surface S is defined as the set of points $X \in S$ forming a curve in 3D-space for which the rays from the camera center to S are perpendicular to the points on the surface normal $\mathbf{N}$. The *apparent contour* γ consists of all points on the image plane forming the projection of Γ [29], as illustrated in Figure 2.5. It is an important feature for curved surfaces, whose deformation contains enough geometrical information to, for instance, recover the shape of curved objects assuming known motion [88].

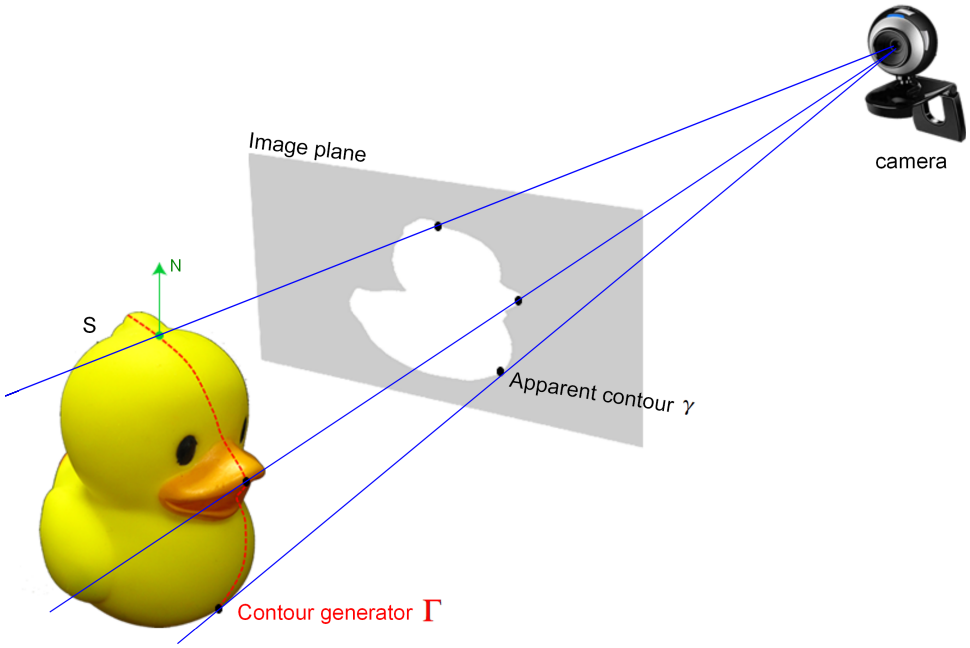


Figure 2.5. Contour generator and apparent contour of a curved surface

2.2.2 Previous work on curved objects tracking

Some attempts aiming to estimate the structure and motion of apparent contours include the use of epipolar parameterization [29]; monocular camera but constrained to orthographic, weak-perspective and affine projection [89]; conic-stereo vision [90] or trinocular stereo [91]. In some cases, these approaches are appealing because a model of the target object is not required. However, they all have some restriction, such as known viewer motion or requiring multiple cameras.

On the other hand, for tracking curved objects using the apparent contour inside a framework similar to the one described in Figure 2.4, some adjustments are required or in some cases restrictions (e.g. shape or motion) are imposed on the target object. In Rosten & Drummond's [92] approach, 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.

A unified approach that can handle fixed and apparent contour edges within the same framework is proposed by Li et al. [93]. A prediction model is formulated based on the local differential geometric analysis of surfaces using quadratic approximation. Then, in contrast of the approach developed by Rosten & Drummond [92], the apparent contour is only updated when the prediction error gets too large, instead of recomputing it for every frame. However, tracking of complex objects are limited to a certain range of motions and it works better for surfaces of revolution. Similarly, in Comport et al.'s [19] approach, although curved primitives (e.g. as circle, cylinders and spheres) are included as features to be tracked, along with straight lines, experiments also targeted only simple shapes.

A combination of contour analysis and optical flow is explored by Brox et al. [61]. Given an initial pose, the segmentation and contour based pose estimation are iterated to successively improve the extracted contour and pose. The optical flow helps to improve the initial pose and to compute additional point correspondences in successive images. However, two requirements are necessary for this approach: the silhouette have to contain enough details to provide a unique pose estimation and the motion of the object cannot be too large.

Azad et al. [94] propose a general framework that requires only an accurate 3D polygon model of the object shape that can be rendered by a graphics card. No assumption regarding the object shape, texture or color is made. A particle filter based tracking is used, where an appearance-based matching is performed using online rendering of a fine-grained local view space using the 3D model. Since a global fine-grained would consume a large amount of memory and computational time, only the views that are close to the current pose estimation are rendered. However, the main drawback of this approach is the high computational cost due to the slow read-back from the frame buffer, which remained even after implementation of a full optimized version for the CPU and GPU.

Kyrki & Kragic [95] propose integration of model-based and model-free cues with robust estimators to track not only polyhedral objects. The model-based tracking module is based on Drummond and Cipolla's [18] approach, including the representation using Lie algebra. For surfaces of revolution, since one DOF cannot be estimated (on the axis of revolution), Lie generators are aligned with the coordinate system attached to the object, removing the effect of the rotation axis. The same strategy is applied for a sphere, which has three unobservable DOF. In the model-free part, interest points are extracted using the Harris corner detector [54] and point tracking is based on minimizing the SSD in the RGB values. Finally, the measurements of these two types of features are combined using an Iterated Extended Kalman Filter (IEKF). However, similar to previous approaches, although curved shapes can be tracked by this framework, they are limited to spheres and surfaces of revolution (cylinder and cone).

2.2.3 Curved objects representation

In model-based tracking, a first step consists of determine the model representation to be used to describe the target object. For curved objects, common representations include collections of *curved primitives*, *polygonal meshes* and *implicit surfaces*.

Curved primitives are simple representations and very appealing because their formulations can be computed efficiently and in real-time. For instance, a representation based on intersections of pairs of quadrics (truncated quadrics) is used by Drummond & Cipolla [27] for tracking highly articulated structures.

A similar approach is implemented by Stenger et al. [96] for tracking articulated hands, in which each part of the hand is modeled by a truncated quadric: the palm is a truncated cylinder, with the top and bottom being half-ellipsoids; segments of a cone represent the phalanx; hemispheres represent the joints and the tips of fingers and thumb. Lastly, ellipsoids, truncated cylinders and truncated cones represent the phalanges of the thumb. This representation has the advantage of being simple and efficient; however, their use is limited to a small class of shapes - the ones resembling these kind of shapes, as exemplified in the description of the hand model.

Polygonal meshes representing the target object shape are another representation, which can be easily constructed using a 3D modeling software or obtained by using a 3D laser scanner or similar technology, when dealing with complex shapes. Once the mesh is obtained, the object can be treated as a polyhedral model [28] or tracking is supported by other visual cues [94, 95]. However, the major problem with this representation is the trade-off between computational efficiency and accuracy when dealing with complex shapes: dense meshes are required to accurately approximate the object shape. Hence, as mentioned in the previous section, most of the approaches only handle simple shapes.

Implicit surfaces are a more general and efficient representation used for curved surfaces. Primitive implicit shapes such as Metaballs [97] can be smoothly combined by summing their functions. For instance, Plaenkers and Fua [98] use metaballs for tracking people using stereo vision. Metaballs are attached to an articulated skeleton and arranged in an anatomically-based approximation and used to simulate the behavior of bone, muscle and fat tissue.

Rosten & Drummond [92] use a sum of Gaussians in $\mathbb{R}^3$ to describe the curved shape. Although it is a general formulation that can define any contour, considering that a suitable starting point is found, the disadvantage is that it is a complex model to be implemented. Nevertheless, tracking of complex objects increases the computational time due to the number of evaluations of the implicit function and its derivatives at each point.

Khan [99] suggested a silhouette-based 2D-3D pose estimation using implicit algebraic surfaces. A explicit 3D mesh is used, which is converted to an algebraic surface. Finally, the pose is estimated from algebraic image silhouette equations. However, the computational efficiency is also limited by the objects complexity:

The degree of projected polynomials grows too large very quickly, making difficult to use it for complex objects tracking in real-time applications.

A comparison among the representations described in this section is summarized in Table 2.1. From the previous works analysis, it is possible to notice the need of a method that can make use of general and accurate models, easy to be constructed and at the same time able to efficiently run in real-time. To fulfill these requirements, in our approach, a new representation is proposed by combining the polygonal mesh and the implicit surface representation.

An implicit algebraic equation, represented by a general quadric, is used to represent the apparent contour. To make the calculation of this equation easier, a polygonal mesh representing the object shape is used: it provides the necessary data for the quadrics fitting, calculated offline. Thus, shortcomings presented by each of the representations can be overcome: a sparse polygonal mesh can be used to achieve computational efficiency, while accurate tracking is kept through the apparent contour of quadrics.

Quadrics have simple contour generators and the calculation of their apparent contour (conic curves) is well defined using differential geometry [29]. Using the conic curves instead of the original edges from the mesh makes the tracking more accurate because they approximate better the object local shape and, therefore, more correct point correspondences can be found. Since this calculation is performed during the offline stage, it does not affect the computational time in the optimization step. Moreover, since this is a local approximation, simple and complex shapes can be easily handled using the same framework.

Table 2.1. Comparison of common model representations of curved surfaces

	Pros	**Cons**
Curved Primitives	Simple	Limited to a small class of shapes
Polygonal Mesh	Simple	Trade-off (efficiency vs. accuracy)
Implicit Surfaces	General	Difficult to construct the models

2.3. Mathematical background

This section presents some mathematical notations used throughout this thesis.

2.3.1 Coordinate frames

Coordinates in 3D space are represented using homogeneous coordinates in the form of $\mathbf{X}_w = (x_w, y_w, z_w, 1)^T$, where $\mathbf{X}$ represents a 4-vector and the subscript w means the point is in the *world* coordinate frame. Similarly, $\mathbf{X}_c = (x_c, y_c, z_c, 1)^T$ represents the point in the *camera* coordinate frame. Corresponding 2D points projected in the image plane are represented as $\mathbf{x} = (x, y, 1)^T$.

The six parameters representing the DOF of the object is represented by the vector

$$s = (w_x, w_y, w_z, t_x, t_y, t_z), \tag{2.1}$$

where $\mathbf{W} = (w_x, w_y, w_z)$ corresponds to the rotation and (t_x, t_y, t_z) to the translation parameters. Considering perspective projection, the rigid body transformation between the world and the camera coordinate frame is related by a 3x3 rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ in $\mathbb{R}^3$. By combining $\mathbf{R}$ and $\mathbf{t}$, it is possible to construct a 4x4 modelview matrix $\mathbf{T}$ used to convert the 3D points from the world coordinate frame to the camera coordinate frame, such that:

$$\mathbf{X}_c = \mathbf{T}\mathbf{X}_w = [\mathbf{R}|\mathbf{t}]\mathbf{X}_w. \tag{2.2}$$

Rotation by an angle θ about an axis in the direction of the unit vector $r = (r_x, r_y, r_z)$ can be represented by the following equation:

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} r_x^2(1-\cos\theta)+\cos\theta & r_x r_y(1-\cos\theta)-r_z\sin\theta & r_x r_z(1-\cos\theta)+r_y\sin\theta \\ r_y r_x(1-\cos\theta)+r_z\sin\theta & r_y^2(1-\cos\theta)+\cos\theta & r_y r_z(1-\cos\theta)-r_x\sin\theta \\ r_x r_z(1-\cos\theta)-r_y\sin\theta & r_y r_z(1-\cos\theta)+r_x\sin\theta & r_z^2(1-\cos\theta)+\cos\theta \end{pmatrix}. \tag{2.3}$$

where

$$\theta = \sqrt{w_x^2 + w_y^2 + w_z^2};$$
$$r_x = \frac{w_x}{\theta}; r_y = \frac{w_y}{\theta} \text{ and } r_z = \frac{w_z}{\theta}.$$

The length of the vector $\mathbf{W}$ represents the angle of rotation and its direction represents the rotation axis. However, when using this representation, if the variation in the rotation is very large, non-linearities appear more frequently, resulting in poor performance. To solve this, information from a previous position and orientation $\mathbf{T}_0$ can be used to represent the amount of change in the parameters, resulting in:

$$\mathbf{T} = \mathbf{T_0}\mathbf{T_M}, \tag{2.4}$$

where

$$\mathbf{T_M} = \begin{pmatrix} & \mathbf{R} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

To project $\mathbf{X}_c$ into the image plane, $\mathbf{T}$ is multiplied to a matrix $\mathbf{K}$, containing the camera internal parameters:

$$\mathbf{K} = \begin{pmatrix} p_{11} & 0 & p_{13} & 0 \\ 0 & p_{22} & p_{23} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{2.6}$$

where p_{11} and p_{22} represent the focal length of the camera in the x and y direction, respectively, and (p_{13}, p_{23}) represent the point where the camera principal axis meets the image plane. 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. In summary:

$$h \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = \mathbf{P} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}. \tag{2.7}$$

2.3.2 Quadric surfaces

Quadric surfaces represent one of the main concepts used in our approach. They are formally defined as algebraic surfaces of degree 2 in $\mathbb{R}^3$ defined implicitly by:

$$\begin{aligned} f(x,y,z) &= a_1x^2 + a_2y^2 + a_3z^2 + 2a_4xy + 2a_5yz + \\ &\quad 2a_6xz + 2b_1x + 2b_2y + 2b_3z + c = 0, \end{aligned} \tag{2.8}$$

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. Some examples of quadric surfaces include spheres, ellipsoids and paraboloids.

By using homogeneous coordinates, quadrics can be conveniently written in matrix form as:

$$f(x,y,z) = \mathbf{X}_w^T Q \mathbf{X}_w = 0, \tag{2.9}$$

with $\mathbf{Q}$ representing the following 4x4 symmetric matrix:

$$\mathbf{Q} = \left[\begin{array}{ccc|c} a_1 & a_4 & a_6 & b_1 \\ a_4 & a_2 & a_5 & b_2 \\ a_6 & a_5 & a_3 & b_3 \\ \hline b_1 & b_2 & b_3 & c \end{array}\right] = \begin{bmatrix} Q_3 & q \\ q^T & c \end{bmatrix}. \tag{2.10}$$

2.3.3 Conic curves

A conic curve is an algebraic curve in $\mathbb{R}^2$ defined implicitly by:

$$f(x,y) = c_1x^2 + c_2y^2 + c_3xy + c_4x + c_5y + c_6 = 0, \tag{2.11}$$

where $(c_1, ..., c_6)$ are real constants and c_1, c_2, c_3 are not all zero. Circles, ellipses and parabolas are some examples of conics. Similar to quadrics, a conic $\mathbf{C}$ can also be represented using homogeneous coordinates:

$$\mathbf{x}^T\mathbf{C}\mathbf{x} = 0 \therefore \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} c_1 & \frac{c_3}{2} & \frac{c_4}{2} \\ \frac{c_3}{2} & c_2 & \frac{c_5}{2} \\ \frac{c_4}{2} & \frac{c_5}{2} & c_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2.12}$$

2.3.4 Apparent contour of quadrics

The apparent contour equation employed in this thesis uses the notation developed by Cipolla & Giblin [29]. Considering the camera center is positioned at $\mathbf{O} = (0, 0, 0, 1)^T$ and using the notation given in Equation 2.10, leads to:

$$\mathbf{A} = \begin{pmatrix} qq^T - cQ_3 & 0 \\ 0 & 0 \end{pmatrix}. \tag{2.13}$$

If the image plane is at $z = f$, the apparent contour in this plane is defined as the intersection of the cone of tangent rays with the image plane, i.e., the points $\mathbf{x} = (x, y, f)^T$ where:

$$\mathbf{x}^T(qq^T - cQ_3)\mathbf{x} = 0 \therefore \mathbf{x}^T\mathbf{A}\mathbf{x} = 0 \tag{2.14}$$

such that $qq^T - cQ_3$ represents the parameters of the corresponding conic. Additional information about calculation of this equation is given in Appendix **B**.

2.4. Concluding remarks

This chapter presented a summary of some existent tracking approaches, including markers and markerless methods.

Although the use of markers provides robust tracking, they are not the appropriate solution for the target object in this research: markers are more suitable for objects with flat faces and in some scenarios, such as the augmented prototyping application developed in Chapter 6, multiple markers are necessary to allow visualization of the augmented information on the target object from different views, making them intrusive elements in the scenario.

From the analysis of previous approaches using natural features, an edge-based approach was chosen as the best option. However, most of the edge-based approaches proposed so far mainly deals with polyhedral objects having flat faces and sharp edges or simple curved shapes. In the next chapter, more details about our framework and the new representation developed for accurate and efficient curved surfaces tracking are presented.

CHAPTER 3

Proposed Tracking Framework

This chapter presents a detailed description of the tracking framework developed in this thesis. In the first section, details about the offline stage are presented, with the necessary steps for constructing the proposed quadrics representation. The online stage is explored in the subsequent section, based on a general edge-based tracking system, with the proper modifications to handle the apparent contour of curved surfaces.

3.1. Framework overview

The framework proposed in this thesis uses an edge-based tracking system with a 3D model of the target object prepared beforehand. An overview is shown in Figure 3.1. During the offline stage (A), the camera internal parameters used to project 3D points in the image plane are calculated and an accurate 3D model representing the object is constructed. This model is simplified to obtain a sparse mesh to be used during tracking and its data are used to create an implicit model representing local approximations of the entire object. In the online stage (B), steps (i) to (v) are similar to a standard edge-based tracking [18].

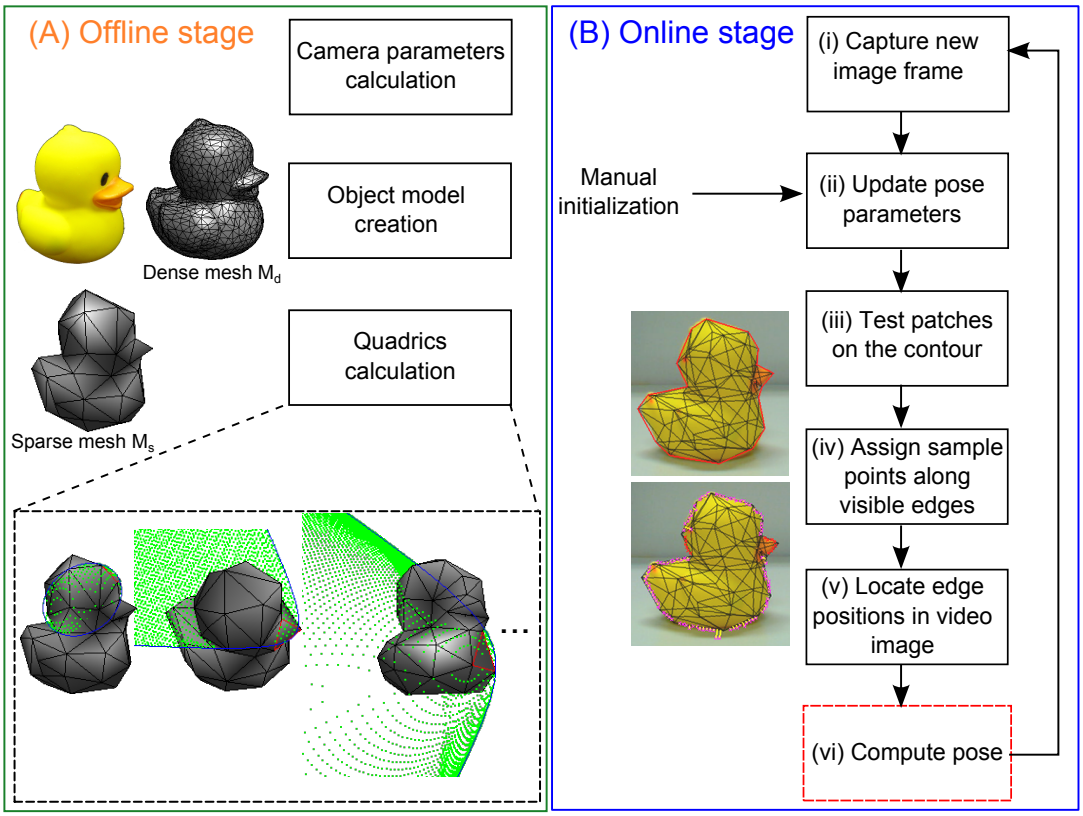


Figure 3.1. Overview of the tracking framework: (A) The offline stage with the camera parameters calculation, model creation and quadrics calculation. (B) The online stage, with the optimization loop highlighted in red.

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). Control points are assigned to the visible edges (iv) of the patches located on the object's apparent contour and used to find nearby edge points (v).

For step (vi), the main changes implemented to use the apparent contour represented by conic curves are shown in Figure 3.2: a cost function (Equation 3.17) is calculated based on the distance between the detected edge points and the conic curves on the contour. If the error value returned by the cost function is smaller than a threshold, the loop returns to step (i); otherwise, the pose parameters are refined.

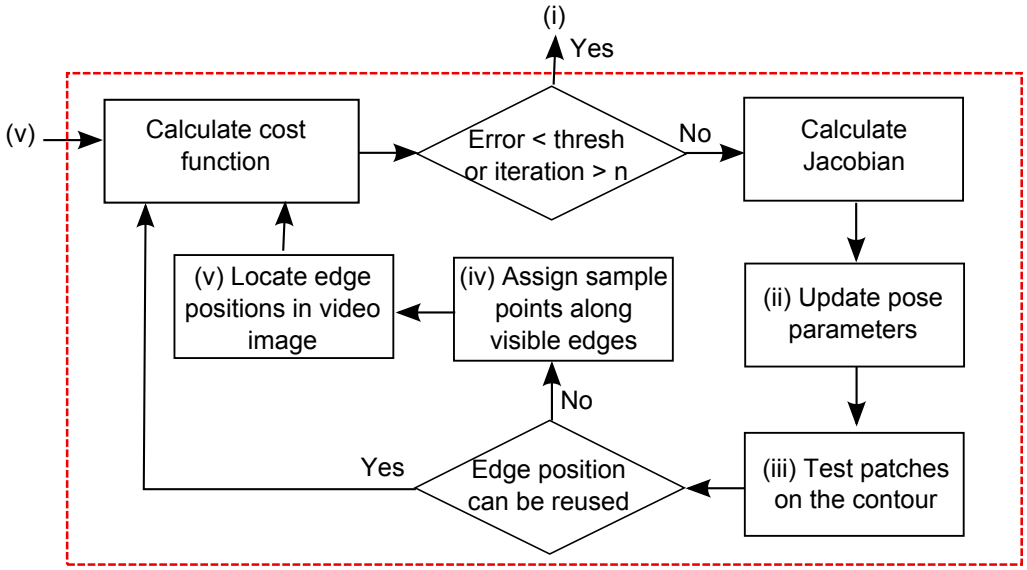


Figure 3.2. Step (vi) in details.

After updating the pose parameters, the edge positions are tested again. This is because a small movement of the apparent contour can change the correct correspondence between the detected edge points and the patches on the contour. If the edge positions can be reused, the tracking loop restarts; otherwise, steps (iv) and (v) are repeated.

3.2. Offline stage

In this stage, parameters of matrix $\mathbf{K}$ (Equation 2.6), representing the camera internal parameters, are obtained by calibrating the camera using OpenCV and a chessboard image.

An accurate 3D model $\mathbf{M}_d$ of the target object is also constructed and for our experiments (Chapter 5), 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 3D modeling software Blender. A remeshing method is applied on $\mathbf{M}_d$ to create a sparse mesh $\mathbf{M}_s$. Both meshes are required in the offline calculation of the quadrics described in Subsection 3.2.2.1. During the online stage, however, $\mathbf{M}_d$ is discarded and only $\mathbf{M}_s$ is used.

3.2.1 Polygonal mesh simplification

Several approaches for simplifying polygonal meshes are available in the literature [100, 101]. The choice of the method to be used in our framework considered mainly the following factors: speed, high fidelity to the original appearance of the object and easy implementation. Since the main goal of our framework is not the simplification method, only two approaches were tested and compared:

- *Poly Reducer* (PR) [102], an extension included in Blender. It simplifies the mesh using a decimation approach, removing vertices at each iteration.
- *Quadric Error Metric* (QME), developed by Garland [103] and available in the software package QSlim. It applies a sequence of vertex pair contractions, each pair having a cost associated, calculated using a quadric metric. At each iteration, the lowest cost pair is contracted.

Two polygonal meshes were used to evaluate the methods mentioned above: the angel and the bunny[1], whose original dense meshes have 25,370 and 25,000 patches, respectively. Figure 3.3 and Figure 3.4 show some of the polygonal meshes obtained after the simplification.

Numerical comparison of the results obtained from both approaches was performed using the software Metro [104], which evaluates the difference *diff* between two triangular meshes. In this method, no knowledge about the simplification approach used is necessary and *diff* is evaluated on the basis of the approximate error between the meshes.

Considering M_1 the original dense polygonal mesh and M_2 the simplified mesh, the approximate error between them is defined in Metro as the distance between corresponding sections of the meshes. The surface of the first mesh (pivot mesh) is sampled and for each element, a *point-to-surface* distance to the non-pivot mesh is calculated. However, since this distance is not symmetric, both a forward distance $\mathrm{E}(M_1 \rightarrow M_2)$ and a backward distance $\mathrm{E}(M_2 \rightarrow M_1)$ are calculated and a two-sided distance (Hausdorff distance) is obtained by taking the maximum of $\mathrm{E}(M_1 \rightarrow M_2)$ and $\mathrm{E}(M_2 \rightarrow M_1)$.

[1] 3D model of the bunny object is available at http://graphics.stanford.edu/data/3Dscanrep/

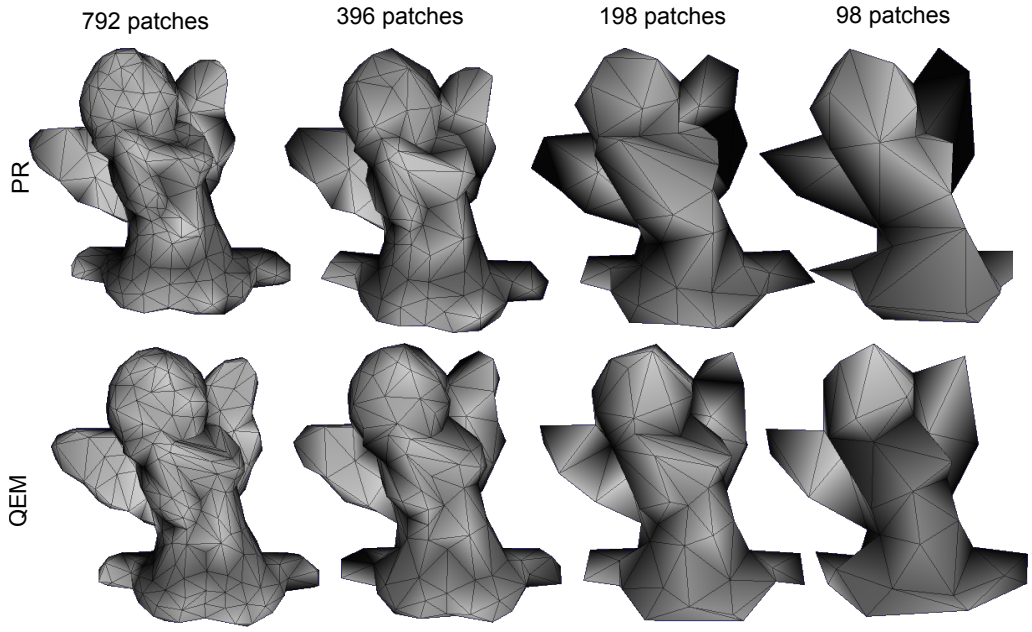


Figure 3.3. Comparison between polygonal mesh simplification using PR and QEM for the Angel.

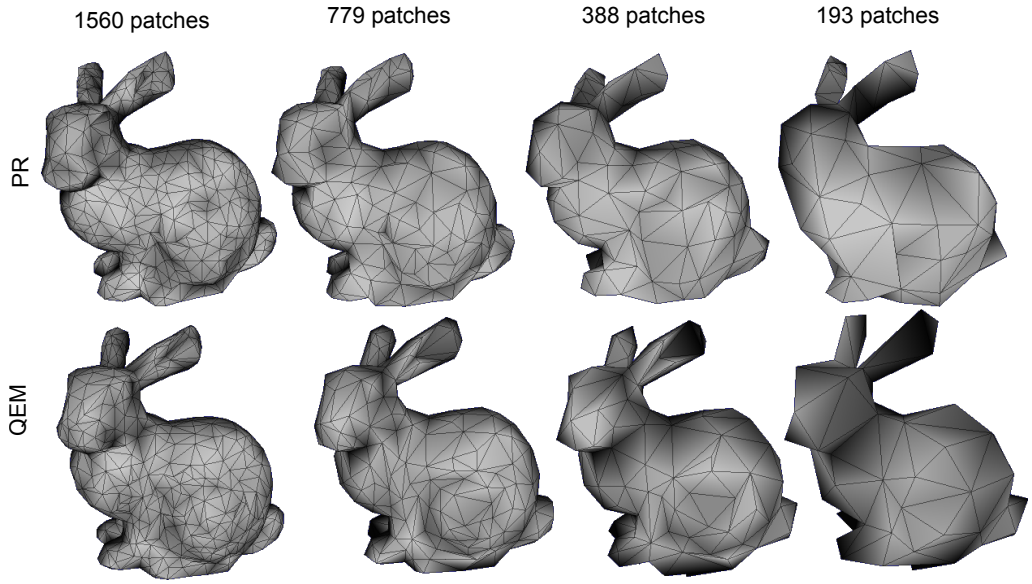


Figure 3.4. Comparison between polygonal mesh simplification using PR and QEM for the Bunny.

Table 3.1 presents the result of the Hausdorff distance, with the error between the models for both objects. Results for the *forward* and *backward* distance can be found in Appendix **C**. For the angel, QEM presented better results with lower error in all cases. For the bunny, although PR presented better results for 779 and 388 patches, the difference was not very significative to choose it over QEM. Furthermore, considering the computational time, QEM is able to perform faster: While PR takes approximately 5-12 sec. to simplify the polygonal mesh, QEM takes about 1-2 sec. to simplify the same model. Therefore, in our approach, QEM was chosen as the polygonal simplification method.

Lastly, the choice of the final model to be used for tracking (i.e., the lower limit of the number of patches that can be used in our framework), was visually done, considering two conditions: (a) The quality of the final mesh, provided that no fundamental part of the mesh was deleted and (b) the number of internal vertices it can produce (to be explained in Section 3.2.2.2).

Table 3.1. Housdorff distance

Object	**# of patches (M_2)**	**Method**	**Distance**
Angel	792 patches	PR	0.014928
		QEM	0.013797
	396 patches	PR	0.035032
		QEM	0.02682
	198 patches	PR	0.055519
		QEM	0.049604
	98 patches	PR	0.098271
		QEM	0.06694
Bunny	1560 patches	PR	0.176407
		QEM	0.081031
	779 patches	PR	0.279456
		QEM	0.291398
	388 patches	PR	0.379191
		QEM	0.496251
	193 patches	PR	0.55219
		QEM	0.496251

3.2.2 Quadrics patch representation

As mentioned in Chapter 1, when dealing with curved objects, high quality meshes are required to accurately represent the object's shape. However, this representation is not computationally efficient, affecting applications where real-time response is expected, such as in AR.

A survey describing some methods for recovering quadrics from triangle meshes can be found in [105]. One simple solution consists of dividing the shape of the target object in curved primitives such as spheres, cylinders or ellipsoids and use them for tracking. This has been previously implemented by Drummond & Cipolla [27] for tracking articulated objects and by Stenger et al. [96] for hands tracking, both using truncated quadrics. Although it is an efficient method, it is restricted to a small class of shapes - with parts resembling quadrics. In our suggested representation, curved primitives represented by quadrics are also used, but our approach is not affected by the same issue because each patch of the mesh (not each part of the object) has a corresponding quadric equation. Hence, it is possible to easily handle simple shapes similar to the duck model as well as complex shapes, such as the angel figurine shown in Chapter 1, Figure 1.3.

3.2.2.1 Quadrics calculation

In the proposed *quadrics patch representation*, each patch on the sparse 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.

In our previous approach [106], quadrics fitting is achieved by a naive calculation using algebraic fitting. The polygonal meshes have quad patches and for each patch, quadrics are calculated by using the coordinates and the normal values of its four vertices. Additionally, vertices belonging to patches surrounding the patch being analyzed are also included in the fitting. Finally, least-squares are used to find the values of the quadric parameters. However, the use of adjacent patches in this implementation may not work in some cases, especially when one of them has different curvature. This problem becomes more noticeable when dealing with sparse meshes, since less data points are available.

To achieve a better quadrics fitting with sparse meshes, an approach considering geometric fitting is suggested. Geometric approaches in general calculate the shortest distance from the given point to the fitting surface using iterative algorithms. One example of geometric fitting of quadrics, including a comparison with results obtained with algebraic fitting can be found at [107].

Equation 2.8 is used to calculate the quadric parameters, where the value $c = 1.0$ is fixed and the other nine parameters are calculated for every patch based on the corresponding vertices positions V_{pk}. However, since the model has triangle patches, the existent data in only one patch is not enough to find a solution. To solve this problem, a set of *internal vertices* is implemented for all patches in the mesh.

3.2.2.2 Internal vertices

The internal vertices $IV_j = (x_j, y_j, z_j)$, with $j = \{1, 2, ..., n\}$, are defined as vertices that originally belong to the dense mesh $\mathbf{M}_d$, but during the simplification process are deleted. Figure 3.5 illustrates the process to obtain the internal vertices. Each patch p_k in the new mesh have n internal vertices, whose value depends on how many of the deleted vertices belong to the area bounded by p_k.

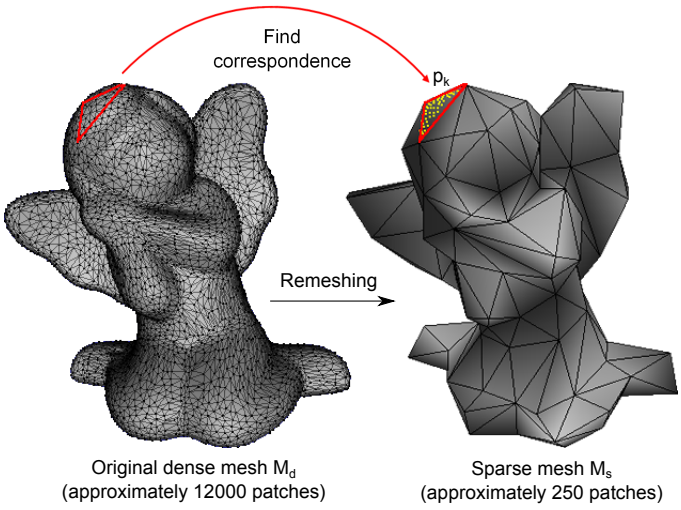


Figure 3.5. Internal vertices calculation. A random patch is highlighted and the yellow dots represent the internal vertices obtained after evaluation of the dense mesh. This process is repeated for all patches in the offline stage.

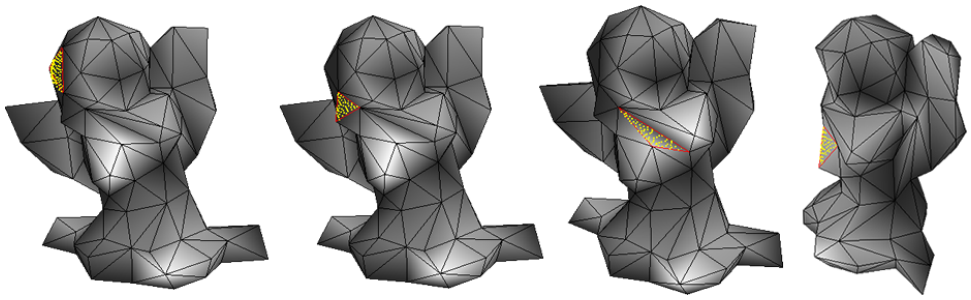


Figure 3.6. Internal vertices example.

Data calculated in this step are used only in the offline stage for conics fitting. During the online stage, just the original vertices V_{pk} are used to avoid increasing the computational time in the tracking step.

To verify if the point $A = (a_x, a_y, a_z)$ is an internal vertex of the patch p_k, first a bounding test is performed comparing each of the coordinates of A with the coordinates of the vertices of p_k. If these coordinates are not inside the interval bounded by p_k, they can be discarded right away.

If A is a candidate point, the next step consists of checking whether A is within the boundaries of p_k in 3D space. In this case, an extension to 3D of the method to check if a point is inside a triangle in 2D using barycentric coordinates [108] was implemented. Figure 3.6 shows some examples of the obtained result, with the internal vertices being represented by the yellow dots.

3.2.2.3 Quadrics evaluation

The quadrics fitting implemented in our framework uses the internal vertices to find the quadric that best fits each patch. Hence, the sparser the mesh is, more data are available for the calculation and a better fitting can be achieved. If less than nine internal vertices 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.

In our experiments, although the obtained fitting is suitable for most of the patches in the mesh, in some cases the fitting may not be the most correct one, which affects the tracking as a consequence. To improve the overall performance of our method, the quadric fitting error is also evaluated, based on the result returned by the implicit equation. If it exceeds a threshold, the patch is discarded and the corresponding edge contour is not considered for tracking.

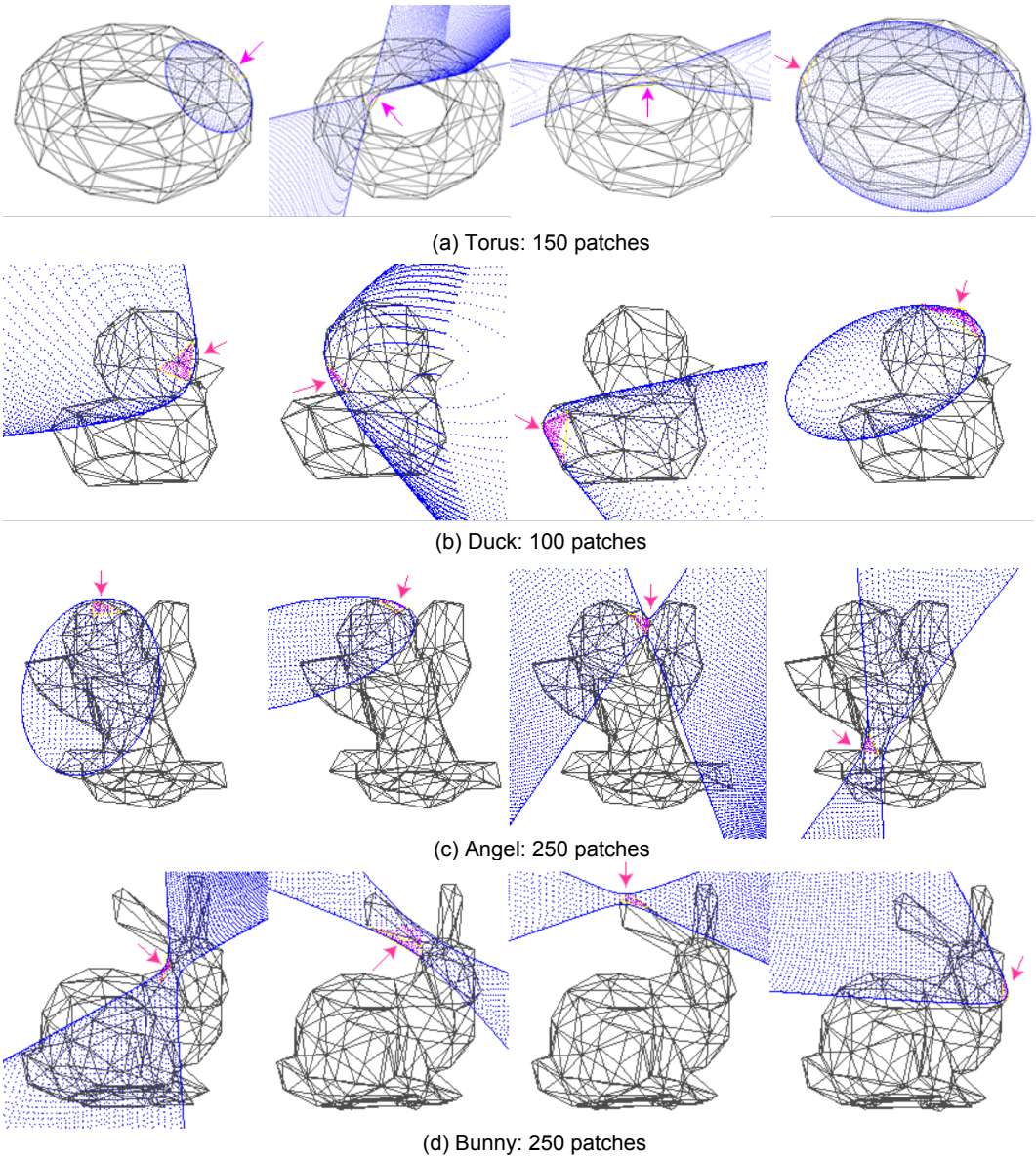


Figure 3.7. Some examples obtained with the quadrics fitting for four different objects: (a) Torus, (b) Duck, (c) Angel and (d) Bunny.

The number of internal vertices available for each object is directly related to the shape of the object and the method used to obtain its 3D model. For instance, simple shapes can be manually modeled using a 3D modeling software. Then, a smoothing function can be used to generate more vertices in the final model and hence more data for the quadrics fitting. If the object model is obtained using a 3D laser scanner, obtaining data for the internal vertices becomes easier because the amount of point data available are usually high.

Figure 3.7 shows some examples obtained with the quadrics fitting. For each object used in our experiments, approximately more than 30 internal vertices was used (this number also depends on the patch location).

3.3. Online stage

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. OpenGL is used to determine the visible edges, where control points are placed and used to find points on the apparent contour for matching. Lastly, a cost function is minimized to find the correct parameters of the object's current pose. It represents the module in our framework with the most significative differences compared to previous approaches.

3.3.1 Contour patches selection

To find which patches from the mesh are on the current object's contour, first a test is performed to find all patches that are facing the camera and at the same time have a neighbor patch facing the opposite direction.

The next step consists in finding which model edges represent the contour lines. In this case, the dot product between the vector from the camera center $\mathbf{O}_c$ to each vertex V_{pki} and the normal vector $N_{pki} = (n_{xi}, n_{yi}, n_{zi})$ at this vertex is performed for all patches to give the vertex direction:

$$dir = (V_{pki} - \mathbf{O}_w) \cdot N_{pki} \tag{3.1}$$

Since this test is performed in the world coordinate frame, it is necessary to convert the coordinates of $\mathbf{O}_c = (0, 0, 0, 1)^T$ from the camera coordinate frame to coordinates in the world coordinate frame $\mathbf{O}_w = (o_{xw}, o_{yw}, o_{zw})$. This can be done using Equation 2.2, resulting in:

$$\mathbf{O}_w = \mathbf{T}^{-1}\mathbf{O}_c = \begin{pmatrix} ti_{11} & ti_{12} & ti_{13} & ti_{14} \\ ti_{21} & ti_{22} & ti_{23} & ti_{24} \\ ti_{31} & ti_{32} & ti_{33} & ti_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} ti_{14} \\ ti_{24} \\ ti_{34} \\ 1 \end{pmatrix} \tag{3.2}$$

Thus, Equation 3.1 becomes:

$$dir = (ti_{14} - x_i, ti_{24} - y_i, ti_{34} - z_i) \cdot (n_{xi}, n_{yi}, n_{zi}) \tag{3.3}$$

The sign of this calculation is used to obtain the direction of each vertex. If there is one vertex with different sign than the other two vertices, then it means they are in opposite directions. The edge connecting the vertices pointing to the same direction is selected as the edge on the contour used for matching with the edges detected on the video image (Figure 3.8).

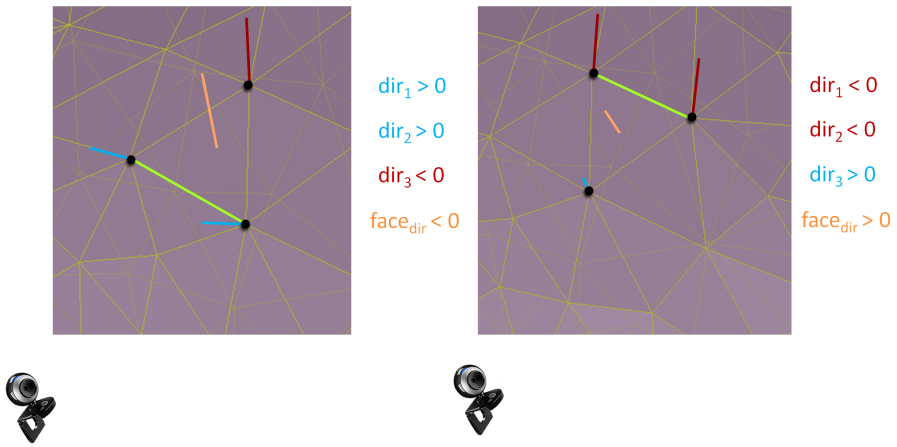


Figure 3.8. Contour edges calculation, where $dir_i, i = 1, 2, 3$ represents the direction of each vertex. If $dir_i > 0$, it means the vertex i is facing the camera; otherwise, it is on the opposite side. The contour edge is highlighted in green.

However, for some object shapes, using this approach straightforward includes edges that are not on the object's contour. For instance, consider the angel figurine model shown in Figure 3.9. Some of the patches on the angel's arms also fulfill the conditions previously mentioned, but they do not belong to the object's contour. To remove these edges, OpenGL is used to perform a depth test and select only the edges located on the actual contour.

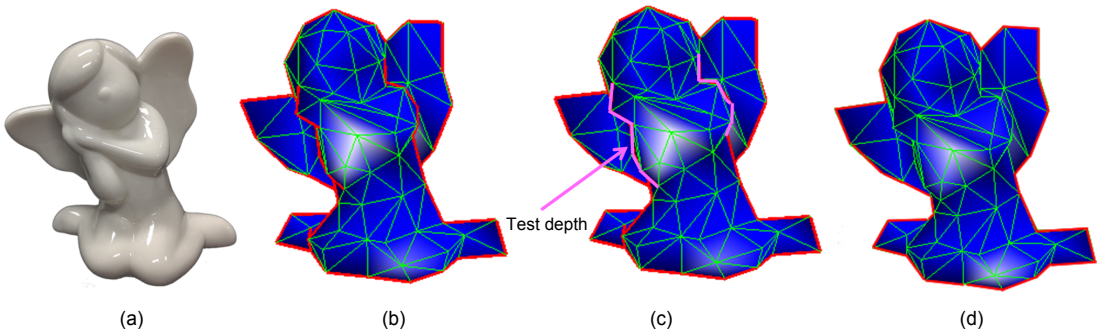


Figure 3.9. Selection of the edges on the outside contour. (a) Target object. (b) Candidate contour edges. (c) Removal of the highlighted edges that do not belong to the contour by testing its depth. (d) Object's actual contour.

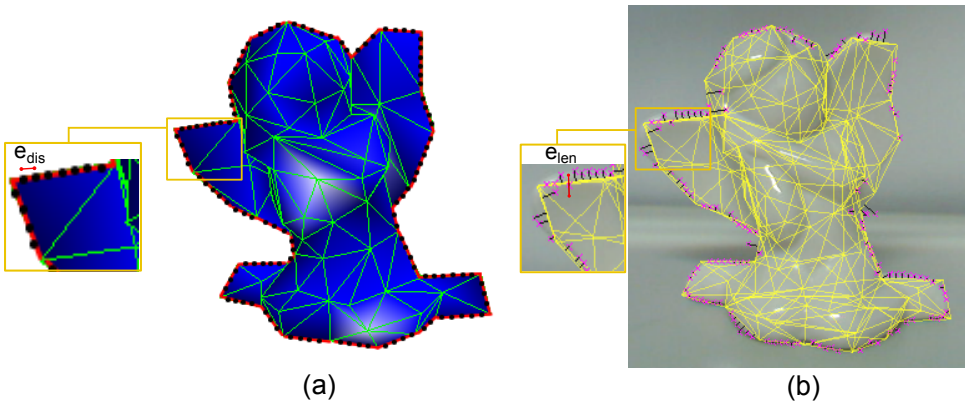


Figure 3.10. Edge points detection and matching: (a) Control points are sampled along the visible contour and (b) used to match with detected edges in the video image. Nearby detected edges are represented by the pink crosses.

3.3.2 Edge points detection and matching

Once the visible edges on the contour are selected, control points are assigned along them, separated by a regular distance e_{dis}, similarly to the method used by Drummond & Cipolla [18]. Then, search in the normal direction with length e_{len} of each control point is done to search for nearby pixels with intensity discontinuity, as shown in Figure 3.10.

Since the control points are placed along the edges selected on the previous section, in some cases, just part of the edge is on the contour, as illustrated in Figure 3.11. Therefore, the control points position are also tested to discard the ones that are sampled on the part of the edge outside the contour.

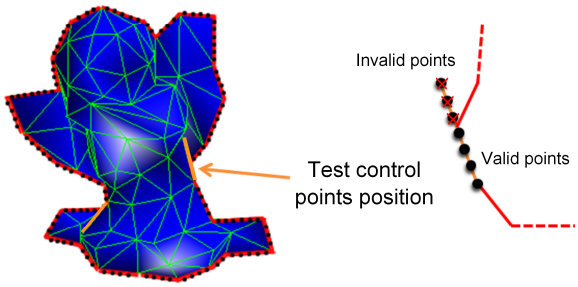


Figure 3.11. Control points selection. Highlighted by orange, the edges partially located on the contour. In this case, control points cannot be sampled along the entire length of the edge.

3.3.3 Apparent contour equation

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 of the conic varies according to the length of the edge of p_k located on the contour. This is illustrated in Figure 3.12: (a) p_k has a quadric Q calculated in world coordinates, which is converted to the camera coordinate frame and projected in (b) image coordinates. Only the conic segment passing exactly on the edge of p_k is used, highlighted by the pink line. This is done for all patches on the contour, generating the apparent contour as a collection of different conic segments as shown in Figure 3.13.

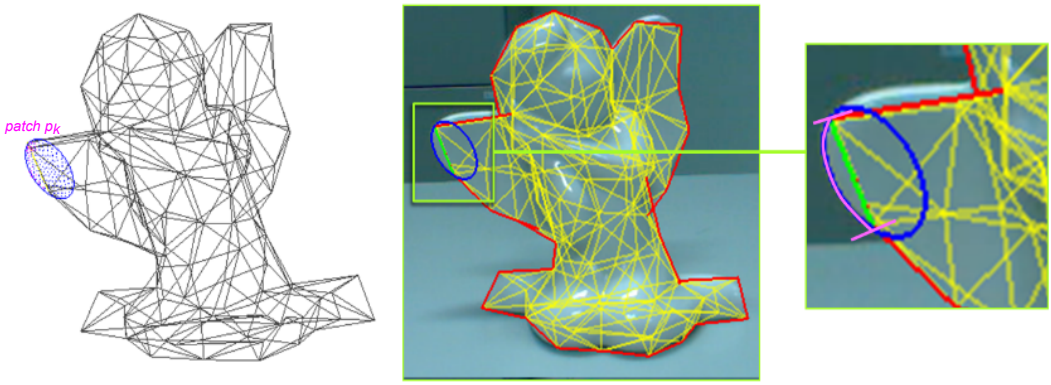


Figure 3.12. Quadrics projection of a patch p_k: In (a) world coordinates and (b) image coordinates, with the length of the conic used highlighted in pink.

3.3.3.1 Quadrics projection

3.3.3.1.1 World to camera coordinates

The quadric parameters $\mathbf{Q}_w$ are converted from world to camera coordinates $\mathbf{Q}_c$ using the notation given in Equation 2.10. By rearranging it using a similar strategy employed in Equation 3.2, the following Equation is obtained:

$$\begin{aligned} \mathbf{X}_w^T \mathbf{Q}_w \mathbf{X}_w &= 0 \\ \mathbf{X}_c^T \mathbf{T}_i^T \mathbf{Q}_w \mathbf{T}_i \mathbf{X}_c &= 0 \end{aligned} \tag{3.4}$$

where $\mathbf{X}_w$ and $\mathbf{X}_c$ are points in the world and camera coordinates, respectively, and $\mathbf{T}_i$ the inverse of the transformation matrix $\mathbf{T}$ (Equation 2.5).

Therefore, the quadric parameters in camera coordinates are:

$$\mathbf{Q}_c = \mathbf{T}_i^T \mathbf{Q}_w \mathbf{T}_i = \begin{pmatrix} a_{c1} & a_{c4} & a_{c6} & b_{c1} \\ a_{c4} & a_{c2} & a_{c5} & b_{c2} \\ a_{c6} & a_{c5} & a_{c3} & b_{c3} \\ b_{c1} & b_{c2} & b_{c3} & c_c \end{pmatrix} \tag{3.5}$$

3.3.3.1.2 Camera to image coordinates

The quadric parameters are converted to image coordinates by replacing Equation 3.5 in Equation B.8, representing the apparent contour of a quadric:

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = 0$$

$$\begin{pmatrix} x & y & f \end{pmatrix} \begin{pmatrix} b_{c1}^2 - c_c a_{c1} & b_{c1}b_{c2} - c_c a_{c4} & b_{c1}b_{c3} - c_c a_{c6} \\ b_{c1}b_{c2} - c_c a_{c4} & b_{c2}^2 - c_c a_{c2} & b_{c2}b_{c3} - c_c a_{c5} \\ b_{c1}b_{c3} - c_c a_{c6} & b_{c2}b_{c3} - c_c a_{c5} & b_{c3}^2 - c_c a_{c3} \end{pmatrix} \begin{pmatrix} x \\ y \\ f \end{pmatrix} = 0. \tag{3.6}$$

However, Equation B.8 considers the origin of the coordinates in the image plane is located at the principal point. Assuming $f = p_{11}$, this means:

$$h' \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & 0 & 0 \\ 0 & p_{11} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix}, \tag{3.7}$$

which differs from the matrix $\mathbf{K}$ used in our approach:

$$h \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & 0 & p_{13} \\ 0 & p_{22} & p_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix}. \tag{3.8}$$

To allow the use of Equation B.8, Equation 3.8 was rearranged as:

$$h \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & p_{13} \\ 0 & \frac{p_{22}}{p_{11}} & p_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_{11} & 0 & 0 \\ 0 & p_{11} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix}. \tag{3.9}$$

Replacing Equation 3.7 in Equation 3.8 leads to:

$$h \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = h' \begin{pmatrix} 1 & 0 & p_{13} \\ 0 & \frac{p_{22}}{p_{11}} & p_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}. \tag{3.10}$$

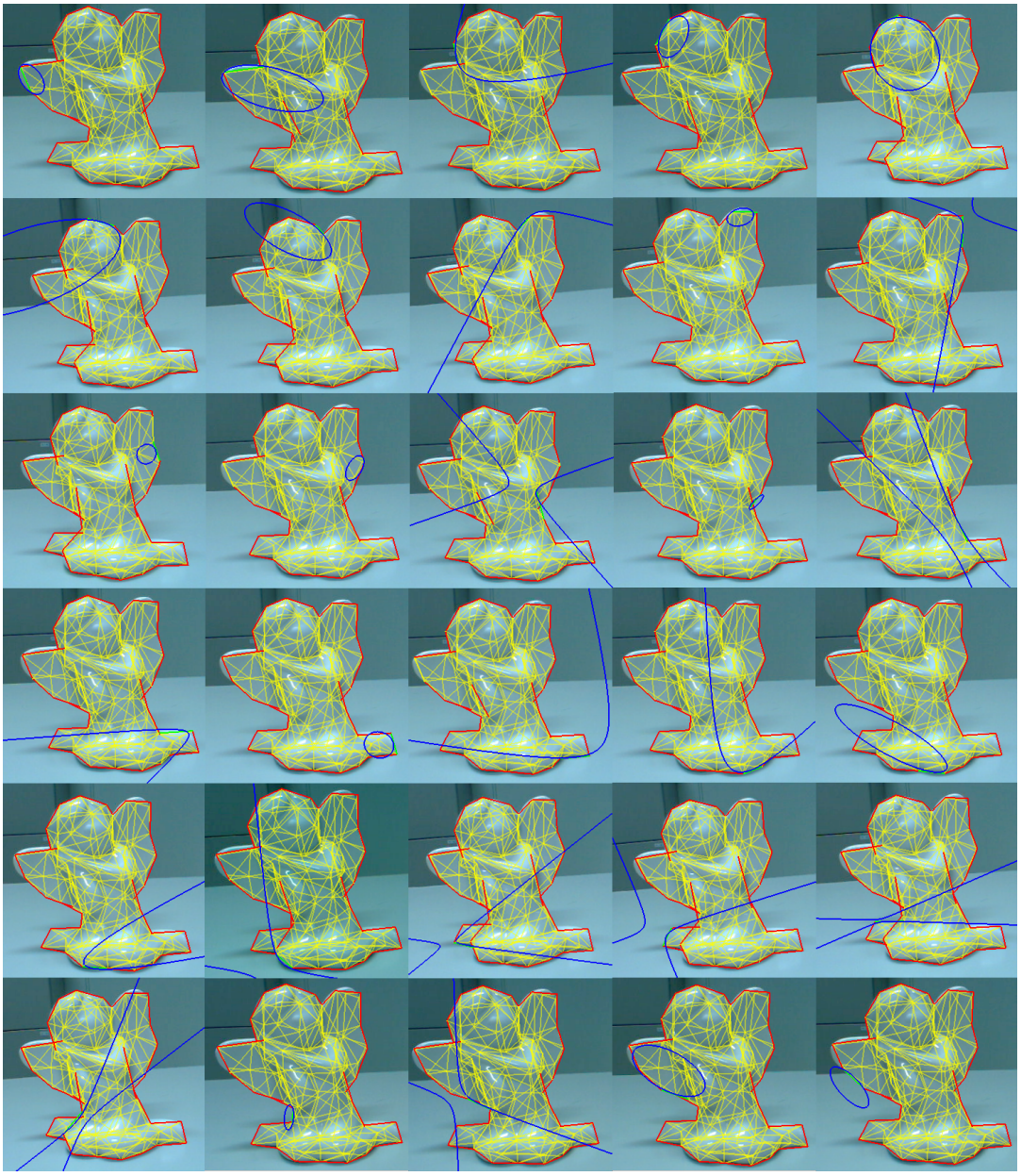


Figure 3.13. Conic curves on the contour: Highlighted in blue, the conics obtained using the proposed quadrics patch representation for the patches belonging to the object contour.

Assuming $h = h'$:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & p_{13} \\ 0 & \frac{p_{22}}{p_{11}} & p_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \therefore \begin{pmatrix} x \\ y \\ p_{11} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \frac{p_{13}}{p_{11}} \\ 0 & \frac{p_{22}}{p_{11}} & \frac{p_{23}}{p_{11}} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ p_{11} \end{pmatrix}. \tag{3.11}$$

Therefore:

$$\begin{pmatrix} x' \\ y' \\ p_{11} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \frac{p_{13}}{p_{11}} \\ 0 & \frac{p_{22}}{p_{11}} & \frac{p_{23}}{p_{11}} \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \\ p_{11} \end{pmatrix}. \tag{3.12}$$

Simplifying the notation:

$$\begin{pmatrix} x' \\ y' \\ p_{11} \end{pmatrix} = \begin{pmatrix} pi_{11} & 0 & pi_{13} \\ 0 & pi_{22} & pi_{23} \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \\ p_{11} \end{pmatrix} = \bar{\mathbf{K}} \begin{pmatrix} x \\ y \\ p_{11} \end{pmatrix}. \tag{3.13}$$

Thus, the original Equation 3.6 becomes:

$$\begin{pmatrix} x & y & p_{11} \end{pmatrix} \bar{\mathbf{K}}^T \mathbf{A} \bar{\mathbf{K}} \begin{pmatrix} x \\ y \\ p_{11} \end{pmatrix} = 0. \tag{3.14}$$

A final modification is necessary to change $\mathbf{x} = (x, y, p_{11})^T$ to $\mathbf{x} = (x, y, 1)^T$:

$$\begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} pi_{11} & 0 & 0 \\ 0 & pi_{22} & 0 \\ pi_{13}p_{11} & pi_{23}p_{11} & p_{11} \end{pmatrix} \mathbf{A} \begin{pmatrix} pi_{11} & 0 & pi_{13}p_{11} \\ 0 & pi_{22} & pi_{23}p_{11} \\ 0 & 0 & p_{11} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0. \tag{3.15}$$

Using the notation given in Equation 2.12, the parameters of the quadric in image coordinates (conic curve) is:

$$\begin{bmatrix} c_1 & \frac{c_3}{2} & \frac{c_4}{2} \\ \frac{c_3}{2} & c_2 & \frac{c_5}{2} \\ \frac{c_4}{2} & \frac{c_5}{2} & c_6 \end{bmatrix} = \begin{pmatrix} pi_{11} & 0 & 0 \\ 0 & pi_{22} & 0 \\ pi_{13}p_{11} & pi_{23}p_{11} & p_{11} \end{pmatrix} \mathbf{A} \begin{pmatrix} pi_{11} & 0 & pi_{13}p_{11} \\ 0 & pi_{22} & pi_{23}p_{11} \\ 0 & 0 & p_{11} \end{pmatrix}. \tag{3.16}$$

3.3.4 Pose parameters computation

The pose parameters are calculated as a minimization problem and Levenberg-Marquardt Algorithm (LMA) is used to compute the registration that minimizes the distance between projected and observed features.

The following cost function $e_c(s)$ is minimized:

$$e_c(s) = \frac{1}{n}\sum_{i=1}^{n} d_e(X_i, \varphi(c(Q_i, s)))^2 \tag{3.17}$$

In the equation above, 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 (conic curve). Finally, the function d_e represents the distance between projected features and detected points $\mathbf{X}_i$ in image coordinates.

If the initial pose is represented by s_0 and n edge points are found in the video image, a nx6 Jacobian matrix is constructed and used to calculate the pose parameters as a solution of the equation:

$$d_e = J_{es}(s_0)\Delta s. \tag{3.18}$$

where Δs represents a small variation in s and J_{es} is the Jacobian matrix of a function describing the influence of each element of d_e on each element of s:

$$J_{es} = \frac{\partial d_e}{\partial s}. \tag{3.19}$$

Solving Equation 3.18 for Δs, where the sum of square errors of d_e are minimized, leads to:

$$\Delta s = (\boldsymbol{J}_{\boldsymbol{es}}^{\mathbf{T}}\boldsymbol{J}_{\boldsymbol{es}})^{-1}\boldsymbol{J}_{es}^{\mathrm{T}} d_e. \tag{3.20}$$

Finally, the pose vector can be iteratively updated using:

$$s_1 = s_0 + \Delta s. \tag{3.21}$$

Calculation of $\boldsymbol{J}_{\boldsymbol{es}}$ is described in details in Appendix **D**.

In addition, the influence of outliers is removed by using M-Estimators. The Tukey estimator [16] is used to weight the error returned by d_e:

$$w_i = \begin{cases} \frac{k^2}{6}\left\{1 - \left[1 - \left(\frac{d_e}{k}\right)^2\right]^3\right\}, if\ |d_e| \leqslant k \\ \frac{k^2}{6}, otherwise \end{cases} \tag{3.22}$$

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 3.22:

$$weight = \begin{cases} d_e\left[1 - \left(\frac{d_e}{k}\right)^2\right]^2, if\ |d_e| \leqslant k \\ 0, otherwise \end{cases} \tag{3.23}$$

3.3.5 Distance calculation

When dealing with static edges, d_e can be calculated by using the formula of the distance from a point to a line (Figure 3.14(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.

Mederos et al. [109] proposed an iterative approach to evaluate this distance, but this makes the original minimization given by Equation 3.17 impractical. Taubin[110] suggested an analytical approximation to the Euclidean distance of a point to an implicit curve. However, this first order approximated distance does not provide good results for our approach - it presents satisfactory results only when the detected point is very close to the conic.

Furthermore, in some cases the distance is erroneously calculated because it is considering the whole conic and not the corresponding conic segment passing exactly on the patch contour. In Figure 3.14(b), although p_1 is closer to the detected point p_0, this distance cannot be used because it is further from the contour edge. Therefore, development of a different strategy to evaluate this distance is necessary.

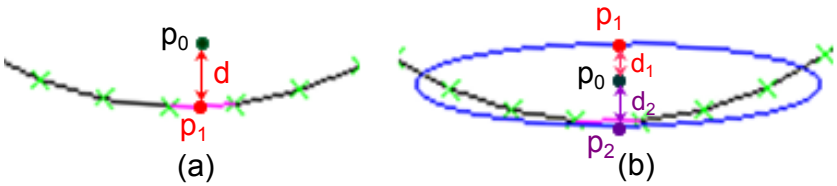


Figure 3.14. Distance calculation: (a) Evaluation using the edge from the mesh. (b) When dealing with conic curves, it is necessary to find the correct conic segment from which the distance will be calculated. In this case, although $d_1 < d_2$, the correct distance to be used is d_2.

3.3.5.1 Reference points

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 conic curve, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, from which the distance is evaluated. This position changes change according to p_0 position, as shown in Figure 3.15, with 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.

To find the position of the reference points, first p_0 is orthogonally projected on the edge from the mesh. The length d_1 of this projection is used to calculate the position of p'_1 and p'_2, as illustrated in Figure 3.16. Once this position is determined, two line equations l_1 (connecting p_0 and p'_1) and l_2 (connecting p_0 and p'_2) are calculated. The points p_1 and p_2 where each of the lines intersect the conic are used to find another line equation l_3. Finally, the distance d_2, representing an approximate distance from the detected point p_0 to the conic, is calculated using the distance of this point to l_3 (Figure 3.17).

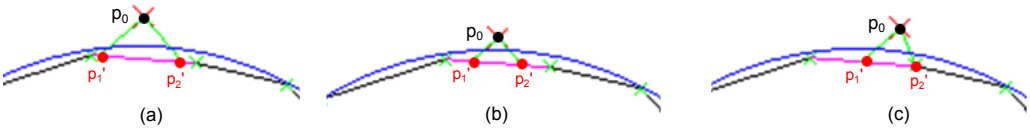


Figure 3.15. Different relationships between the detected point p_0 and the mesh edge as well as the conic.

3.3.5.2 Point to conic distance calculation

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:

$$\begin{cases} x_i = (x_0 - x'_i)p_i + x'_i \\ y_i = (y_0 - y'_i)p_i + y'_i \end{cases} \tag{3.24}$$

where $(c_1, ..., c_6)$ are real constants and c_1, c_2, c_3 are not all zero.

p_i is found by replacing Equation 3.24 in Equation 2.11, representing a general conic, resulting in the following equation:

$$\begin{aligned} &(c_1(x_0 - x'_i)^2 + c_2(y_0 - y'_i)^2 + c_3(x_0 - x'_i)(y_0 - y'_i))p_i^2 + \\ &(2c_1x'_i(x_0 - x'_i) + 2c_2y'_i(y_0 - y'_i) + c_3(x_0y'_i + x'_iy_0 - 2x'_iy'_i) + \\ &c_4(x_0 - x'_i) + c_5(y_0 - y'_i))p_i + \\ &(c_1x'^2_i + c_2y'^2_i + c_3x'_iy'_i + c_4x'_i + c_5y'_i + c_6) = 0 \end{aligned} \tag{3.25}$$

Equation 3.25 is a quadratic equation in the form $\mathbf{a}_i x_2 + \mathbf{b}_i x + \mathbf{c}_i = 0$ with components:

$$\begin{aligned} \mathbf{a}_i &= c_1(x_0 - x'_i)^2 + c_2(y_0 - y'_i)^2 + \\ &\quad c_3(x_0 - x'_i)(y_0 - y'_i) \\ \mathbf{b}_i &= 2c_1x'_i(x_0 - x'_i) + 2c_2y'_i(y_0 - y'_i) + \\ &\quad c_3(x_0y'_i + x'_iy_0 - 2x'_iy'_i) + \\ &\quad c_4(x_0 - x'_i) + c_5(y_0 - y'_i) \\ \mathbf{c}_i &= c_1x'^2_i + c_2y'^2_i + c_3x'_iy'_i + c_4x'_i + c_5y'_i + c_6 \end{aligned} \tag{3.26}$$

which is calculated by finding the roots of the respective quadratic equations:

$$p_i = \frac{-\mathbf{b}_i \pm \sqrt{\mathbf{b}_i^2 - 4\mathbf{a}_i\mathbf{c}_i}}{2\mathbf{a}_i} \tag{3.27}$$

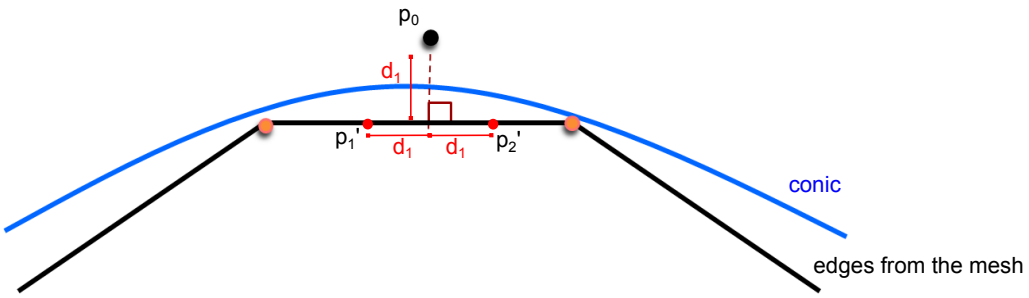


Figure 3.16. Reference points position: The length d_1 of the orthogonal projection of p_0 is used to find the position of p'_1 and p'_2.

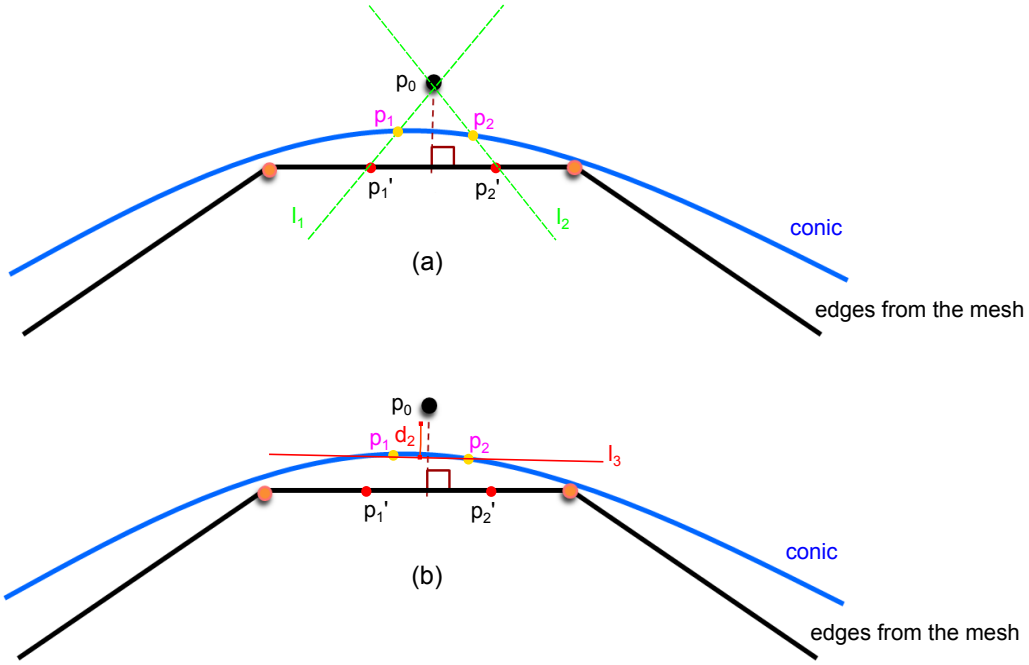


Figure 3.17. Point to conic distance: (a) First, the line passing through p_0 and p'_1 as well as the line passing through p_0 and p'_2 are calculated, to find two points on the conic, namely p_1 and p_2. (b) A third line equation, connecting p_1 and p_2, is calculated, which is used to find the approximate distance of p_0 to the conic.

Equation 3.27 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. Then, the distance from the point p_0 to the line passing through p_1 and p_2 is evaluated, being expressed by:

$$(y_2 - y_1)x + (x_1 - x_2)y + (x_2y_1 - y_2x_1) = 0 \tag{3.28}$$

with $a = (y_2 - y_1)$, $b = (x_1 - x_2)$ and $c = (x_2y_1 - y_2x_1)$. After this calculation, the formula for the distance of a point to a line can be used.

$$d_e = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} \tag{3.29}$$

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 3.18 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 affects the result if the correct correspondence is not always verified.

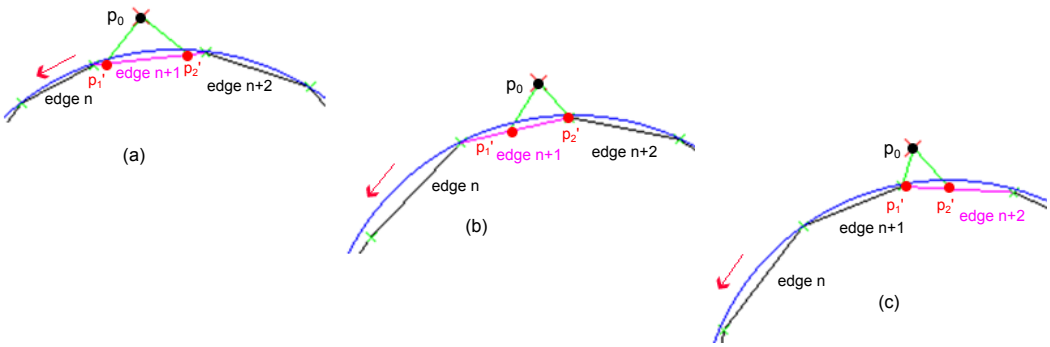


Figure 3.18. Updating the reference points position: A slight movement of the object to the left results in a new correspondence between detected point and contour edge.

3.4. Concluding remarks

In this chapter, a framework to solve the trade-off between computational efficiency and accuracy of non-textured rigid curved objects tracking is presented. To achieve this, the following ideas are introduced:

- Quadrics patch representation: Each patch of the sparse mesh has a corresponding quadric equation, aiming at reducing the error between detected and projected edges and improving accuracy. It is a simple representation, easy to calculate and efficient for dealing with curved shapes.
- Internal vertices: Introduced to improve the quadrics fitting, it is implemented using the relationship between the dense and the sparse mesh.
- Formulation of the distance between a point and a conic curve: A new method for evaluating the distance between projected and detected features is developed to attend the particularities of the quadrics patch representation.

Results of a complete evaluation of the proposed framework are given in Chapter 5.

CHAPTER 4

Dealing with different number of observable DOF

In general, many rigid objects found in the real world have six DOF, which means the Jacobian matrix $\boldsymbol{J_{es}}$ has rank six and pose parameters estimation is possible using the implementation developed in Chapter 3. However, there are some scenarios in which this estimation can fail, such as:

(a) The number of observable DOF is less than six and it does not change over time. For instance, the torus (five DOF) and the sphere (three DOF) shown in Figure 4.1 (a) make $\boldsymbol{J_{es}}$ rank-deficient.

(b) The object has variable number of observable DOF, which changes according to the viewpoint. For instance, when the handle of the mug on Figure 4.1(b) is visible, six DOF estimation is possible, but if the mug is rotated and the handle becomes occluded, the mug loses one DOF. If the mug is positioned again sideways, the six pose parameters cannot be correctly recovered.

(c) The object has six DOF but if after some point during tracking, the main body of the object is framed out and only part of it is shown. For example, the teapot in Figure 4.1 (c): When only the spout is visible to the camera, an ambiguity is detected and accurate pose estimation of the whole body of the teapot becomes difficult.

To deal with these problems, a method to calculate the measurable DOF of the target object is included in the proposed framework. Then, SVD is used to estimate the pose for objects with less than six DOF. The same method is used to trigger a recovery module when changes in the observable DOF of the target object is detected.

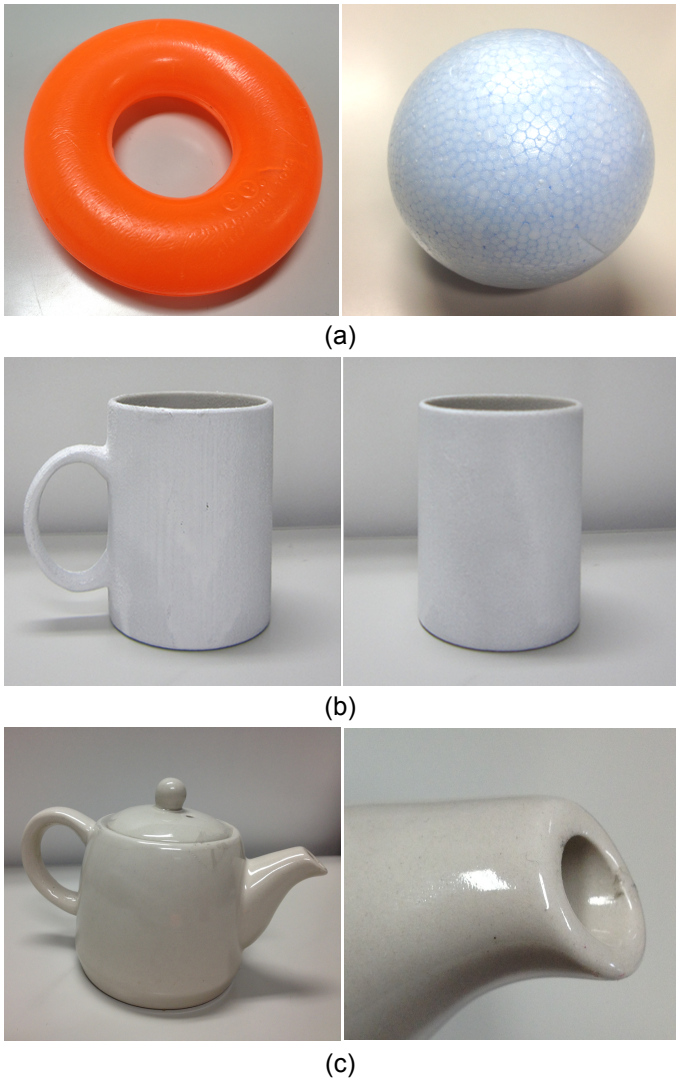


Figure 4.1. Examples in which pose parameters can be inaccurate: (a) torus and sphere, with less than six unobservable DOF, (b) a mug with different number of observable DOF depending on the viewpoint, and (c) a teapot with 6DOF, whose tracking of the main body cannot be recovered after part of it is framed out.

4.1. Measuring the object DOF

To allow the proposed framework handle the problems described above, first the measurable DOF of the target object is calculated by counting the non-zero singular values obtained from the SVD of $\boldsymbol{J_{es}}$ [31].

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 decomposed in:

$$\begin{aligned} \boldsymbol{J_{es}} &= \boldsymbol{U\Sigma V^T} \\ &= \begin{pmatrix} u_{11} & \dots & u_{1n} \\ \vdots & \ddots & \vdots \\ u_{n1} & \dots & u_{nn} \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & \sigma_6 \end{pmatrix} \begin{pmatrix} v_{11}^t & \dots & v_{16}^t \\ \vdots & \ddots & \vdots \\ v_{61}^t & \dots & v_{66}^t \end{pmatrix} \end{aligned} \quad (4.1)$$

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:

$$\Sigma = diag(\sigma_1, ..., \sigma_6); \sigma_1 > \sigma_2 > ... > \sigma_6 \quad (4.2)$$

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 value of the missing DOF becomes zero. However, due to computational approximations, these values may not be completely zero. Some examples are shown in Table 4.1, with objects having less than six observable DOF (except for the Angel, whose values are included for comparison reasons). Based on these values, 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.

Table 4.1. Some examples of singular values

Object	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6
Sphere	1.0	0.915580	0.493656	0.000002	0.000001	0.000001
Torus	1.0	0.859124	0.468501	0.312101	0.112534	0.073765
Cylinder[1]	1.0	0.950181	0.1443138	0.137475	0.000001	0.0
Angel	1.0	0.789099	0.647687	0.260372	0.192607	0.174466

[1] Disregarding the top and bottom of the cylinder.

4.2. Recovering one DOF

In this section, a method to recover the rotation components of one DOF is proposed, which means $\sigma_6 \approx 0$. The vector $\boldsymbol{v_6^t}$ is used to denote the corresponding values of σ_6 on matrix $\boldsymbol{V^T} = (\boldsymbol{v_1^t}\ \boldsymbol{v_2^t}\ \boldsymbol{v_3^t}\ \boldsymbol{v_4^t}\ \boldsymbol{v_5^t}\ \boldsymbol{v_6^t})^T$ and also represents the null space base of $\boldsymbol{J_{es_m}}$.

The method to recover one missing DOF is mainly based on the work developed by Kumagai [111]. The main difference is the method for setting the threshold value: In the previous work, it is based on the error between the 3D model and the target object, which is calculated in every frame. However, to use it with the quadrics representation, this calculation is not necessary, since instead of using the edges from the 3D model, the quadrics associated to each patch are used. Therefore, the threshold is set manually once, by checking the singular value using the method explained in Section 4.1.

If six DOF can be estimated from the contour information of the target object, the solution for the pose parameters exist in this null space. A one-dimensional search is performed in this null space to find the necessary values to recover the object DOF. Figure 4.2 illustrates the changes made in the framework shown in Figure 3.2, highlighted by the dashed rectangle: Test of the number of observable DOF and, when necessary, recovery of the missing DOF.

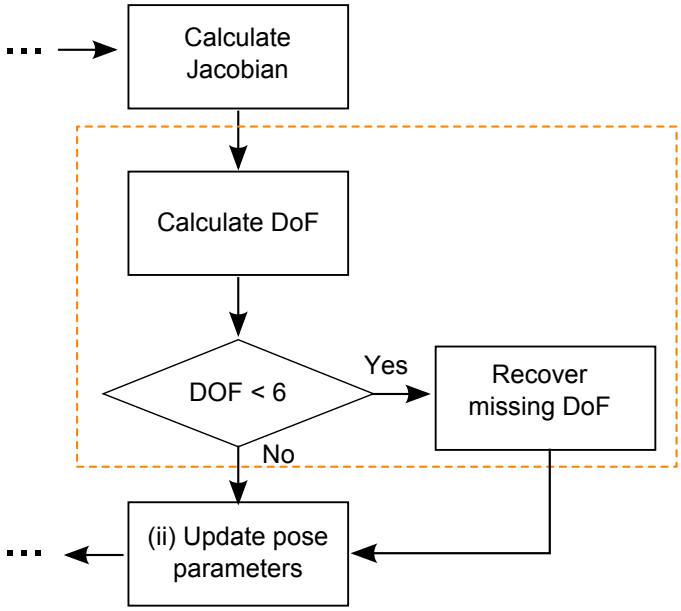


Figure 4.2. New modules added to the proposed framework, highlighted by the orange rectangle.

4.2.1 Rotation axis calculation

If the rotation is represented by a null space base, it means the model is rotated around an axis where full search of this space is performed. The position of this rotation axis $\boldsymbol{r_0}$ is calculated by using the vector $\boldsymbol{v_6^t} = (v_{61}\, v_{62}\, v_{63}\, v_{64}\, v_{65}\, v_{66})$, where $\boldsymbol{v_r^t} = (v_{61}\, v_{62}\, v_{63})$ represents the rotation and $\boldsymbol{v_t^t} = (v_{64}\, v_{65}\, v_{66})$ the translation parameters.

When the center of rotation does not pass through the origin, i.e. the directional vector of $\boldsymbol{r_0}$ does not match the coordinate system axis $\mathbf{C}$, the directional vector representing the null space is given by:

$$\boldsymbol{v^t} = (v_x\, v_y\, v_z) = \frac{\boldsymbol{v_r^t}}{|\boldsymbol{v_r^t}|} \tag{4.3}$$

The vector $l^t = (l_x\, l_y\, l_z)$ representing the distance from the center of $\mathbf{C}$ to $\boldsymbol{r_0}$ is calculated as:

$$\boldsymbol{l} = \frac{\boldsymbol{v} \times \boldsymbol{v'}}{|\boldsymbol{v} \times \boldsymbol{v'}|} \frac{|\boldsymbol{v'}|}{|\boldsymbol{v}|} \tag{4.4}$$

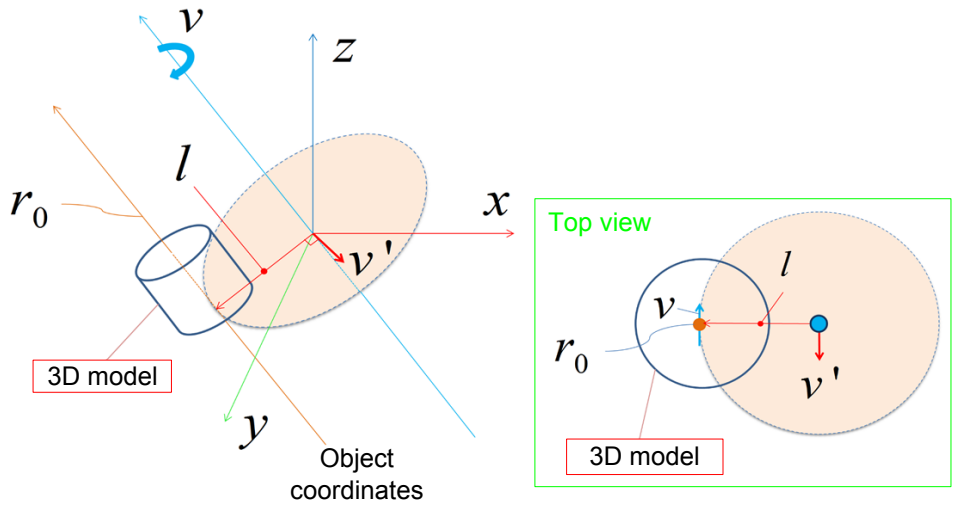


Figure 4.3. Rotation axis r_0 calculation [111, 31]

Finally, the six parameters representing the position of $\boldsymbol{r_0}$ on the image are calculated using the direction vector $\boldsymbol{v}$ with the distance vector $\boldsymbol{l}$:

$$\boldsymbol{r_0} = (\boldsymbol{v}\, \boldsymbol{l}) \tag{4.5}$$

4.2.2 Null space search

As mentioned previously, if the last singular value is below a threshold, it means that one DOF is missing, which triggers the recovery process. In this case, first the rotation axis is calculated from the null space represented by the vector $\boldsymbol{v_6^t}$. Next, the model is rotated with step of width h and recalculation of the pose parameters is performed. This step is performed once around the rotation axis representing the null space.

During the search process, the values that exceed the threshold are considered as possible solutions for the pose. The largest one among them is adopted as the sought pose parameters. If none of the values are larger than the threshold, the values used in the pose estimation of the previous frame is used as the result for the pose of the current frame. This process is illustrated in Figure 4.4.

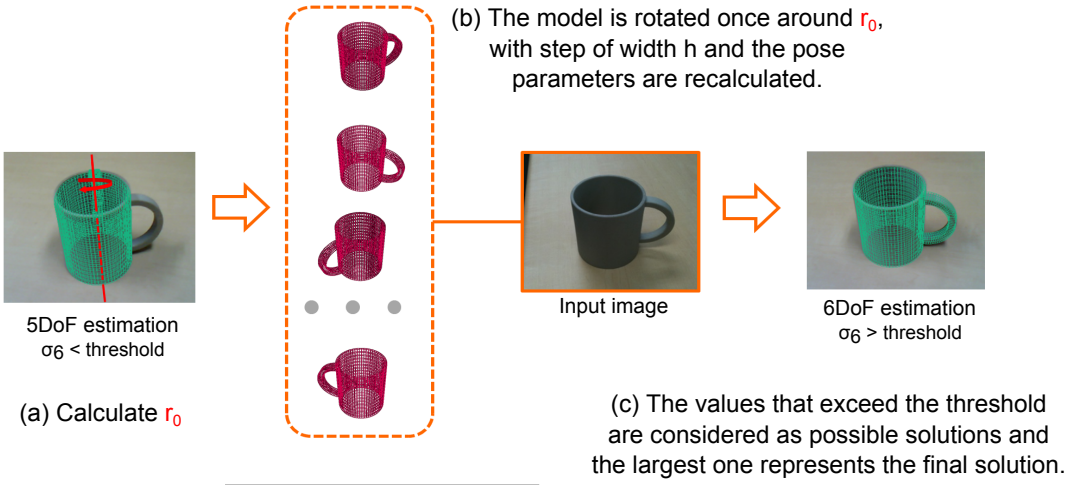


Figure 4.4. Null space search and the DOF recovery process.

4.3. Concluding remarks

This chapter describes a method to allow continuous tracking of objects with less than six observable DOF, and when the number of observable DOF of the object changes during the tracking sequence.

The number of observable DOF is calculated by counting the number of non-zero singular values obtained from the SVD of $\boldsymbol{J_{es}}$. Lastly, null space search is used to find one missing DOF.

Although the current implementation is able to correctly recover one missing DOF, increasing the recovery method for two or more missing DOF also increases the complexity of the implementation. Furthermore, this implementation is strongly affected by false positives caused by wrong edge detection, which affects the null space search. Experiments regarding this implementation are introduced in Chapter 5, Section 5.2.1).

CHAPTER 5

Experiments

Experimental results using our proposed framework are described in two scenarios: (a) with synthetic data generated by a simulator and (b) with video images from the real world. 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) against a standard Line Tracking approach using Sparse (SLT) and Dense (DLT) meshes.

5.1. Quantitative evaluation

In the quantitative evaluation, numerical results to quantify the tracking are obtained using a simulator [112], whose parameters can be easily controlled.

5.1.1 The simulator

In this simulator, three models of the target object in different levels of quality are accepted as input:

(a) A *high quality mesh*, which generates a synthetic representation of the real object and it is used in the edge search step.

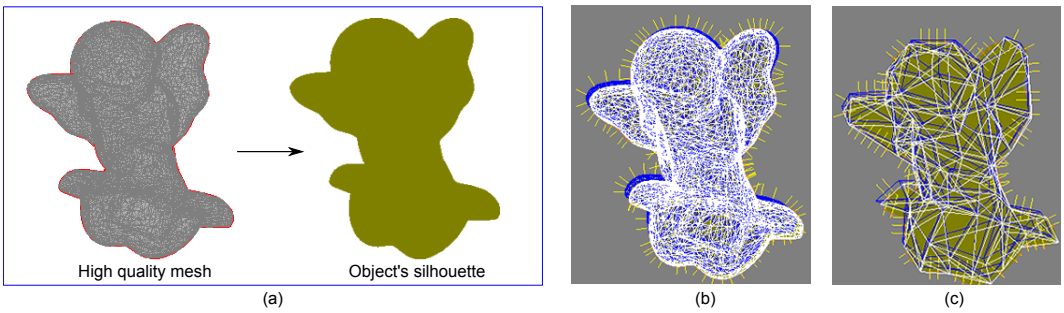


Figure 5.1. Simulator input: (a) high quality, (b) medium quality and (c) sparse mesh.

(b) A *medium quality mesh* used with DLT and having approximately 10% of the total number of patches of the mesh used in (a).

(c) A *sparse mesh* used to compare SLT and CT.

While the actual evaluation is done mainly between tracking using the mesh in (c), the high quality mesh in (a) is used only to create a realistic outline of the target object, as shown in Figure 5.1(a). In Figure 5.1(b) and Figure 5.1(c), the dense and the sparse mesh, respectively, are represented by the white lines and overlaid on the target object silhouette.

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. For instance, if the model is obtained using a 3D laser scanner, the final model usually has more than 100,000 patches. However, this amount of data is neither necessary nor computationally efficient. Therefore, to get the model used in (a), the original model is simply reduced until all redundant data is deleted and the smoothness of the object is not affected.

For (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 real-time. Since data of this polygonal mesh are used later to define the internal vertices (Chapter 3, Section 3.2.2.2) used in the conics fitting, the polygonal mesh is tested to ensure that all patches, or more than 95% 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 considered if the mesh is manually modeled using a 3D modeling software. Lastly, for item (c), the number of patches is chosen according to the experiment goal as described in the next sections.

The simulation consists of a series of different trials, each one representing the object in a different initial position P_0, manually initialized. Each trial has a number n of runs specified by the user and in the beginning of each run, the test model moves back to its initial position. Then, a new pose P_n is produced by the simulator using Gaussian noise, whose values are the same for all approaches being compared. The tracking approaches are evaluated by computing the camera displacement between P_0 and P_n. Another parameter that can be also controlled by the user is the presence of noise in the edge detection process. This noise aims to simulate similar noise detected when video images are taken from the real environment.

5.1.2 Experiments configuration

In our experiments, four objects with different shapes and complexity were considered. For each object, the simulation consisted of a series of 5 trials, each trial corresponding to the object in a different initial pose and having a total of 100 runs. The initial poses of each object are shown in Figure 5.2. The mesh size of the four different objects' models (torus, duck, angel and bunny) used in this experiment can be found in Table 5.1.

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. 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.

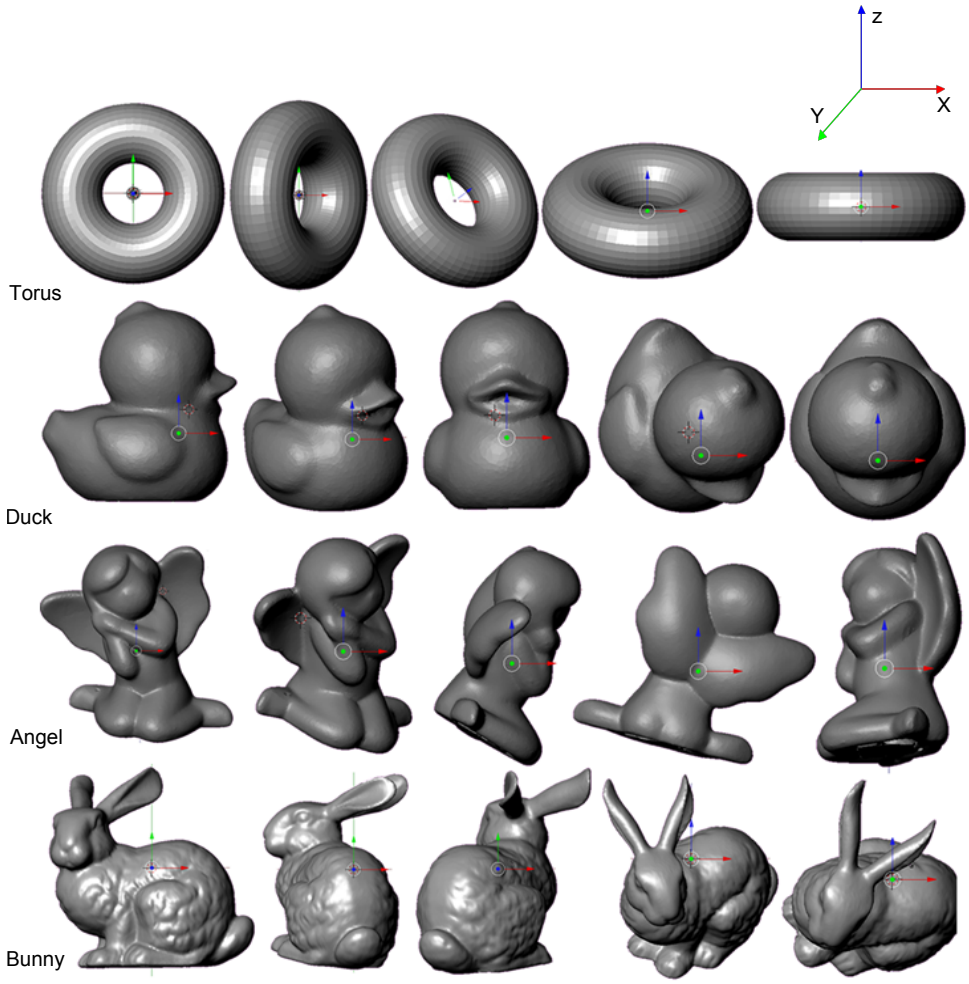


Figure 5.2. Poses used in the quantitative evaluation. Each pose was tried 100 times with different noise values.

Table 5.1. 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)

5.1.2.1 Experiment I

In this first experiment, the main goal is to compare CT and SLT performance by varying the number of patches in the polygonal mesh. This range is defined according to the object complexity: Since the torus has a simple shape, a range from 500 to 50 patches is used; for the duck, 750 to 75 patches, and for the angel and the bunny, a range of 1,000 to 100 patches. The noise values used to generate the poses have standard deviation of $\sigma_{a1} = 1.0^\circ$ 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.

For each object, the following items are evaluated:

- *Accuracy*: Mean Squared Error (MSE) of the rotation and the translation components of the pose parameters vector s.
- *Computational time*: average processing time.
- *Robustness*: success rate in 100 runs, based on the error value returned by the cost function.

The graphs in Figure 5.3 show the accuracy results obtained for all objects in the 5 poses according to the number of patches of the polygonal mesh. Graphs (a)(c)(e)(g) summarize the error results for the rotation component in degrees. They are obtained by calculating the angle θ between the vector $\vec{r}_0$ (initial rotation parameters given to the simulator) and the vector $\vec{r}_s$ (rotation parameters after the optimization process).

When the number of patches is high, SLT performs better than CT. This happens because for dense meshes, the number of valid quadrics that can be used for tracking is low, as shown in Table 5.2. The problem becomes worse if for a certain viewpoint, most of them are located on the contour. For sparse meshes, the number of quadrics is close to the number of patches, which improves the results. As the number of patches decreases, SLT performance decreases and at a certain point, the MSE of CT gets better results. Similar behavior can be seen in graphs (b)(d)(f)(h) on the right, with the results for the translation components (in mm): the MSE for CT decreases as the number of patches decreases.

Furthermore, while the error of SLT increases with the decrease of the number of patches, the error of CT decreases up to an optimal point where the error starts to increase again (though with values still lower than SLT). This behavior can be explained by analyzing the simplified meshes of the target object, illustrated in Figure 5.4.

Some problems caused by overdoing the polygonal simplification include: Changes in the object shape (the hole in the middle of the torus when it has less than 150 patches) or deletion of important parts of the object (the beak of the duck when it has 50 patches and part of the wing and the leg of the angel when it has 100 patches). All these problems affect the quadrics fitting and, as a consequence, the tracking accuracy. 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 mesh simplification process.

Figure 5.5 (a)(c)(e)(g) shows the average of the success rate in 100 runs, with CT having low results for dense meshes, but performing better than SLT when the number of patches decreases, as expected.

Regarding computational time, the average of the results of each method is shown in Figure 5.5(b)(d)(f)(h). Among the three methods, CT has the best results. For simple shapes (torus and duck), the computational 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.

Table 5.2. Number of valid quadrics according to the number of patches

Torus		**Duck**		**Angel**		**Bunny**	
Patches	Quadrics	Patches	Quadrics	Patches	Quadrics	Patches	Quadrics
500	228 (45.6%)	750	679 (90.53%)	1000	708 (70.8%)	1000	694 (69.4%)
250	237 (94.8%)	500	480 (96%)	750	670 (89.33%)	750	616 (82.13%)
150	149 (99.33%)	250	239 (95.6%)	500	480 (96%)	500	428 (85.6%)
100	94 (94%)	150	140 (93.33%)	250	239 (95.6%)	250	215 (86%)
70	54 (77.14%)	100	87 (87%)	150	130 (86.66%)	150	110 (73.33%)
50	34 (68%)	75	63 (84%)	100	71 (71%)	100	67 (67%)

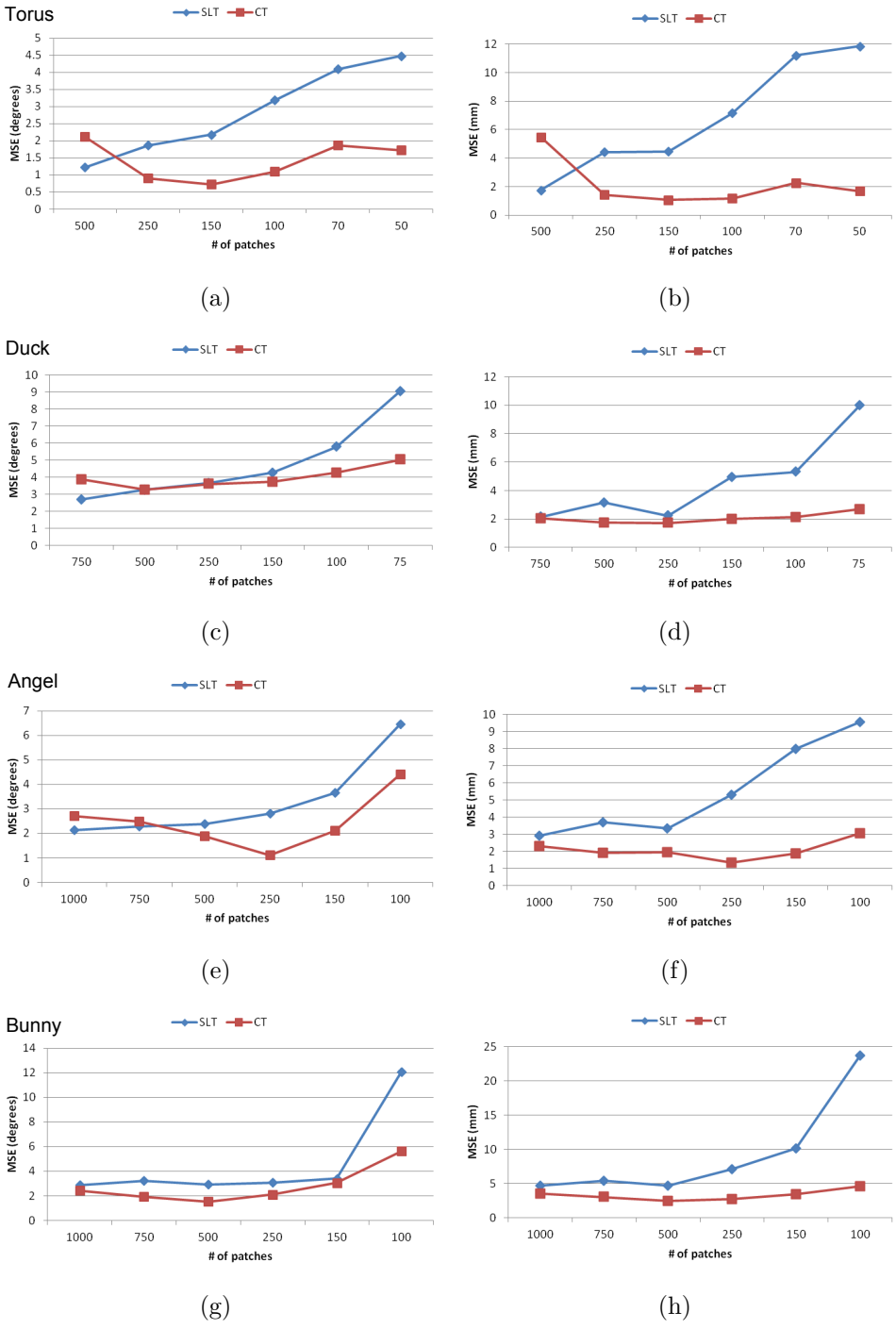


Figure 5.3. Experiment I: results for accuracy, where each row represents one object: torus, duck, angel and bunny, in this order. **Left**: MSE of the rotation in degrees. **Right**: MSE of the translation in mm.

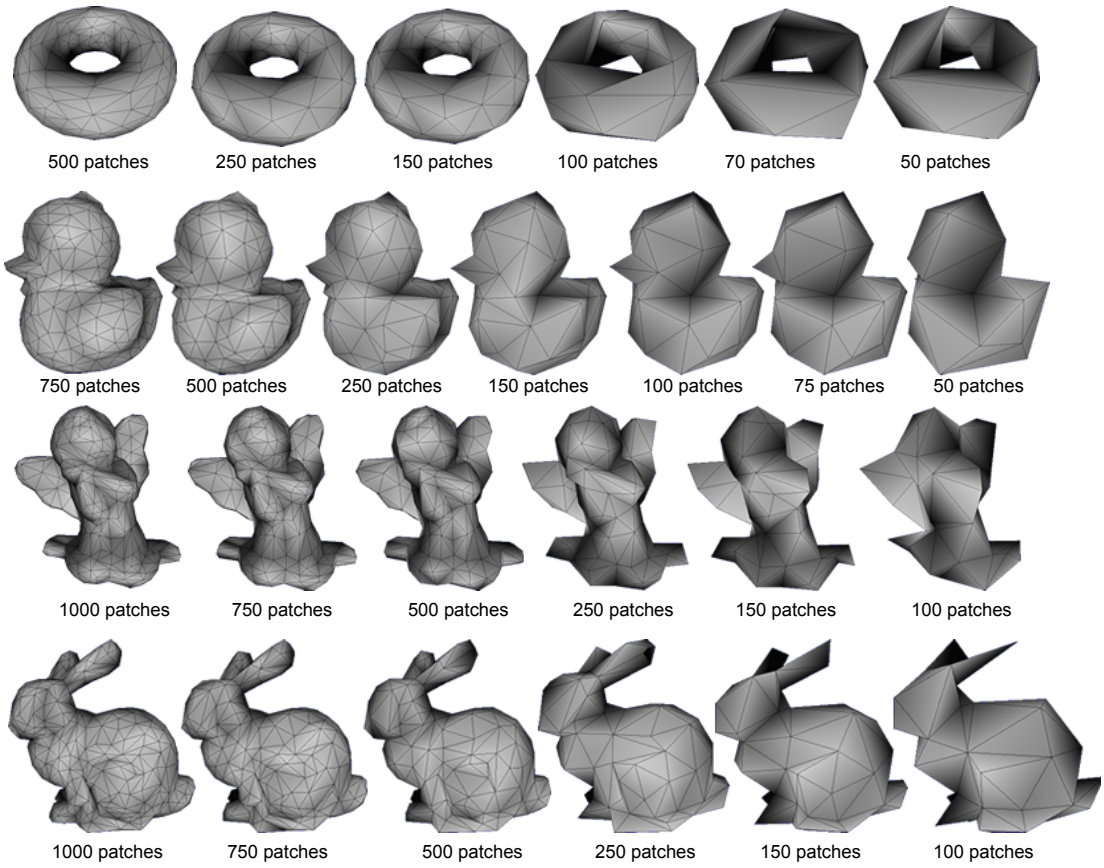


Figure 5.4. Polygonal meshes in different levels of details: When the remeshing is overdone, accuracy of CT becomes lower because of changes in the original shape of the object.

On the other hand, CT performs well in both accuracy (up to the optimal point) and computational time. For all patches, CT has lower computational time, even though more tests are performed inside the tracking loop. This is because the conic shape is closer to the actual object's contour, which makes the convergence of CT to the right pose parameter values faster than SLT.

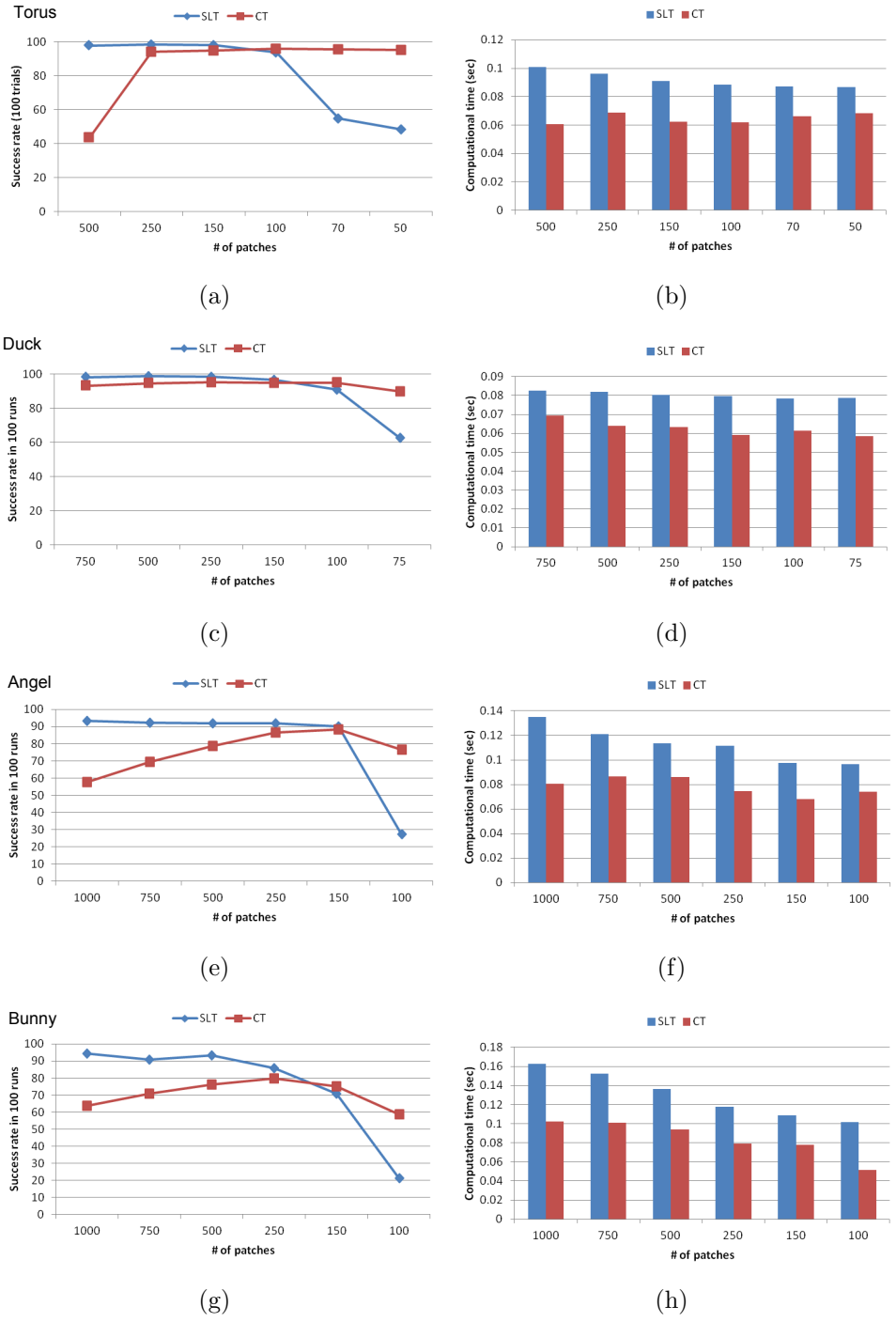


Figure 5.5. Experiment I: results for computational time and success rate in 100 runs, where each row represents one object: duck, angel and bunny, in this order. **Left**: Success rate. **Right**: Computational time.

5.1.2.2 Experiment II

In this experiment, the polygonal mesh has a fixed number of patches and the simulator parameters are changed according to the following conditions:

(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 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 is chosen after analysis of graphs in Figure 5.3 and Figure 5.5. They basically represent the point where CT had better performance. Thus, the following number of patches were chosen: 150, 150, 250 and 250, for the torus, duck, angel and bunny, respectively. Graphs in Figure 5.6 and Figure 5.7 show the success rate and accuracy results, respectively, 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 (graphs (a)(c)(e) and (b)(d)(f)). This is consistent with Table 5.3, in which the average of the re-projection error for each condition 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 5.7 with graphs (c) and (e) of Figure 5.5, the 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 was 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 graphs (b)(d)(f)(h) of Figure 5.5. DLT results are also presented in Table 5.3 for comparison purposes, showing CT can perform better than DLT in some cases. This confirms the assumption that our proposed tracking system using sparse polygonal meshes and conics can be as accurate as DLT, with the advantage that it consumes less computational time.

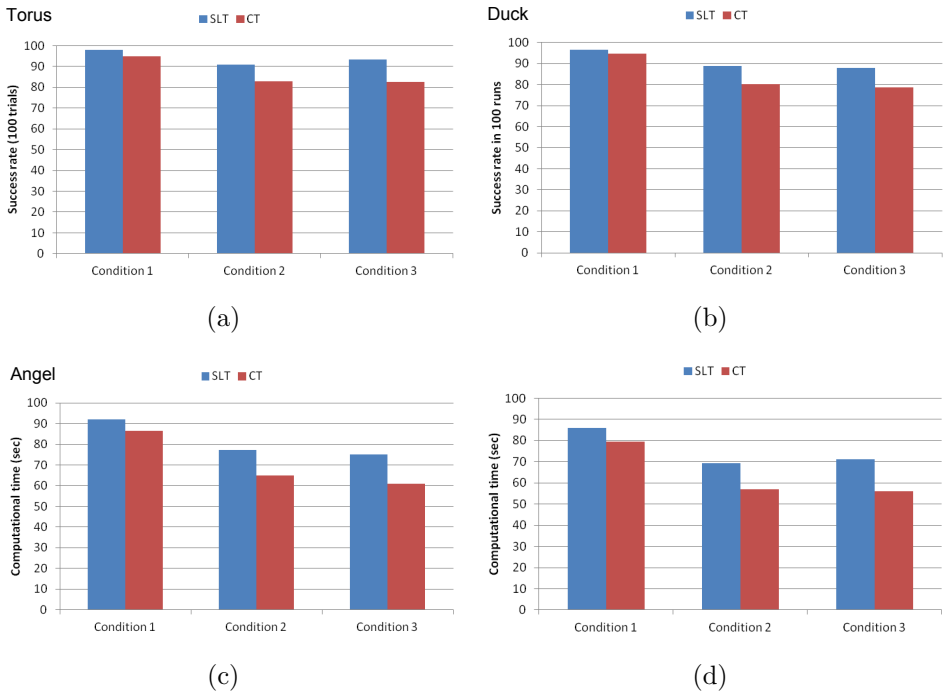


Figure 5.6. Experiment II: results for success rate.

Table 5.3. Re-projection error average (mm)

Object	**Conditions**	**CLT**	**SLT**	**DLT**
Torus	Condition 1	0.875738	1.22747	0.912317
	Condition 2	0.881122	1.283912	0.95264
	Condition 3	0.887358	1.272762	0.959054
Duck	Condition 1	0.913658	1.40086	0.892352
	Condition 2	0.9288624	1.439328	0.9640726
	Condition 3	0.9251336	1.490208	0.9994832
Angel	Condition 1	0.863258	1.432444	0.9307736
	Condition 2	0.8701316	1.454418	1.0154104
	Condition 3	0.9031862	1.499616	1.0396562
Bunny	Condition 1	0.9950724	1.579148	0.964695
	Condition 2	1.0593286	1.63444	1.0283068
	Condition 3	1.0409218	1.678012	1.0490236

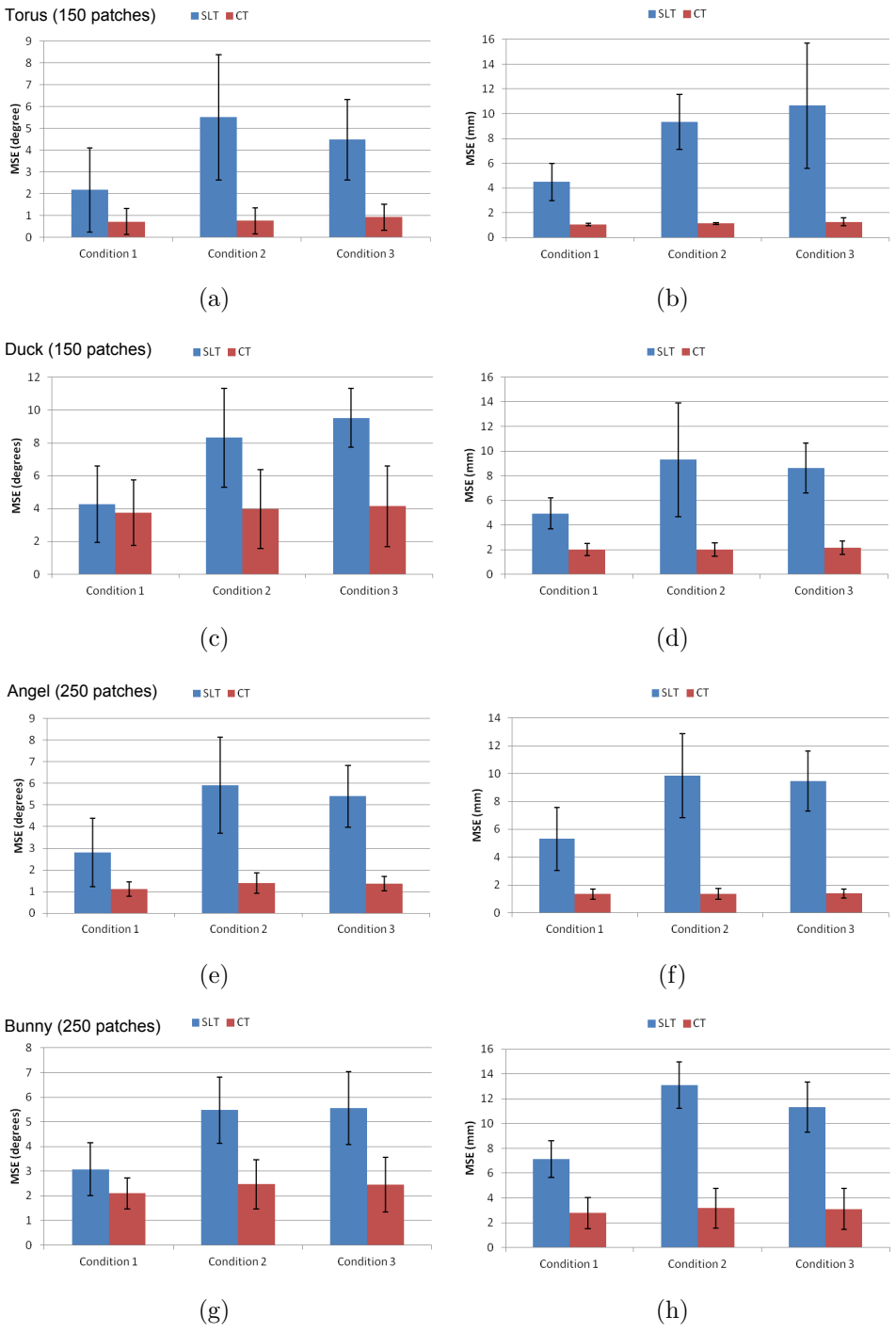


Figure 5.7. Experiment II: results for accuracy. **Left**: MSE for the rotation in degrees. **Right**: MSE for translation in mm.

5.1.2.3 Experiment III

Regarding objects having different number of observable DOF presented in Chapter **??**, a synthetic experiment was also implemented to evaluate the accuracy of the rotation axis $\boldsymbol{r_0}$. Evaluation was done using the simulator in a series of 100 trials with a cylinder having 64 patches (five DOF). The input images for this simulation were generated by adding Gaussian noise to the ground truth values. The mean values of the noise for translation and rotation were set to zero, with standard deviation of $\sigma_{c1} = 0.5cm$ and $\sigma_{c2} = 1°$, respectively. The following conditions were tested (Figure 5.8), where **C** represents the coordinate system:

1. **Condition 1**: r_0 passes in the origin of **C** and its direction is also consistent with the direction of **C**.
2. **Condition 2**: r_0 passes in the origin of **C** but its direction is not consistent with the direction of **C**.
3. **Condition 3**: r_0 does not pass in the origin of **C** and its direction is consistent with the direction of **C**.
4. **Condition 4**: r_0 does not pass in the origin of **C** but its direction is not consistent with the direction of **C**.

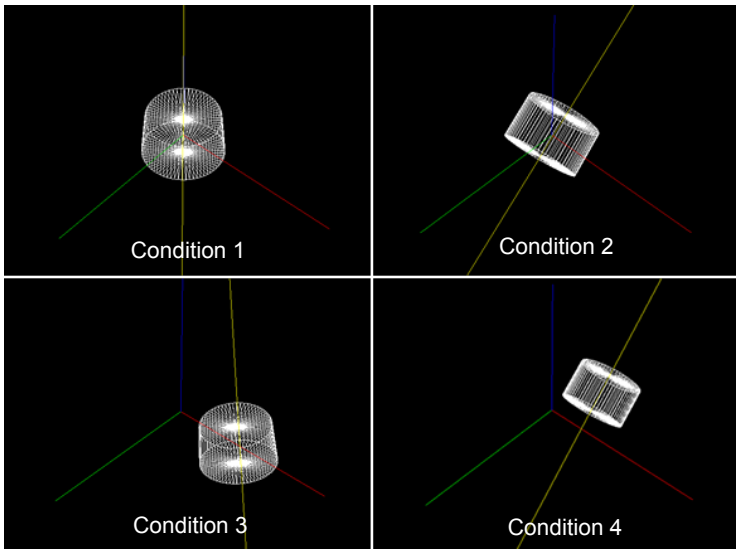


Figure 5.8. Conditions to evaluate $\boldsymbol{r_0}$ [113, 31]

Results of the average error of this estimation are shown in Figure 5.9, validating all evaluated conditions. However, it was noticed that the error in vector $\boldsymbol{l}$ increases according to the distance of $\boldsymbol{r_0}$ from the origin of $\mathbf{C}$. Hence, if the distance that separates their origin is big, the error in estimation also increases.

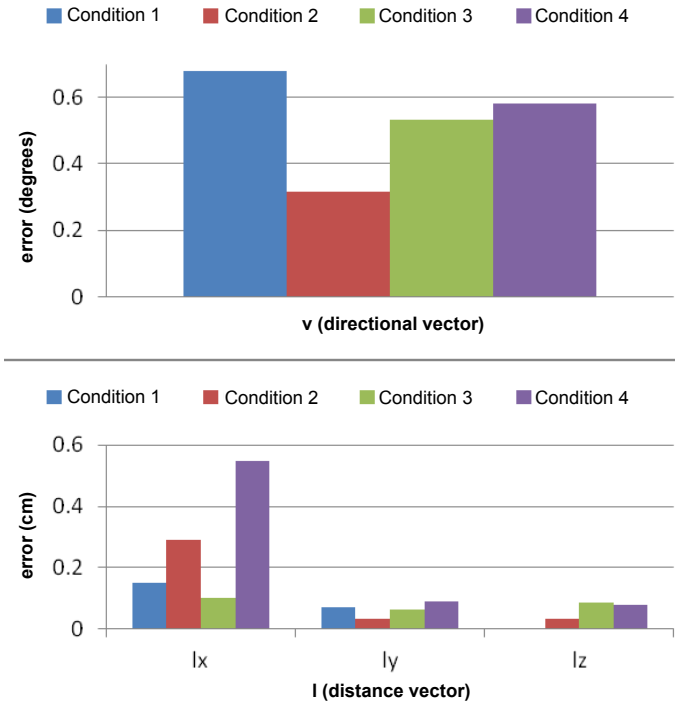


Figure 5.9. Average error for all conditions: (a) error for the directional vector and (b) error for the distance vector [113, 31].

5.2. Qualitative evaluation

Experiments in the previous section provide 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. For this evaluation, a handheld monocular camera Logitech QuickCam Fusion was chosen, with resolution of 640x480 pixels.

To ensure a fair comparison, CT and SLT are tested using the same video sequence, recorded beforehand with frame rate of 30fps. Three conditions are evaluated:

(a) the user holds the object and moves it arbitrarily;

(b) the object is fixed, while the camera is moving;

(c) tracking in a cluttered background.

To visualize the tracking, the polygonal mesh is rendered on the object's surface: when tracking succeeds, the mesh is green; if it fails, its color changes to pink. The performance is finally evaluated by counting the number of frames in which the tracking fails. Two objects were tested in this first section: torus (simple shape) and angel (complex shape). Figure 5.10 and Figure 5.11(a) show the results when the object is being moved by the user, while in Figure 5.11(b), the camera is moving around the object. For all sequences, the upper images show CT performing better than SLT, without any failures.

Tracking in a cluttered background is shown in Figure 5.12. As expected, the number of failures increased for both approaches, due to wrong edge detection caused by the objects and texture present in the background. Although CT had lower number of failures compared to SLT, most of the frames where they failed were the same.

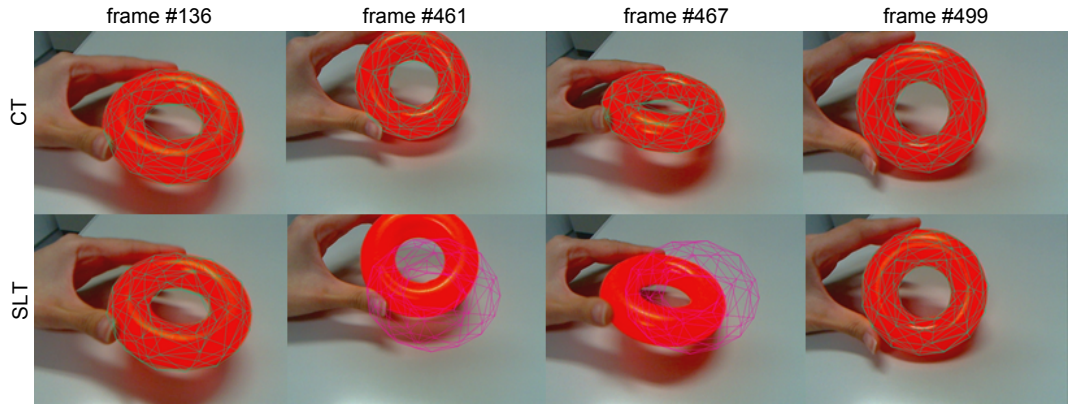


Figure 5.10. Qualitative evaluation for the torus. In the upper sequence, CT does not present any failure. In the bottom sequence, SLT fails in some poses.

Table 5.4 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.

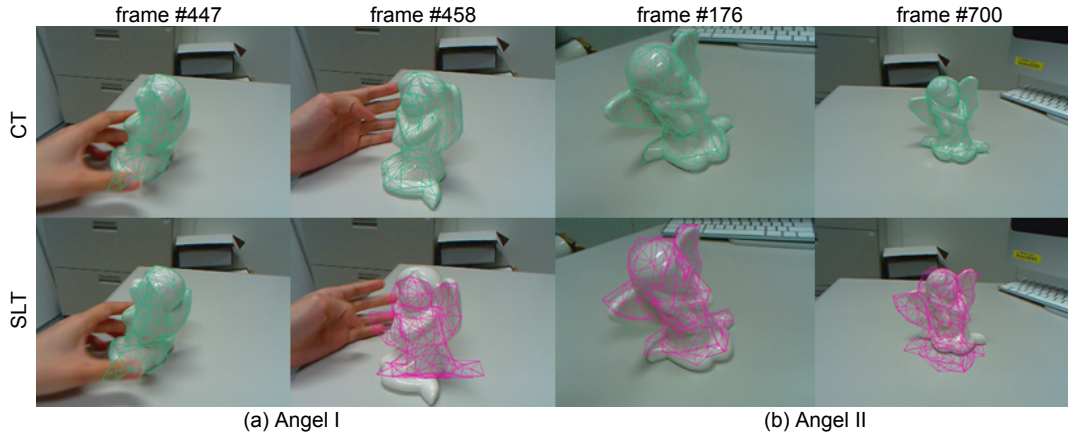


Figure 5.11. Qualitative evaluation for the angel. (a) Angel I: The user moves the object (two images on the left). (b) Angel II: The camera moves around the object (two images on the right). In the upper sequence, CT does not present any failure. In the bottom sequence, SLT fails in some poses.

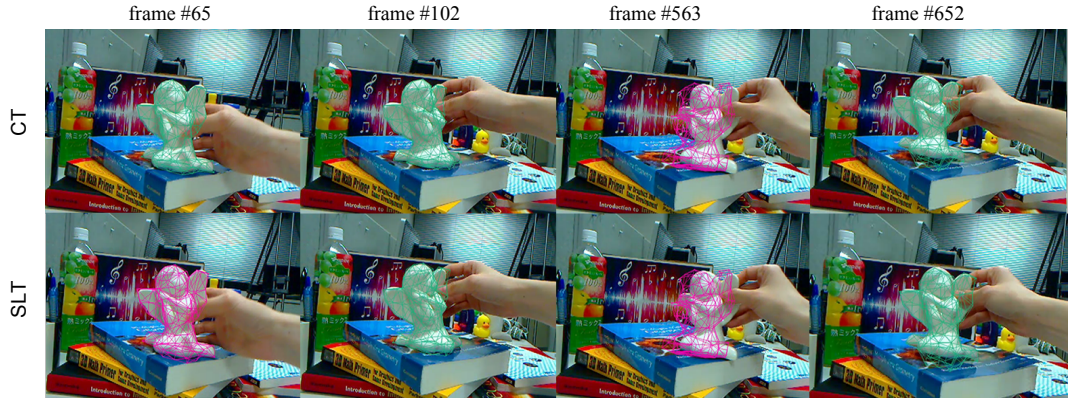


Figure 5.12. Angel III: Tracking evaluation in a cluttered background.

Table 5.4. Number of Failures

Object	# of Frames	CT failures	SLT failures
Torus	605	- (0%)	9 ($\approx$ 1.48%)
Angel I	749	- (0%)	89($\approx$ 11.88%)
Angel II	702	- (0%)	532($\approx$ 75.78%)
Angel III	746	173 (23.19%)	213 (28.55%)

5.2.1 Objects having different number of observable DOF

Experiments using video sequences were also performed for objects having different number of observable DOF, described in Chapter 4.

In the upper sequence of Figure 5.13, the mug is being successfully tracked using the proposed approach. The handle, not visible in the beginning, appears correctly tracked at the end. In the bottom images, the same result is not achieved using previous methods. However, the experiments in this case are considered using standard-edge based tracking [31], that is, the mug model does not use the quadrics definition presented in our framework. One reason is because the top and the bottom of the mug are non-smooth patches and cannot be properly handled by the quadrics patch representation, but the results are presented in this chapter for the validation of the proposed idea.

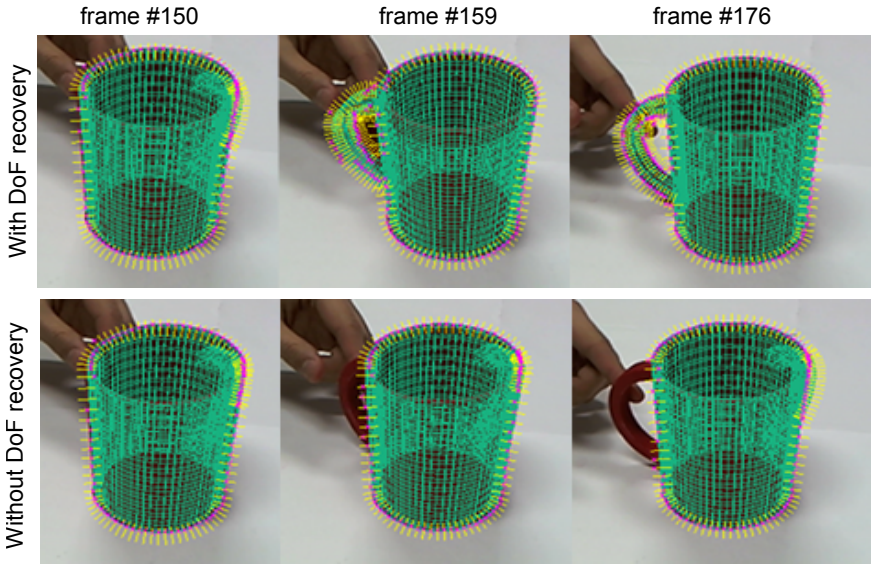


Figure 5.13. Tracking a mug with and without recovery [113, 31]

For the teapot, the quadrics patch representation was used and experiments were performed to illustrate tracking recovery after the main body of the object was framed out. In Figure 5.14 (a), the proposed method with recovery and in Figure 5.14 (b) the previous method without recovery. In this sequence, from top to bottom, the teapot in the beginning is entirely visualized in the scene. After only the spout was framed in and the whole teapot is visible to the camera again, the model is not correctly aligned to the object in (b), but remains aligned in (a).

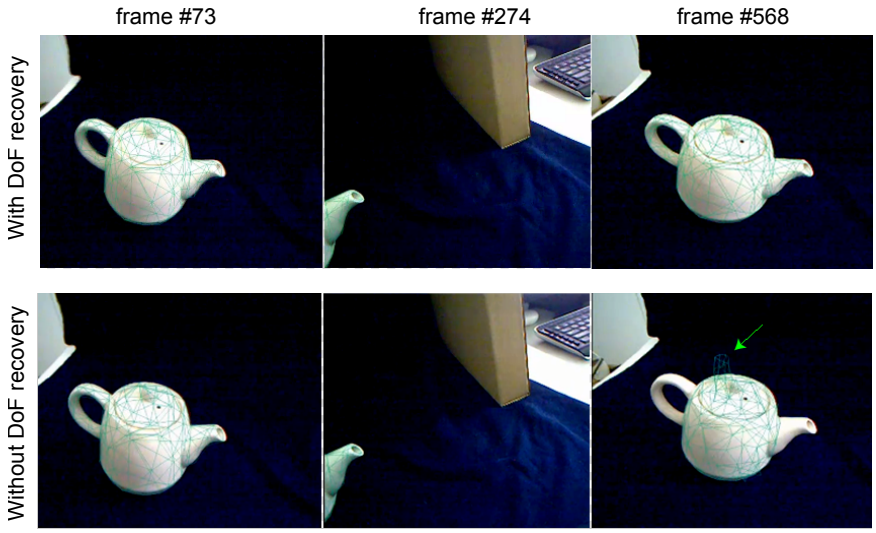


Figure 5.14. Tracking a teapot with and without recovery.

5.2.2 Limitations and failure cases

As previously mentioned, the proposed method is sensitive to cluttered background due to wrong edge detection. Other failure cases include high inter-frame motion (Figure 5.15) and large occlusion of the object, though in both cases tracking is recovered after few frames and partial occlusion can still be handled (Figure 5.16).

For the recovery method, failure can happen when search in null space is performed on the wrong rotation axis (Figure 5.17). To prevent these failures, a strategy is necessary to avoid wrong edge detection of the object contour.

More details about all video sequences used in this section can be found in Appendix A.

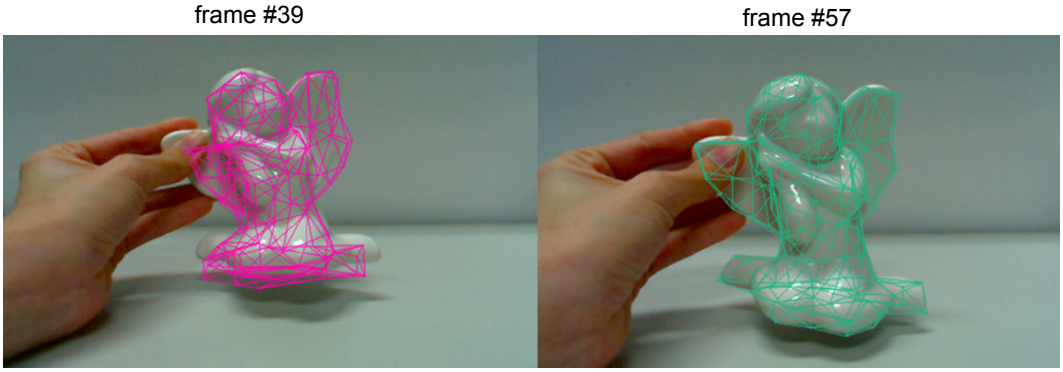


Figure 5.15. Failure caused by large inter-frame motion and tracking being automatically recovered after some frames.

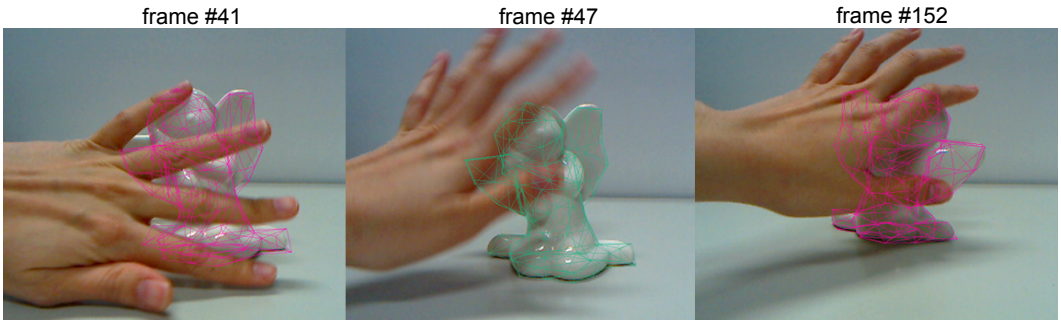


Figure 5.16. Failure caused by large occlusion of the target object in the second image. In the other images, partial occlusion can be handled.

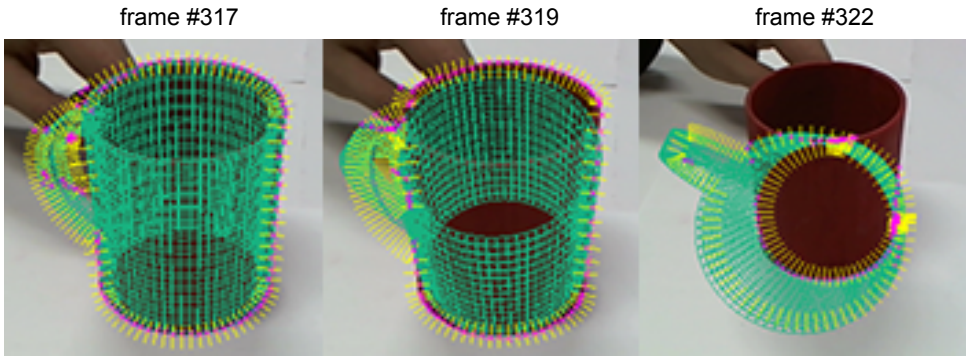


Figure 5.17. Failure caused by null space search in the wrong rotation axis [113, 31].

5.2.3 Concluding remarks

In this chapter, a description of the experimental results performed in both synthetic and in the real environment are presented. 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 how accuracy, speed and success rate of each method are affected. 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. In both scenarios, CT is able to perform better than SLT in most of the cases, validating the proposed approach.

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. Further results are also included in the next chapter, with an augmented prototyping application developed using the proposed framework.

CHAPTER 6

Augmented Prototyping

This chapter presents an application of the proposed framework for augmented prototyping of curved objects. By enhancing physical prototypes with AR, evaluation of its design and aesthetic concepts in real-time becomes easier, saving time and production costs. Compared to previous approaches using AR to assist design evaluation, the method proposed in this chapter has an additional advantage by considering the environment illumination effects on the virtual textures. Results from a pilot user study comparing the use of a 3D software and the proposed application are also presented.

6.1. Rapid Prototyping

In a typical design process, after ideas are discussed and represented through sketches, a 3D model of the product can be constructed for better visualization. Rapid Prototyping (RP) adds one step forward improving design testing by automatically constructing physical prototypes using, for instance, a 3D printer. Prototypes have different usages: functionality and physical testing, visual evaluation, marketing, and proof-of-concept [114], as well as facilitate communication among designers or between designers and clients.

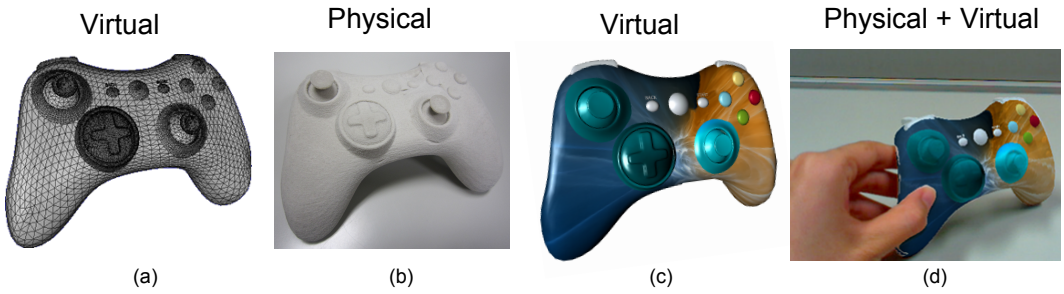


Figure 6.1. Prototype representations: (a) 3D model constructed in early stages of the design process, (b) non-textured physical prototype, (c) textured 3D model, and (d) the augmented prototype, obtained by combining (b) and (c).

However, RP is still time consuming, with some models taking several hours to be completely done. For instance, Figure 6.1 (b) shows a game controller [1] prototype (width = 13cm, height = 2.5cm, depth = 8cm) built using a 3D printer ZPrinter 450. It required approximately four hours to print, one night to be completed dry and around one hour for the final touch.

Using standard RP, the product's aesthetic is probably the hardest item to be evaluated, since it is trial-and-error basis. For instance, consider the scenario where a designer wants to evaluate different textures in a new product. Current 3D printers can print colored and textured prototypes, but producing new prototypes to evaluate each design is not worth the time and cost, specially for complex shapes. However, with AR 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.

6.2. Related work

During conception and realization of a new product, in general some kind of model is created to attend as close as possible the product requirements. This model can be represented by sketches, virtual models or rough physical models, which are valuable for design evaluation and do not require a complex manufacturing process. However, when these representations are used alone, some important visualization properties can be overlooked, leading to unsatisfactory results.

[1] 3D model available at http://www.blendswap.com/

Therefore, the use of prototypes in the form of physical models play an important role in this process, allowing designers to experience the shape details, compositions, functionality and improve the final design in a relative fast and less expensive way [114]. These physical models can be constructed using RP, but regarding evaluation of aesthetic features, such as color, texture and materials, they may not be the ideal solution because these features cannot be easily changed in the physical prototype by traditional means, yet they represent important features to the final product evaluation. In this sense, virtual and augmented reality represent promising technologies to overcome RP shortcomings, as it is presented in details in the next sections.

6.2.1 Virtual Prototyping

Virtual Prototyping (VP) uses digital models or virtual prototypes in a computational environment to analyze and validate a product design before physical fabrication. It can reduce the number of physical iterations and therefore the associated manufacturing overheads, leading to as faster and cost-effective product development [115, 116]. Common fields using Virtual Prototyping (VP) are aerospace industries [117] and automotive design [118].

Virtual Reality (VR) is a common technology used in VP. However, although it offers the possibility to reduce production costs when compared to RP, there are still some drawbacks; for instance, physically touching and manipulating the prototypes require expensive devices such as VR gloves or haptic devices. This is specially necessary when evaluating ergonomics, usability, physical interaction and response [119]. With the advent of AR, simplification and speed-up of the design process became possible, without the need of special devices.

6.2.2 Augmented Prototyping

Augmented Prototyping (AP) employs AR technology to combine virtual and physical prototypes [120]. Compared to VP, some advantages include: Evaluation of aesthetic and ergonomic aspects using a single physical prototype; interaction in real-time without using special haptic devices and no need to simulate details of the target scenario, e.g. illumination.

Benefits of AR for design model perception of users and collaborating partners are discussed by Dunston et al. [121]. Although in their system no physical prototypes are constructed and fiducial markers are used to position the 3D model in the users' environment, it shows the importance of considering spatial cognition issues during systems validation.

Lee & Park [122] suggests an augmented foam as an approach of tangible AR for product design. Foam mock-ups is used because they are easily modifiable compared to the hard materials used in RP and can be produced in a short period of time. Fiducial markers are used to register virtual objects on the mock-up, with hand occlusion correction, so that the virtual object and the users' hand are seamlessly synthesized. Tangible AP using markers is further explored by Park et al. [123] for handheld products development. AR-based tangible interaction is combined with functional behavior simulation using two tangible objects: The product's prototype and a pointer to be used as an interaction tool, both of them having fiducial markers placed on it to augment the real world image.

Although these systems represent good scenarios of what can be achieved by AP, using fiducial markers brings some disadvantages, as previously mentioned in Chapter 2: Markers are intrusive elements in the environment, specially if the texture to be applied on the object has some transparency; partial occlusion of the marker may cause tracking failures, and depending on the object shape or size, it is difficult to place the markers or they may occlude important parts of the prototype. Approaches for hiding the marker in the video stream exist (Diminished Reality), but they can only be applied for planar targets [124, 125]. More recently, Herling & Broll [126] proposed an approach that can target almost planar backgrounds and also non-planar backgrounds. However, for the last case, the camera needs to remain static or it is limited to small movements.

Spatial AR represents another approach explored for augmentation in RP. Verlinden & Horvath [120] and Porter et al. [127] proposes augmenting the prototypes using digital projectors to display the virtual information on the object surface. The main advantages are that HMD or handheld devices are not required and collaboration between the users can be easily performed. On the other hand, some drawbacks of this approach includes the shadows that are cast onto the objects by the user and problems inherent with attempting to use camera based tracking where the object being tracked is also being projected onto.

More recently, Gay-Bellile et al. [128] developed an augmented prototyping application for the automotive field using constrained SLAM with an edge-based model [75]. In their application, users can interactively customize a real car and see different designs overlaid on it using a tablet. It is an accurate and stable approach, but the appearance of the car needs to be learned in an offline step to initialize the constrained SLAM. Using only point-based model with relocalization proved not to be enough for the non-textured car. Therefore, if the original position of the car changes, the environment needs to be relearned.

6.3. Proposed AP application

The AP application proposed in this chapter aims at create a system for fast evaluation of the product's aesthetic by a designer. Different appearances of the target object can be created and overlaid on the physical prototype in real-time, making visual comparison an easier task. The 3D model of the target object used to build the prototype is also used for matching with the edge information found on the video image, fitting well the framework proposed in Chapter 3.

A similar approach aiming at reducing development time during the product design process was previously developed by Canon Inc. The Canon Mixed Reality system [129] is a marker-based system, which also supports optical and magnetic-type sensors based on the requirements of the application. The prototype has a blue color to deal with the hand occlusion and different designs can be overlaid on real-time on the prototype's surface. However, since this approach uses markers, it cannot handle the use of transparent textures and the effects of the environment illumination on the virtual design are not considered either.

In the implementation proposed in this chapter, two main points are considered: How to correctly render the virtual texture on the target object's surface, considering its material properties and the effects of the environment illumination, and occlusion of the texture by the user's hand or other objects.

Figure 6.2 shows an example of the angel figurine, where the user can easily modify the object color and texture and see how they fit to the prototype in a given scenario. It is possible to notice correctly handling of transparent textures and the hand's shadow on the object.

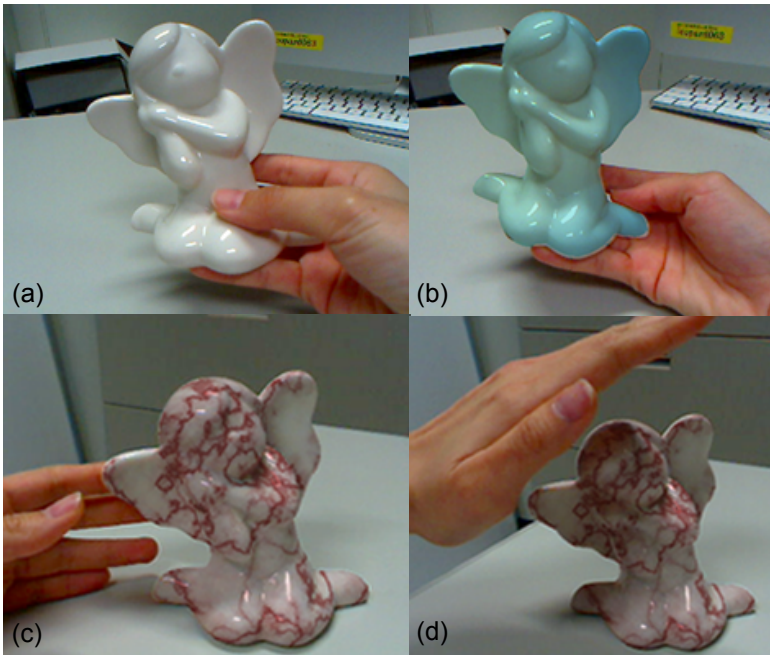


Figure 6.2. Augmented angel figurine. (a) Non-textured prototype. Different colors (b) and textures (b) can be tested using one physical prototype. (c) Correct rendering of the texture considering the shadows from the hand.

6.3.1 Texture composition algorithm

For realistic rendering of the virtual texture, OpenGL is used with a 2-path algorithm responsible to combine the texture's color with the object's color, where:

(a) *Path 1*: update the depth buffer for the target object without updating the color buffer.

(b) *Path 2*: update the color buffer if the Z-Buffer value is equal to the fragment z value. The following equation is used for the texture blending:

$$outputColor = frameBufferColor * textureColor. \tag{6.1}$$

This 2-path approach was adopted because the texture blending has to be applied only for the most front surface patches. If standard Z-buffer rendering is used, this texture blending would be applied for several patches which are not located on the most front surface, if they are rendered before the most front

surface patches. Theoretically, this implementation would work only for diffuse objects; however, as shown in Figure 6.2, no problems were identified when using a specular object, resulting in a realistic texture blending.

6.3.2 Handling the texture occlusion

As shown in Figure 6.3, the occlusion of the object texture is inevitably caused by the hand or another object existent on the scene, which makes user's interaction unnatural. To correctly handle the occlusion of the texture, HSV was used for color segmentation. Each HSV channel was thresholded and combined into a mask used to distinguish the object shape from other elements in the scene. Since the hand color is hard to be segmented because of different skin color tones, the prototype's color was used (printed with a color close to white). Hence, it became easier to test with different users without thresholding the color segmentation for each one.

Figure 6.3. Texture occlusion on the (a) game controller prototype and the (c) angel figurine. On the left, the object texture is rendered on the user's hand. On the right, correct display of the texture by removing it from the occluded parts.

6.4. User study

A pilot user study comparing our proposed AP application against a 3D modeling software (Blender version 2.49) was designed to evaluate the perception of the user when visualizing different textures on a prototype.

6.4.1 Participants

A total of 10 graduate school students, 7 male and 3 female, ranging in age from 25 to 37 years old, participated in this study. 6 participants declared not having previous experience with any kind of 3D modeling software but all of them have tried AR before.

6.4.2 Physical setup

The machine used for the tests was a Corei7 3.20Ghz, with 8GB of RAM and NVIDIA GeForce GTX 560Ti. For the experiment with the 3D modeling software, a standard PC desktop interface with monitor, keyboard, and mouse were used.

For the AP application, a special HMD was built, using two Point Grey Dragonfly cameras attached to a Sony Personal 3D viewer headset, HMZ-T2 (Figure 6.5). A resolution of 1024x768 was used for the video images, achieving a frame rate of approximately 15fps.

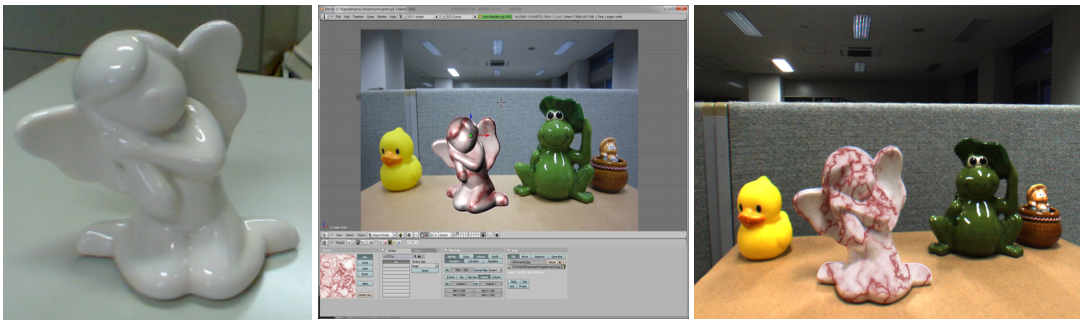


Figure 6.4. Prototype used in the AP evaluation: (a) Angel figurine. (b) Blender interface. (c) View from the HMD in the augmented environment.

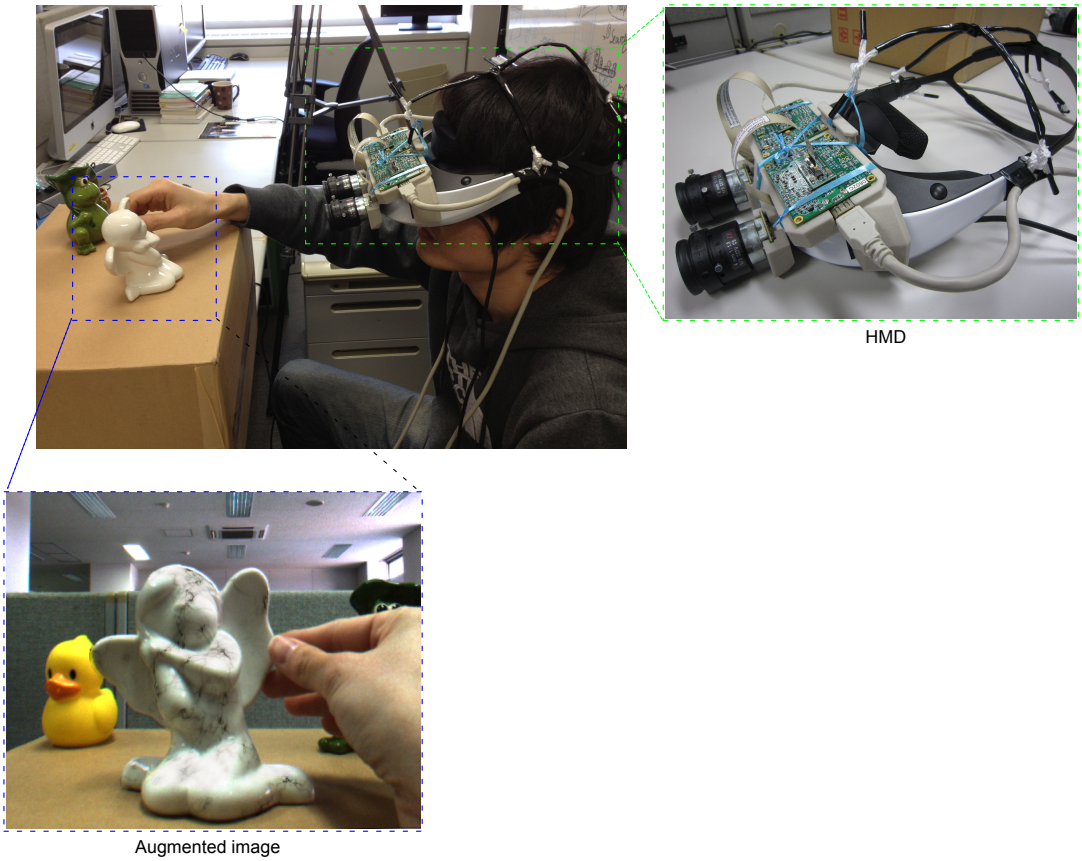


Figure 6.5. User interacting with the AP system. Highlighted by the colored boxes, the HMD constructed for this experiment and the user's augmented view.

6.4.3 Task description

This experiment was divided in two sessions, one for each of the conditions (3D modeling software and AP application), and the participants were asked to complete the same task in each session. The order for testing each condition was also changed to reduce order effects (e.g. learning or fatigue). After each session, participants answered a questionnaire whose contents are shown in Table6.1.

The object used in the experiments was a prearranged non-textured small angel figurine. Participants were asked to play the role of designers developing a new skin to this angel figurine. Finally, they had to choose which of the available designs, in their opinion, represented the best match with the other objects in the scenario (Figure 6.4).

Three designs were available (Figure 6.6) and the participants could browse among them as many times as they wanted. An explanatory and training session was given before each condition and the users' actions were observed during the entire session.

For the 3D modeling software evaluation, the scenario was constructed by taking a picture of the real environment and using it in the 3D environment as a static background. Interaction with the 3D model (rotate or translate) could be done using mouse and keyboard inputs. For the augmented environment, the user was asked to wear a HMD and the augmented information could be visualized in the real world. In this case, an expert of the system supported the user when he/she needed to change the object's texture. However, all other interactions with the object were done by the user himself/herself using bare hands (Figure 6.5).

Table 6.1. Post-task questionnaire contents

#	**Statements**
S1	I found the system helpful for deciding the most suitable texture.
S2	It was easy to compare among different textures on the object.
S3	I perceived the textured object as part of the real environment.
S4	I think the texture was correctly aligned to the object.
S5	It was easy to visualize the texture on the object.

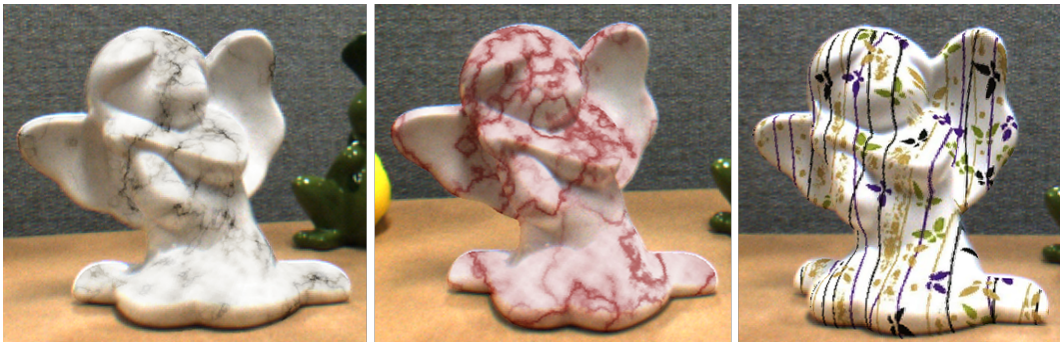


Figure 6.6. Virtual textures available to the user, rendered on the physical prototype: (a) Transparent marble, (b) red marble and (c) stripped pattern.

6.4.4 Results

A within-subjects design was used with a single independent variable (3D modeling software and AP system). A Likert scale (7-points, 1 = Disagree, 7 = Agree) was used to evaluate the overall performance of the system according to five measures: *Helpfulness, Ease of Comparison, Ease of Understanding, Precision* and *Ease of Visualization* [130]. The hypotheses considered are:

H_0: There is no difference between the design selection process using a 3D modeling software and an AP system.

H_a: There is a significant difference between the design selection process using a 3D modeling software and an AP system.

A two-tailed t-test was applied and significant differences between conditions were found for Helpfulness, with $t(9) = -3.88065, p < 0.05$ and Ease of Understanding, with $t(9) = -3.43063$, rejecting the null hypothesis and accepting the alternate hypothesis for these metrics. Figure 6.7 shows the average scores of each measure. The time participants required for completion of each task was also measured and its average is shown in Table 6.2.

Since the differences between the conditions were evaluated using ordinal variables (Likert scale), a further evaluation was performed using a non-parametric statistical test. The null hypothesis was maintained and the Friedman test was applied. Similar to the t-test, significant differences were found for the metric Helpfulness and Ease of Understanding, with $X^2(9) = 7.0, p = 0.008$ for both of them. Therefore, the results of this user study was consistent for both parametric and non-parametric statistical tests.

Table 6.2. Time for completion of each task (minutes)

	3D software	AP system
Average	1.7	1.9
Upper bound	7.25	5.55
Lower bound	0.31	0.30

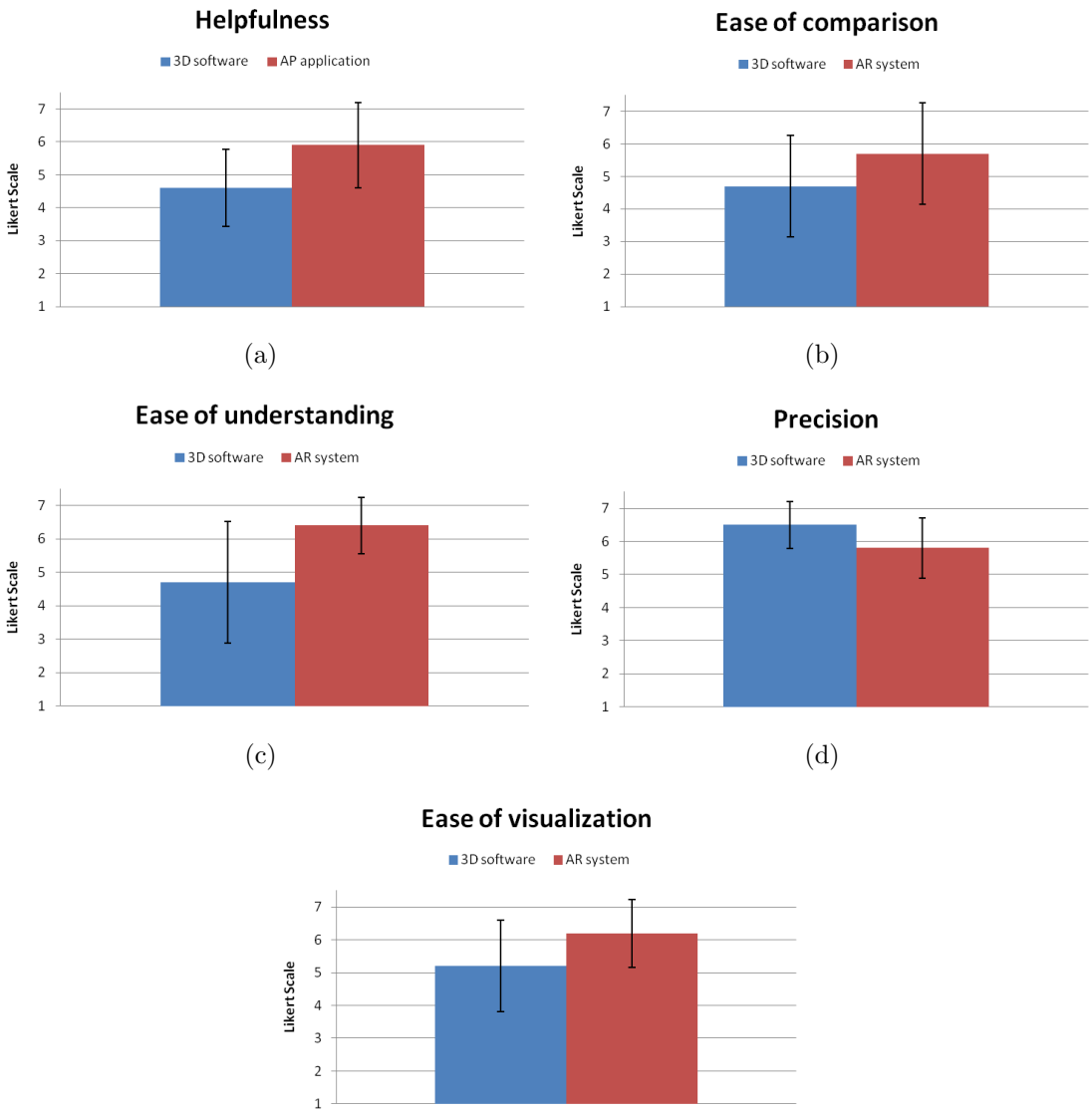


Figure 6.7. Average scores for the significant Likert scale measures: (a) Helpfulness, (b) Ease of Comparison, (c) Ease of Understanding, (d) Precision and (e) Ease of Visualization.

6.5. Discussion

The results of this pilot user study showed good results for the current implementation and overall participants enjoyed the experience provided by the AP application. However, there are some issues to be solved, most of them related to the tracking, not the application itself: some users pointed out the jittering of the 3D model as distracting and most probably the main problem with this approach. This explains the lower rating of the Precision metric (Figure 6.7 (d)).

A well known drawback of model-based tracking approaches was also mentioned by the participants: although they could move the object being tracked, fast movements caused tracking failures. Thus, after tracking failed once, some participants started to manipulate the object more carefully and slowly, making the interaction unnatural. Some solutions regarding this problem include improvement of the quadrics fitting approach and also application of a smoothing filter on the calculated pose parameters.

On the other hand, many participants agreed that the AP application made the visualization of the texture easier, being more realistic and with a color richness they could not perceive while using the 3D modeling software. The possibility to move the object in the real world using bare hands were also added as advantages. These comments confirmed the results obtained through the questionnaires, with the AP system being more helpful to decide the most suitable texture and to perceive the textured object as part of the real environment.

Regarding the 3D modeling software, the main complaints were about the time required to get used to the interface. Although few actions were actually required for the user to learn, sometimes they got confused with the shortcuts. Many participants complained about the rotation, which was considered not easy to control - One participant mentioned that small movements of the mouse were resulting in large changes. Other comments include the fact that the object did not seem real because it did not blend correctly with environment elements such as the lighting direction and shadows. However, the perfect alignment of the texture on the object was mentioned as a good point in this condition.

At the end of the experiment, users who changed their choice of the design after evaluation using the different systems were asked to mention the main reason for this change. Eight of the participants changed their initial choice, mentioning

that some textures appeared to be fake in the 3D modeling software, but very realistic when visualized through the HMD. The stripped pattern was the most commented one, whose lines could be easily seen following the object's curves on the AP application, but according to the participants, it appeared flat on the other system. They also mentioned about the color of the texture as being more vivid with AR and opaque in the 3D modeling software.

These observations shows that AP application developed is able to give a better visualization of the final product design, and by using the suggested implementation considering the effects of the illumination on the object's texture was important to help the decision of the final design. The preference of the AP system was also confirmed with a post-session questionnaire, where users were asked which system they would prefer if they had to perform a similar task again: 2 participants chose the 3D modeling software and 8 the AP system.

6.6. Concluding remarks

RP is a widely used technique during the development process of new products to evaluate and validate design solutions. However, there is still room for improvements specially regarding minimization of time and cost when changes on the visual appearance of the design are necessary.

This chapter proposes an AP application for easy evaluation of aesthetic aspects of a product in the real world environment, allowing interaction and browsing among different virtual textures in real-time. Different from previous approaches, fiducial markers are not necessary and since our method completely relies on the polygonal mesh representing the object shape, it does not require the object to have any kind of texture, being a suitable application for prototypes evaluation, since usually they are non-textured objects. Furthermore, effects of the environment illumination were taken into account in this implementation, making the object's augmented appearance more realistic.

With the pilot user study, it was possible to get a first impression of the acceptance of this idea and improvement points in two main directions: The tracking approach and future experiments. In the former case, the jittering is the most important issue to be solved and improvement of the tracking to allow faster movements of the object are also being considered by improving the tracking

recovery method. A more complete user study is necessary for further evaluation and improvement of the application, considering a higher number of participants, who would be real designers.

CHAPTER 7

Conclusions

7.1. Thesis summary

This thesis has investigated the problem of determining the pose of non-textured 3D rigid curved objects using sparse polygonal meshes. A method to solve the trade-off between computational efficiency and accuracy when dealing with curved objects using an edge-based tracking system was developed by creating a new model representation. Furthermore, to improve the framework generality, methods for dealing with objects having less than six DOF as well as different number of observable DOF depending on the viewpoint were also implemented. Lastly, the applicability of the proposed framework was explored with implementation of an application for AR. To review, these main contributions are summarized below:

- **Quadrics patch representation**: This representation was constructed aiming at the possibility of tracking simple and complex curved shapes in an efficient manner. A local approximation using a general quadric equation for each patch on a sparse mesh representing the target object was suggested, whose calculation was based on the relationship between the dense mesh that originated the sparse mesh.

Curves representing the quadrics projection of the patches located on the object contour are used for matching with detected points in the video image. Finally, a standard model-based tracking framework was modified to handle these changes and the proposed method was validated trough quantitative and qualitative experiments.

- **Dealing with different observable DOF**: A method to test the number of observable DOF of the target object was included in the tracking framework, to solve two problems: Tracking of objects having less than six DOF, and dealing with objects with unobservable DOF at certain poses. When necessary, a recovery process is triggered, which is able to recover one missing DOF.

- **An Augmented Prototyping application**: By using the proposed framework, an augmented prototyping application was developed. Compared to previous approaches using AR to assist design evaluation, the proposed method had an additional advantage by considering the environment illumination effects on the virtual textures. Results from a pilot user study comparing the use of a 3D software and the proposed application were also presented.

7.2. Future work and open problems

Some suggestions to further improve the framework proposed in this thesis include:

- Edge-based trackers, in general, works well for non-textured objects. However, some disadvantages include sensibility to high inter-frame motion and it is prone to jittering, which for AR applications may be disturbing, as pointed out by some of the participants of the user study in Chapter 6. One possible direction for reducing the jittering include improvements on the quadrics fitting approach and application of a smoothing filter on the calculated pose parameters.

- The current version of the DOF recovery process described in Chapter 4, Section 4.2 is strongly affected by false positives caused by wrong edge detection. Therefore, future works include improvements to reduce these false

positives and also implementation of new modules to allow pose estimation when two or more DOF are missing.

- Currently, the quadrics fitting implementation is closely related to the polygonal simplification method used. Although the QEM approach [103] is good enough for most of the objects, in some cases, if the object has small details on it its surface, calculation of the quadrics approximation for these specific parts becomes difficult. This is because only few internal vertices are available after the mesh simplification. For instance, consider the game controller used in Chapter 6. From a side view (Figure 7.1(b)-(c)), the edges on the game controller buttons also belong to the outline. However, it is not possible to obtain a good quadrics fitting for these cases. For this specific object, since only the front view was used for testing the system, it did not represent a major problem, but depending on the goal of the application, different simplification methods should be considered.

- To allow a large variety of objects to be tracked by our framework, it is possible to extend it to deal with curved objects that are not entirely smooth. An example is the cup shown in Figure 1.2, whose upper and bottom part are not smooth. To obtain accurate tracking for these types 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 should be triggered.

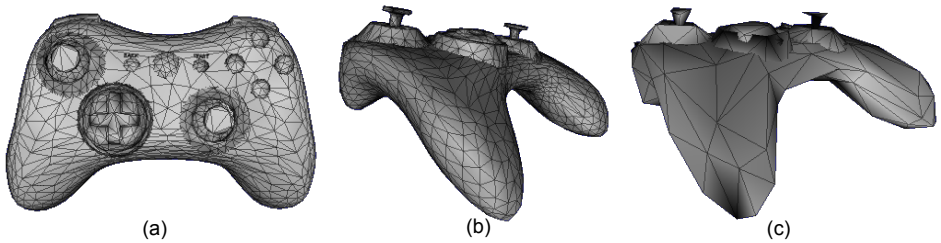


Figure 7.1. Detail on the game controller buttons: (a) front view, (b) side view with 100 patches and (c) side view, with 500 patches.

- With the necessary modifications to attend the restrictions of mobile devices, another possible application of the proposed framework includes AR targeting mobile devices, in which the data size of the 3D model can be critical for the tracking performance. By using the proposed approach with sparse meshes, loading 3D models from a remote server is faster as well as the tracking itself, with less model data to be analyzed during tracking.

Publication List

Journal Paper

1. Marina A. Oikawa, Takafumi Taketomi, Goshiro Yamamoto, Makoto Fujisawa, Toshiyuki Amano, Jun Miyazaki and Hirokazu Kato, "A model-based tracking framework for textureless 3D rigid curved objects". SBC Journal on 3D Interactive Systems, v. 3, no.2, pp. 2-15, 2012 (related to *Chapter 3* and *5*).

International Conferences

1. Kenzo Kumagai, Marina A. Oikawa, Takafumi Taketomi, Goshiro Yamamoto, Jun Miyazaki and Hirokazu Kato, "Robust model-based tracking considering changes in the measurable DoF of the target object", Proc. of *The 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 2157-2160, Tsukuba, Japan, Nov. 2012 (related to *Chapter 4* and *5*).

2. Marina A. Oikawa, Igor de S. Almeida, Takafumi Taketomi, Goshiro Yamamoto, Jun Miyazaki and Hirokazu Kato, "Augmented prototyping of 3D rigid curved surfaces", Proc. of *International Symposium on Mixed and Augmented Reality (ISMAR2012)*, pp. 307-308, Poster, Atlanta, USA, Nov. 2012 (related to *Chapter 6*).

3. Marina A. Oikawa, Takafumi Taketomi, Goshiro Yamamoto, Makoto Fujisawa, Toshiyuki Amano, Jun Miyazaki and Hirokazu Kato, "Local quadrics surface approximation for real-time tracking of textureless 3D rigid curved objects", Proc. of *The XIV Symposium on Virtual and Augmented Reality (SVR2012)*, pp. 246-253, Niteroi, Brazil, May 2012 (related to *Chapter 3* and *5*).

4. Marina A. Oikawa, Goshiro Yamamoto, Makoto Fujisawa, Toshiyuki Amano, Jun Miyazaki and Hirokazu Kato, "Model-based tracking of rigid curved objects using sparse polygonal meshes", Proc. of *The 21st International Conference on Artificial Reality and Telexistence (ICAT2011)*, p.145, Poster, Osaka, Japan, Nov. 2011 (related to *Chapter 3*).

5. Marina A. Oikawa, Goshiro Yamamoto, Makoto Fujisawa, Toshiyuki Amano, Jun Miyazaki and Hirokazu Kato, "Quantitative evaluation method for model-based tracking of 3D rigid curved objects", Proc. of *The 2nd International Workshop on AR/MR Registration, Tracking and Benchmarking (TrakMark2011)*, Basel, Switzerland, Oct. 2011 (related to *Chapter 5*).

6. Marina A. Oikawa, Makoto Fujisawa, Toshiyuki Amano, Jun Miyazaki and Hirokazu Kato, "Apparent contour tracking of 3D rigid curved objects based on quadrics approximation of the surface", Proc. of *The 3rd Korea-Japan Workshop in Mixed Reality (KJMR2010)*, pp.24-27, GyeongJu, Korea, Apr. 2010 (related to *Chapter 3*).

Award

1. SVR2012 Best Paper Award, May 2012.

Acknowledgments

Six years ago I literally crossed half the world to start my path in graduate school. However, this achievement would not have been possible without the support of my supervisor Professor Hirokazu Kato, to whom I am truly and heartily grateful for kindly welcome me in Japan and at the Interactive Media Design Laboratory. His patience while explaining (and sometimes re-explaining) in details basic concepts in my research field encouraged me to work harder and it was the strength that pushed me forward during the hard times I wanted to give up. It is countless the number of times he generously helped me, giving me valuable advices not only regarding my research topic but also showing me how to be a good researcher. I also thank him for the numerous opportunities given to me to present our work in international conferences, which allowed me to enjoy unforgettable new experiences, to meet and exchange ideas with other researchers, and also made me realize how amazing it is to explore other parts of the world. I feel blessed to have been lucky enough to have the chance of working with him during all these years.

I also would like to thank my thesis committee and co-supervisors Professor Naokazu Yokoya and Associate Professor Jun Miyazaki, as well as Associate Professor Masayuki Kanbara. Thank you very much for reviewing my thesis and for the insightful comments and suggestions that helped me to improve the overall quality of this thesis.

A special thanks to Assistant Professor Takafumi Taketomi, Assistant Professor Goshiro Yamamoto and to former Assistant Professors of IMD Lab, Makoto Fujisawa and Toshiyuki Amano. During my stay in this Lab, they helped me in so many different ways, giving many constructive comments during the lab meetings and never being too busy to help me whenever I needed. My thanks are also extended to all current and former members of IMD Lab. It was a great pleasure to work in this mixed environment, with students from different countries and having diverse personalities. In special, I would like to thank the Lab's secretary, Ms. Makiko Ueno, for her kindness and help with official documents and other bureaucracies, Kenzo Kumagai for helping me to complete part of my work in this thesis and Max Krichenbauer for preparing the HMD used in my experiments. Also, I would like to thank Atsushi Keiyaki and Yuichiro Fujimoto for helping me every time I struggled with the Japanese language.

During these six years, I was able to have a comfortable student life thanks to the support of a Japanese government scholarship. So, I am very grateful to the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan for the support given to me during my Master and PhD course at NAIST.

I also owe my deepest gratitude to my best friend, Igor Almeida. I am very thankful that we were able to share another journey together. Thank you for all nights you spent in the Lab helping me with my experiments, for helping me to improve my presentation skills and for the good advices. Thank you for the patience, for the humorous jokes that always make me forget my tiredness and for putting up with my moments of stubbornness for more than 15 years.

I also cannot thank enough Henry Chu, Erlyn Manguilimotan and Yuliana Lukamto for their friendship and help every single time I needed, related to research or not. Our talks and your advices always made me feel better. Also, to great friends I met in Japan, in special, Remy Martin, Estella Cheung, Elizabeth Ishikawa, Jordi Polo, Engelene Obien, Nico Prananta, and other NAIST friends, thank you for the wonderful moments we shared together. Living far from my family and not being able to visit them as much as I wanted was not easy. But thanks to you guys, I could feel Japan as my second home and daily life here became easier and enjoyable. Thank you for your company, for the serious and the silly talks, for always supporting me and for the good memories that will always warm my heart.

I am also indebted to all of you who I asked to proofread my papers and some chapters of my thesis. This acknowledgment is extended to Rodney Berry and Samuel Felix. Your comments and corrections were indeed a great help.

I also would like to thank a special person I met in Japan, Humberto Baba, whose love, friendship and unfaltering patience gave me the support I needed to keep going through all the ups and downs in this stage of my life.

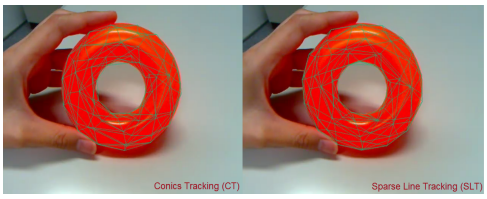
To my friends in Brazil, who always supported me and made their presence felt even with the distance and the time zone difference, thank you for always keeping in touch and cheering me up.

Finally, words will never be enough to thank the ones responsible for all successful steps in my life: My wonderful family, whose love and support give me enough strength to overcome all obstacles, and are my daily dose of motivation.

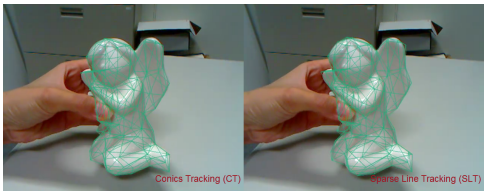
Appendix

A. Video sequences

This appendix contains descriptions of the video sequences used in the qualitative evaluation of the proposed framework. All videos are available in the following url: http://imd.naist.jp/videos/quadrics.html.

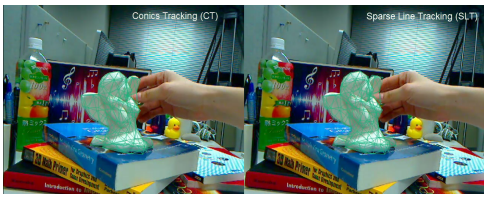


torus_ct_slt.avi
Comparison between CT and SLT for the torus.

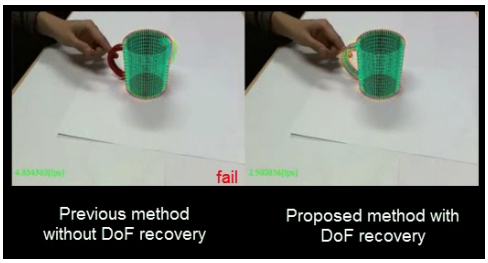


angel_object_ct_slt.avi
Comparison between CT and SLT for the angel when the camera is fixed.

angel_camera_ct_slt.avi
Comparison between CT and SLT for the angel when the camera is moving.

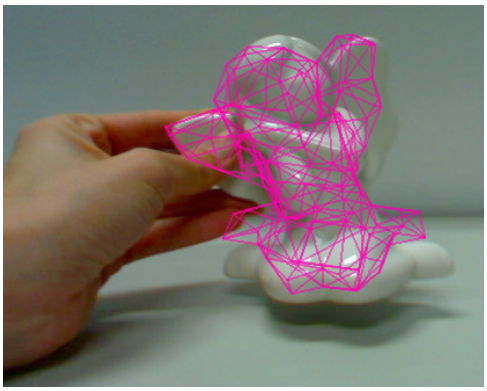


angel_cluttered.avi
Comparison between CT and SLT for the angel in a cluttered background.

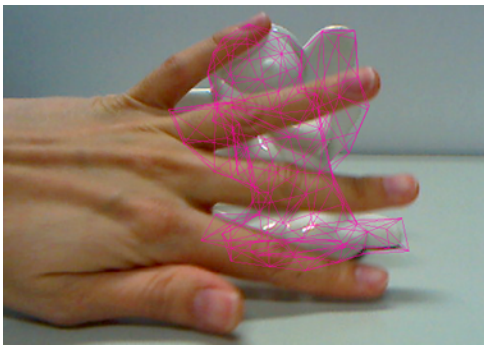


mug_ct_slt.avi
Comparison with and without DoF recovery for the mug.

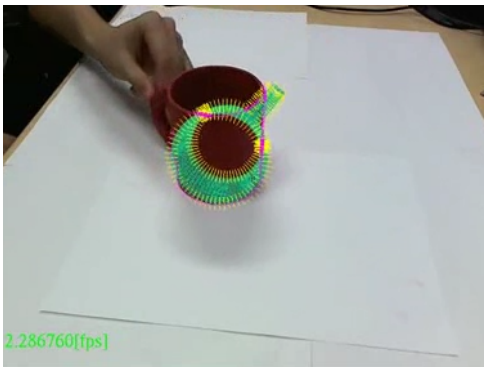
teapot.avi
Comparison with and without DoF recovery for the teapot.



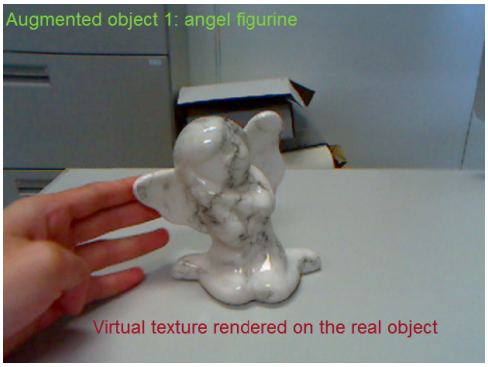
angel_failure.avi
Example of failure conditions caused by high inter-frame motion.



angel_occlusion.avi
Example of failure conditions caused by object occlusion.



mug_fail.avi
Example of failure conditions caused by wrong calculation of the rotation axis.



ap_demo.avi
Augmented Prototyping application demonstration.

B. Apparent contour of quadrics

To obtain the apparent contour equation of quadrics, Cipolla & Giblin [29] consider the cone of rays from the camera center $\mathbf{O} = (o_{xc}, o_{yc}, o_{zc}, 1)^T$ that are tangent to the quadric $\mathbf{Q}$. If a point $\mathbf{X}_w$ is on the quadric, any point on the line joining $\mathbf{X}_w$ to $\mathbf{O}$ can be written as:

$$\mathbf{V} = \lambda\mathbf{O} + (1-\lambda)\mathbf{X}_w \tag{B.1}$$

where $\mathbf{V} = (v_1, v_2, v_3, 1)^T$. The condition $\mathbf{V}^T\mathbf{Q}\mathbf{V} = 0$ can be tested to know if $\mathbf{V}$ lies on the quadric:

$$(\lambda\mathbf{O}^T + (1-\lambda)\mathbf{X}_w^T)\mathbf{Q}(\lambda\mathbf{O} + (1-\lambda)\mathbf{X}_w) = 0. \tag{B.2}$$

Solving the previous equation gives a quadratic equation for λ:

$$\begin{aligned}(\mathbf{O}^T\mathbf{Q}\mathbf{O} - \mathbf{O}^T\mathbf{Q}\mathbf{X}_w - \mathbf{X}_w^T\mathbf{Q}\mathbf{O} + \mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w)\lambda^2 + \\ (\mathbf{X}_w^T\mathbf{Q}\mathbf{O} + \mathbf{O}^T\mathbf{Q}\mathbf{X}_w - 2\mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w)\lambda + \mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w = 0.\end{aligned} \tag{B.3}$$

Since $\mathbf{O}^T\mathbf{Q}\mathbf{X}_w$ is a 1x1 matrix and $\mathbf{Q}$ is a symmetric matrix, it is possible to write $\mathbf{O}^T\mathbf{Q}\mathbf{X}_w = \mathbf{X}_w^T\mathbf{Q}^T\mathbf{O} = \mathbf{X}_w^T\mathbf{Q}\mathbf{O}$. This equality allows Equation B.3 to be rewritten as:

$$(\mathbf{O}^T\mathbf{Q}\mathbf{O} - 2\mathbf{X}_w^T\mathbf{Q}\mathbf{O} + \mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w)\lambda^2 + 2(\mathbf{X}_w^T\mathbf{Q}\mathbf{O} - \mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w)\lambda + \mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w = 0. \tag{B.4}$$

The equation above is equivalent to the quadratic equation represented by $a\lambda^2 + 2b\lambda + c = 0$, with $a = \mathbf{O}^T\mathbf{Q}\mathbf{O} - 2\mathbf{X}_w^T\mathbf{Q}\mathbf{O} + \mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w$, $b = \mathbf{X}_w^T\mathbf{Q}\mathbf{O} - \mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w$ and $c = \mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w$. The line is tangent to the surface $\mathbf{S}$ when this quadratic equation has equal roots, which leads to the condition $b^2 = ac$:

$$(\mathbf{X}_w^T\mathbf{Q}\mathbf{O} - \mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w)^2 = (\mathbf{O}^T\mathbf{Q}\mathbf{O} - 2\mathbf{X}_w^T\mathbf{Q}\mathbf{O} + \mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w)(\mathbf{X}_w^T\mathbf{Q}\mathbf{X}_w). \tag{B.5}$$

Rearranging this equation and considering $\mathbf{O}^T\mathbf{Q}\mathbf{O}$ is a scalar, it is possible to obtain:

$$\mathbf{X}_w^T[(\mathbf{Q}\mathbf{O})(\mathbf{O}\mathbf{Q})^T - (\mathbf{O}^T\mathbf{Q}\mathbf{O})\mathbf{Q}]\mathbf{X}_w = 0, \tag{B.6}$$

where $A = (\mathbf{Q}\mathbf{O})(\mathbf{O}\mathbf{Q})^T - (\mathbf{O}^T\mathbf{Q}\mathbf{O})\mathbf{Q}$ and represents the matrix of the quadric cone with vertex in the camera center $\mathbf{O}$ and with lines tangent to $\mathbf{S}$.

Considering the camera center is positioned at $\mathbf{O} = (0, 0, 0, 1)^T$ and using the notation given in Equation 2.10, leads to:

$$\mathbf{A} = \begin{pmatrix} qq^T - cQ_3 & 0 \\ 0 & 0 \end{pmatrix}. \tag{B.7}$$

If the image plane is at $z = f$, the apparent contour in this plane is defined as the intersection of the cone of tangent rays with the image plane, i.e., the points $\mathbf{x} = (x, y, f)^T$ where:

$$\mathbf{x}^T(qq^T - cQ_3)\mathbf{x} = 0 \therefore \mathbf{x}^T\mathbf{A}\mathbf{x} = 0 \tag{B.8}$$

such that $qq^T - cQ_3$ represents the parameters of the corresponding conic.

C. Polygonal mesh simplification results

Table C.1. Forward distance ($M_1 \rightarrow M_2$)

Object	**# of patches (M_2)**	**Method**	**Max**	**Mean**	**RMS**
Angel	792 patches	PR	0.013907	0.001987	0.002657
		QEM	0.010014	0.001749	0.002226
	396 patches	PR	0.031571	0.003967	0.005322
		QEM	0.022625	0.003576	0.004601
	198 patches	PR	0.055495	0.007677	0.010241
		QEM	0.042865	0.00735	0.00948
	98 patches	PR	0.098271	0.015631	0.021469
		QEM	0.06694	0.013479	0.017462
Bunny	1560 patches	PR	0.141957	0.014024	0.020734
		QEM	0.066781	0.009004	0.011632
	779 patches	PR	0.221248	0.023636	0.033621
		QEM	0.117667	0.016974	0.021918
	388 patches	PR	0.379191	0.049374	0.072899
		QEM	0.194659	0.035582	0.045758
	193 patches	PR	0.55219	0.089603	0.125469
		QEM	0.424752	0.072233	0.089421

Table C.2. Backward distance ($M_2 \rightarrow M_1$)

Object	**# of patches (M_2)**	**Method**	**Max**	**Mean**	**RMS**
Angel	792 patches	PR	0.014928	0.002695	0.00348
		QEM	0.013797	0.001735	0.00226
	396 patches	PR	0.035032	0.005052	0.006627
		QEM	0.02682	0.003449	0.004538
	198 patches	PR	0.055519	0.009276	0.012295
		QEM	0.049604	0.007061	0.009387
	98 patches	PR	0.084748	0.015648	0.020485
		QEM	0.060874	0.012748	0.016421
Bunny	1560 patches	PR	0.176407	0.012569	0.018592
		QEM	0.081031	0.009611	0.012604
	779 patches	PR	0.279456	0.021548	0.031643
		QEM	0.291398	0.017787	0.024495
	388 patches	PR	0.352182	0.038441	0.057035
		QEM	0.496251	0.037833	0.05431
	193 patches	PR	0.484969	0.067173	0.093858
		QEM	0.496251	0.069241	0.088432

D. Jacobian matrix J_{es}

The Jacobian matrix J_{es} relating the distance between projected and detected edges and the pose parameter vector s is calculated by:

$$J_{es} = J_{eC} J_{CQ_c} J_{Q_c T_i} J_{T_i T_{Mi}} J_{T_{Mi} s}, \tag{D.9}$$

where each term represent the Jacobian of:

a) J_{eC}: the distance function d_e with respect to the conic parameters.

b) J_{CQ_c}: the conic parameters with respect to the quadric parameters in camera coordinates $\mathbf{Q}_c$.

c) $J_{Q_c T_i}$: the quadric parameters in camera coordinates $\mathbf{Q}_c$ with respect to the parameters of $\mathbf{T}_i$, the inverse of the modelview matrix $\mathbf{T}$.

d) $J_{T_i T_{Mi}}$: the inverse of the modelview matrix $\mathbf{T}_i$ with respect to the inverse matrix $\mathbf{T}_{Mi}$.

e) $J_{T_{Mi} s}$: the inverse matrix $\mathbf{T}_{Mi}$ with respect to the pose vector s.

D.1 Jacobian matrix J_{eC}

The Jacobian of the distance function with respect to the conic parameters is a 1x6 matrix calculated using the following equation:

$$J_{eC} = J_{eL}\, J_{LX}\, J_{XP}\, J_{PA}\, J_{AC}. \tag{D.10}$$

J_{eL} represents a 1x3 matrix with the partial derivatives of the distance between the detected point $p_0 = (x_0, y_0)$ and the line $l : ax + by + c = 0$ passing through the points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, given in Equation 3.29.

$$J_{eL} = \left(\frac{\partial l}{\partial a} \frac{\partial l}{\partial b} \frac{\partial l}{\partial c} \right), \tag{D.11}$$

where:

$$\frac{\partial l}{\partial a} = \frac{x_0(a^2+b^2) - a(ax_0+by_0+c)}{(a^2+b^2)(\sqrt{a^2+b^2})},$$

$$\frac{\partial l}{\partial b} = \frac{y_0(a^2+b^2) - b(ax_0+by_0+c)}{(a^2+b^2)(\sqrt{a^2+b^2})},$$

$$\frac{\partial l}{\partial c} = \frac{\sqrt{a^2+b^2}}{(a^2+b^2)} = \frac{1}{\sqrt{(a^2+b^2)}}. \tag{D.12}$$

From Equation 3.28, each of the components of the line l has values $a = (y_2 - y_1)$, $b = (x_1 - x_2)$, $c = (x_2y_1 - y_2x_1)$. The partial derivatives of those terms with respect to the points p_1 and p_2 is given by the 3x4 matrix:

$$J_{LX} = \begin{pmatrix} \frac{\partial a}{\partial x_1} & \frac{\partial a}{\partial y_1} & \frac{\partial a}{\partial x_2} & \frac{\partial a}{\partial y_2} \\ \frac{\partial b}{\partial x_1} & \frac{\partial b}{\partial y_1} & \frac{\partial b}{\partial x_2} & \frac{\partial b}{\partial y_2} \\ \frac{\partial c}{\partial x_1} & \frac{\partial c}{\partial y_1} & \frac{\partial c}{\partial x_2} & \frac{\partial c}{\partial y_2} \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ -y_2 & x_2 & y_1 & -x_1 \end{pmatrix}. \tag{D.13}$$

J_{XP} is a 4x2 matrix with the partial derivatives of Equation 3.24 with respect to p_1 and p_2:

$$J_{XP} = \begin{pmatrix} \frac{\partial x_1}{\partial p_1} & \frac{\partial x_1}{\partial p_2} \\ \frac{\partial y_1}{\partial p_1} & \frac{\partial y_1}{\partial p_2} \\ \frac{\partial x_2}{\partial p_1} & \frac{\partial x_2}{\partial p_2} \\ \frac{\partial y_2}{\partial p_1} & \frac{\partial y_2}{\partial p_2} \end{pmatrix} = \begin{pmatrix} x_0 - x_1' & 0 \\ y_0 - y_1' & 0 \\ 0 & x_0 - x_2' \\ 0 & y_0 - y_2' \end{pmatrix}. \tag{D.14}$$

J_{PA} is a 2x6 matrix with partial derivatives of p_1 and p_2 with respect to the parameters given in Equation 3.27:

$$J_{PA} = \begin{pmatrix} \frac{\partial p_1}{\partial \mathbf{a}_1} & \frac{\partial p_1}{\partial \mathbf{b}_1} & \frac{\partial p_1}{\partial \mathbf{c}_1} & \frac{\partial p_1}{\partial \mathbf{a}_2} & \frac{\partial p_1}{\partial \mathbf{b}_2} & \frac{\partial p_1}{\partial \mathbf{c}_2} \\ \frac{\partial p_2}{\partial \mathbf{a}_1} & \frac{\partial p_2}{\partial \mathbf{b}_1} & \frac{\partial p_2}{\partial \mathbf{c}_1} & \frac{\partial p_2}{\partial \mathbf{a}_2} & \frac{\partial p_2}{\partial \mathbf{b}_2} & \frac{\partial p_2}{\partial \mathbf{c}_2} \end{pmatrix} = \begin{pmatrix} \frac{\partial p_1}{\partial \mathbf{a}_1} & \frac{\partial p_1}{\partial \mathbf{b}_1} & \frac{\partial p_1}{\partial \mathbf{c}_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial p_2}{\partial \mathbf{a}_2} & \frac{\partial p_2}{\partial \mathbf{b}_2} & \frac{\partial p_2}{\partial \mathbf{c}_2} \end{pmatrix}, \tag{D.15}$$

where:

$$\frac{\partial p_1}{\partial \mathbf{a}_1} = \frac{(\mathbf{b}_1 \pm \sqrt{\mathbf{b}_1^2 - 4\mathbf{a_1}\mathbf{c_1}}}{2\mathbf{a}_1^2} \pm \frac{\mathbf{c}_1}{(\mathbf{a}_1 - \sqrt{\mathbf{b}_1^2 - 4\mathbf{a_1}\mathbf{c_1}}},$$

$$\frac{\partial p_1}{\partial \mathbf{b}_1} = \left(-1 \pm \frac{\mathbf{b}_1}{\sqrt{\mathbf{b}_1^2 - 4\mathbf{a_1}\mathbf{c_1}}}\right)\left(\frac{1}{2\mathbf{a}_1}\right),$$

$$\frac{\partial p_1}{\partial \mathbf{c}_1} = \pm\frac{1}{\sqrt{\mathbf{b}_1^2 - 4\mathbf{a_1}\mathbf{c_1}}},$$

$$\frac{\partial p_1}{\partial \mathbf{a}_2} = \frac{(\mathbf{b}_2 \pm \sqrt{\mathbf{b}_2^2 - 4\mathbf{a_2}\mathbf{c_2}}}{2\mathbf{a}_2^2} \pm \frac{\mathbf{c}_2}{(\mathbf{a}_2 - \sqrt{\mathbf{b}_2^2 - 4\mathbf{a_2}\mathbf{c_2}}},$$

$$\frac{\partial p_1}{\partial \mathbf{b}_2} = \left(-1 \pm \frac{\mathbf{b}_2}{\sqrt{\mathbf{b}_2^2 - 4\mathbf{a_2}\mathbf{c_2}}}\right)\left(\frac{1}{2\mathbf{a}_2}\right),$$

$$\frac{\partial p_1}{\partial \mathbf{c}_2} = \pm\frac{1}{\sqrt{\mathbf{b}_2^2 - 4\mathbf{a_2}\mathbf{c_2}}}. \tag{D.16}$$

J_{AC} is a 6x6 matrix derived from Equation 3.26, representing the partial derivatives of the quadratic equation parameters $\mathbf{a}_i x_2 + \mathbf{b}_i x + \mathbf{c}_i = 0$ with respect to the conic parameters:

$$J_{AC} = \begin{pmatrix} \frac{\partial \mathbf{a_1}}{\partial c_1} & \frac{\partial \mathbf{a_1}}{\partial c_2} & \cdots & \frac{\partial \mathbf{a_1}}{\partial c_6} \\ \frac{\partial \mathbf{b_1}}{\partial c_1} & \frac{\partial \mathbf{b_1}}{\partial c_2} & \cdots & \frac{\partial \mathbf{b_1}}{\partial c_6} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{c_2}}{\partial c_1} & \frac{\partial \mathbf{c_2}}{\partial c_2} & \cdots & \frac{\partial \mathbf{c_2}}{\partial c_6} \end{pmatrix}$$

$$= \begin{pmatrix} (x_0 - x_1')^2 & (y_0 - y_1')^2 & (x_0 - x_1')(y_0 - y_1') & 0 & 0 & 0 \\ 2x_1'(x_0 - x_1') & 2y_1'(y_0 - y_1') & (x_0 y_1' + x_1' y_0 - 2x_1' y_1') & (x_0 - x_1') & (y_0 - y_1') & 0 \\ x_1^2 & y_1'^2 & x_1' y_1' & x_1' & y_1' & 1 \\ (x_0 - x_2')^2 & (y_0 - y_2')^2 & (x_0 - x_2')(y_0 - y_2') & 0 & 0 & 0 \\ 2x_2'(x_0 - x_2') & 2y_2'(y_0 - y_2') & (x_0 y_2' + x_2' y_0 - 2x_2' y_2') & (x_0 - x_2') & (y_0 - y_2') & 0 \\ x_2^2 & y_2'^2 & x_2' y_2' & x_2' & y_2' & 1 \end{pmatrix}. \tag{D.17}$$

D.2 Jacobian matrix J_{CQ_c}

Solving Equation 3.16 using the matrix $\mathbf{A}$ given in Equation 3.6, the conic parameters $\mathbf{C} = (c_1\, c_2\, c_3\, c_4\, c_5\, c_6)^T$ have the following values:

$$
\begin{aligned}
c_1 &= pi_{11}^2(b_{c1}^2 - c_c a_{c1}),\\
c_2 &= pi_{22}^2(b_{c2}^2 - c_c a_{c2}),\\
c_3 &= 2pi_{11}pi_{22}(b_{c1}b_{c2} - c_c a_{c4}),\\
c_4 &= 2pi_{11}p_{11}[pi_{13}(b_{c1}^2 - c_c a_{c1}) + pi_{23}(b_{c1}b_{c2} - c_c a_{c4}) + (b_{c1}b_{c3} - c_c a_{c6})],\\
c_5 &= 2pi_{22}p_{11}[pi_{13}(b_{c2}b_{c1} - c_c a_{c4}) + pi_{23}(b_{c2}^2 - c_c a_{c2}) + (b_{c2}b_{c3} - c_c a_{c5})],\\
c_6 &= p_{11}^2[pi_{13}^2(b_{c1}^2 - c_c a_{c1}) + 2pi_{13}pi_{23}(b_{c1}b_{c2} - c_c a_{c4}) +\\
&\quad 2pi_{13}(b_{c1}b_{c3} - c_c a_{c6}) + pi_{13}^2(b_{c2}^2 - c_c a_{c2}) +\\
&\quad 2pi_{23}(b_{c3}b_{c2} - c_c a_{c5}) + (b_{c3}^2 - c_c a_{c3})].
\end{aligned}
\tag{D.18}
$$

These expressions are used to calculate the Jacobian J_{CQ_c}, which corresponds to a 6x10 matrix with the partial derivatives of the conic parameters with respect to the quadric parameters in camera coordinates:

$$
\begin{aligned}
J_{CQ_c} &= \begin{pmatrix}
\frac{\partial c_1}{\partial a_{c_1}} & \frac{\partial c_1}{\partial a_{c_2}} & \cdots & \frac{\partial c_1}{\partial c_c}\\
\frac{\partial c_2}{\partial a_{c_1}} & \frac{\partial c_2}{\partial a_{c_2}} & \cdots & \frac{\partial c_2}{\partial c_c}\\
\vdots & \vdots & \ddots & \vdots\\
\frac{\partial c_6}{\partial a_{c_1}} & \frac{\partial c_6}{\partial a_{c_2}} & \cdots & \frac{\partial c_6}{\partial c_c}
\end{pmatrix}\\
&= \begin{pmatrix}
\frac{\partial c_1}{\partial a_{c_1}} & 0 & 0 & 0 & 0 & 0 & \frac{\partial c_1}{\partial b_{c1}} & 0 & 0 & \frac{\partial c_1}{\partial c_c}\\
0 & \frac{\partial c_2}{\partial a_{c_2}} & 0 & 0 & 0 & 0 & 0 & \frac{\partial c_2}{\partial b_{c2}} & 0 & \frac{\partial c_2}{\partial c_c}\\
0 & 0 & 0 & \frac{\partial c_3}{\partial a_{c_4}} & 0 & 0 & \frac{\partial c_3}{\partial b_{c_1}} & \frac{\partial c_3}{\partial b_{c_2}} & 0 & \frac{\partial c_3}{\partial c_c}\\
\frac{\partial c_4}{\partial a_{c_1}} & 0 & 0 & \frac{\partial c_4}{\partial a_{c_4}} & 0 & \frac{\partial c_4}{\partial a_{c_6}} & \frac{\partial c_4}{\partial b_{c_1}} & \frac{\partial c_4}{\partial b_{c_2}} & \frac{\partial c_4}{\partial b_{c_3}} & \frac{\partial c_4}{\partial c_c}\\
0 & \frac{\partial c_5}{\partial a_{c_2}} & 0 & \frac{\partial c_5}{\partial a_{c_4}} & \frac{\partial c_5}{\partial a_{c_5}} & 0 & \frac{\partial c_5}{\partial b_{c_1}} & \frac{\partial c_5}{\partial b_{c_2}} & \frac{\partial c_5}{\partial b_{c_3}} & \frac{\partial c_5}{\partial c_c}\\
\frac{\partial c_6}{\partial a_{c_1}} & \frac{\partial c_6}{\partial a_{c_2}} & \frac{\partial c_6}{\partial a_{c_3}} & \frac{\partial c_6}{\partial a_{c_4}} & \frac{\partial c_6}{\partial a_{c_5}} & \frac{\partial c_6}{\partial a_{c_6}} & \frac{\partial c_6}{\partial b_{c_1}} & \frac{\partial c_6}{\partial b_{c_2}} & \frac{\partial c_6}{\partial b_{c_3}} & \frac{\partial c_6}{\partial c_c}
\end{pmatrix},
\end{aligned}
\tag{D.19}
$$

where:

$$\frac{\partial c_1}{\partial a_{c_1}} = -c_c pi_{11}^2 \qquad \frac{\partial c_1}{\partial b_{c1}} = 2b_{c1} pi_{11}^2 \qquad \frac{\partial c_1}{\partial c_c} = -a_{c1} pi_{11}^2$$

$$\frac{\partial c_2}{\partial a_{c_2}} = -c_c pi_{22}^2 \qquad \frac{\partial c_2}{\partial b_{c2}} = 2b_{c2} pi_{22}^2 \qquad \frac{\partial c_2}{\partial c_c} = -a_{c2} pi_{22}^2$$

$$\frac{\partial c_3}{\partial a_{c_4}} = -2pi_{11} pi_{22} c_c \qquad \frac{\partial c_3}{\partial b_{c_1}} = 2pi_{11} pi_{22} b_{c2} \qquad \frac{\partial c_3}{\partial b_{c_2}} = 2pi_{11} pi_{22} b_{c1}$$

$$\frac{\partial c_3}{\partial c_c} = -2pi_{11} pi_{22} a_{c4} \qquad \frac{\partial c_4}{\partial a_{c_1}} = -2pi_{11} p_{11} pi_{13} c_c \qquad \frac{\partial c_4}{\partial a_{c_4}} = -2pi_{11} p_{11} pi_{23} c_c$$

$$\frac{\partial c_4}{\partial a_{c_6}} = -2pi_{11} p_{11} c_c \qquad \frac{\partial c_4}{\partial b_{c_2}} = 2pi_{11} p_{11} pi_{23} b_{c1} \qquad \frac{\partial c_4}{\partial b_{c_3}} = 2pi_{11} p_{11} b_{c1}$$

$$\frac{\partial c_4}{\partial b_{c_1}} = 2pi_{11} p_{11}(2pi_{13} b_{c1} + pi_{23} b_{c2} + b_{c3}) \quad \frac{\partial c_4}{\partial c_c} = -2pi_{11} p_{11}(pi_{13} a_{c1} + pi_{23} a_{c4} + a_{c6})$$

$$\frac{\partial c_5}{\partial a_{c_2}} = -2pi_{22} pi_{23} p_{11} c_c \qquad \frac{\partial c_5}{\partial a_{c_4}} = -2pi_{22} p_{11} pi_{13} c_c \qquad \frac{\partial c_5}{\partial a_{c_5}} = -2pi_{22} p_{11} c_c$$

$$\frac{\partial c_5}{\partial b_{c_1}} = 2pi_{22} p_{11} pi_{13} b_{c2} \qquad \frac{\partial c_5}{\partial b_{c_2}} = 2pi_{22} p_{11}(pi_{13} b_{c1} + 2pi_{23} b_{c2} + b_{c3})$$

$$\frac{\partial c_5}{\partial b_{c_3}} = 2pi_{22} p_{11} b_{c2} \qquad \frac{\partial c_5}{\partial c_c} = -2pi_{22} p_{11}(pi_{13} a_{c4} + pi_{23} a_{c2} + a_{c5})$$

$$\frac{\partial c_6}{\partial a_{c_1}} = -pi_{13}^2 p_{11}^2 c_c \qquad \frac{\partial c_6}{\partial a_{c_2}} = -pi_{23}^2 p_{11}^2 c_c \qquad \frac{\partial c_6}{\partial a_{c_3}} = -p_{11}^2 c_c$$

$$\frac{\partial c_6}{\partial a_{c_4}} = -2p_{11}^2 pi_{13} pi_{23} c_c \qquad \frac{\partial c_6}{\partial a_{c_5}} = -2p_{11}^2 pi_{23} c_c \qquad \frac{\partial c_6}{\partial a_{c_6}} = -2p_{11}^2 pi_{13} c_c$$

$$\frac{\partial c_6}{\partial b_{c_1}} = 2p_{11}^2(pi_{13}^2 b_{c1} + pi_{13} b_{c3} + pi_{23} pi_{13} b_{c2}) \quad \frac{\partial c_6}{\partial b_{c_2}} = 2p_{11}^2(pi_{23}^2 b_{c2} + pi_{23} b_{c3} + pi_{23} pi_{13} b_{c1})$$

$$\frac{\partial c_6}{\partial b_{c_3}} = 2p_{11}^2(b_{c3} + pi_{13} b_{c1} + pi_{23} b_{c2})$$

$$\frac{\partial c_6}{\partial c_c} = -p_{11}^2[(pi_{13}^2 a_{c1} + pi_{23}^2 a_{c2} + a_{c3}) + 2(pi_{23} pi_{13} a_{c4} + pi_{13} a_{c6} + pi_{23} a_{c5})]. \tag{D.20}$$

D.3 Jacobian matrix $J_{Q_cT_i}$

$J_{Q_cT_i}$ represents a 10x12 matrix with the partial derivatives of the quadric parameters in camera coordinates with respect to the parameters of the inverse modelview matrix $\mathbf{T}_i$. Considering the following notation:

$$\mathbf{Q}_c = \begin{pmatrix} a_{c1} & a_{c4} & a_{c6} & b_{c1} \\ a_{c4} & a_{c2} & a_{c5} & b_{c2} \\ a_{c6} & a_{c5} & a_{c3} & b_{c3} \\ b_{c1} & b_{c2} & b_{c3} & c_c \end{pmatrix}; \mathbf{T}_i = \begin{pmatrix} ti_{11} & ti_{12} & ti_{13} & ti_{14} \\ ti_{21} & ti_{22} & ti_{23} & ti_{24} \\ ti_{31} & ti_{32} & ti_{33} & ti_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}; \mathbf{Q}_w = \begin{pmatrix} a_{w1} & a_{w4} & a_{w6} & b_{w1} \\ a_{w4} & a_{w2} & a_{w5} & b_{w2} \\ a_{w6} & a_{w5} & a_{w3} & b_{w3} \\ b_{w1} & b_{w2} & b_{w3} & c_w \end{pmatrix},$$

the necessary expressions are obtained using Equation 3.5:

$$\begin{aligned}
a_{c1} &= ti_{11}^2 a_{w1} + 2ti_{11}a_{w4}ti_{21} + 2ti_{11}a_{w6}ti_{31} + ti_{21}^2 a_{w2} + 2ti_{21}a_{w5}ti_{31} + ti_{31}^2 a_{w3}, \\
a_{c2} &= ti_{12}^2 a_{w1} + 2ti_{12}a_{w4}ti_{22} + 2ti_{12}a_{w6}ti_{32} + ti_{22}^2 a_{w2} + 2ti_{22}a_{w5}ti_{32} + ti_{32}^2 a_{w3}, \\
a_{c3} &= ti_{13}^2 a_{w1} + 2ti_{13}a_{w4}ti_{23} + 2ti_{13}a_{w6}ti_{33} + ti_{23}^2 a_{w2} + 2ti_{23}a_{w5}ti_{33} + ti_{33}^2 a_{w3}, \\
a_{c4} &= ti_{11}(a_{w1}ti_{12} + a_{w4}ti_{22} + a_{w6}ti_{32}) + ti_{21}(a_{w4}ti_{12} + a_{w2}ti_{22} + a_{w5}ti_{32}) + \\
&\quad ti_{31}(a_{w6}ti_{12} + a_{w5}ti_{22} + a_{w3}ti_{32}), \\
a_{c5} &= ti_{12}(a_{w1}ti_{13} + a_{w4}ti_{23} + a_{w6}ti_{33}) + ti_{22}(a_{w4}ti_{13} + a_{w2}ti_{23} + a_{w5}ti_{33}) + \\
&\quad ti_{32}(a_{w6}ti_{13} + a_{w5}ti_{23} + a_{w3}ti_{33}), \\
a_{c6} &= ti_{11}(a_{w1}ti_{13} + a_{w4}ti_{23} + a_{w6}ti_{33}) + ti_{21}(a_{w4}ti_{13} + a_{w2}ti_{23} + a_{w5}ti_{33}) + \\
&\quad ti_{31}(a_{w6}ti_{13} + a_{w5}ti_{23} + a_{w3}ti_{33}), \\
b_{c1} &= ti_{11}(a_{w1}ti_{14} + a_{w4}ti_{24} + a_{w6}ti_{34} + b_{w1}) + ti_{21}(a_{w4}ti_{14} + a_{w2}ti_{24} + a_{w5}ti_{34} + b_{w2}) + \\
&\quad ti_{31}(a_{w6}ti_{14} + a_{w5}ti_{24} + a_{w3}ti_{34} + b_{w3}), \\
b_{c2} &= ti_{12}(a_{w1}ti_{14} + a_{w4}ti_{24} + a_{w6}ti_{34} + b_{w1}) + ti_{22}(a_{w4}ti_{14} + a_{w2}ti_{24} + a_{w5}ti_{34} + b_{w2}) + \\
&\quad ti_{32}(a_{w6}ti_{14} + a_{w5}ti_{24} + a_{w3}ti_{34} + b_{w3}), \\
b_{c3} &= ti_{13}(a_{w1}ti_{14} + a_{w4}ti_{24} + a_{w6}ti_{34} + b_{w1}) + ti_{23}(a_{w4}ti_{14} + a_{w2}ti_{24} + a_{w5}ti_{34} + b_{w2}) + \\
&\quad ti_{33}(a_{w6}ti_{14} + a_{w5}ti_{24} + a_{w3}ti_{34} + b_{w3}), \\
c_c &= ti_{14}^2 a_{w1} + 2ti_{14}a_{w4}ti_{24} + 2ti_{14}a_{w6}ti_{34} + 2ti_{14}b_{w1} + ti_{24}^2 a_{w2} + \\
&\quad 2ti_{24}a_{w5}ti_{34} + 2ti_{24}b_{w2} + ti_{34}^2 a_{w3} + 2ti_{34}b_{w3} + c_w,
\end{aligned} \tag{D.21}$$

and $J_{Q_cT_i}$ is calculated as:

$$J_{Q_cT_i} = \begin{pmatrix} \frac{\partial a_{c1}}{\partial ti_{11}} & \frac{\partial a_{c1}}{\partial ti_{12}} & \cdots & \frac{\partial a_{c1}}{\partial ti_{34}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_{c1}}{\partial ti_{11}} & \frac{\partial a_{c1}}{\partial ti_{12}} & \cdots & \frac{\partial a_{c1}}{\partial ti_{34}} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\partial a_{c1}}{\partial ti_{11}} & 0 & 0 & 0 & \frac{\partial a_{c1}}{\partial ti_{21}} & 0 & 0 & 0 & \frac{\partial a_{c1}}{\partial ti_{31}} & 0 & 0 & 0 \\ 0 & \frac{\partial a_{c2}}{\partial ti_{12}} & 0 & 0 & 0 & \frac{\partial a_{c2}}{\partial ti_{22}} & 0 & 0 & 0 & \frac{\partial a_{c2}}{\partial ti_{32}} & 0 & 0 \\ 0 & 0 & \frac{\partial a_{c3}}{\partial ti_{13}} & 0 & 0 & 0 & \frac{\partial a_{c3}}{\partial ti_{23}} & 0 & 0 & 0 & \frac{\partial a_{c3}}{\partial ti_{33}} & 0 \\ \frac{\partial a_{c4}}{\partial ti_{11}} & \frac{\partial a_{c4}}{\partial ti_{12}} & 0 & 0 & \frac{\partial a_{c4}}{\partial ti_{21}} & \frac{\partial a_{c4}}{\partial ti_{22}} & 0 & 0 & \frac{\partial a_{c4}}{\partial ti_{31}} & \frac{\partial a_{c4}}{\partial ti_{32}} & 0 & 0 \\ 0 & \frac{\partial a_{c5}}{\partial ti_{12}} & \frac{\partial a_{c5}}{\partial ti_{13}} & 0 & 0 & \frac{\partial a_{c5}}{\partial ti_{22}} & \frac{\partial a_{c5}}{\partial ti_{23}} & 0 & 0 & \frac{\partial a_{c5}}{\partial ti_{32}} & \frac{\partial a_{c5}}{\partial ti_{33}} & 0 \\ \frac{\partial a_{c6}}{\partial ti_{11}} & 0 & \frac{\partial a_{c6}}{\partial ti_{13}} & 0 & \frac{\partial a_{c6}}{\partial ti_{21}} & 0 & \frac{\partial a_{c6}}{\partial ti_{23}} & 0 & \frac{\partial a_{c6}}{\partial ti_{31}} & 0 & \frac{\partial a_{c6}}{\partial ti_{33}} & 0 \\ \frac{\partial b_{c1}}{\partial ti_{11}} & 0 & 0 & \frac{\partial b_{c1}}{\partial ti_{14}} & \frac{\partial b_{c1}}{\partial ti_{21}} & 0 & 0 & \frac{\partial b_{c1}}{\partial ti_{24}} & \frac{\partial b_{c1}}{\partial ti_{31}} & 0 & 0 & \frac{\partial b_{c1}}{\partial ti_{34}} \\ 0 & \frac{\partial b_{c2}}{\partial ti_{12}} & 0 & \frac{\partial b_{c2}}{\partial ti_{14}} & 0 & \frac{\partial b_{c2}}{\partial ti_{22}} & 0 & \frac{\partial b_{c2}}{\partial ti_{24}} & 0 & \frac{\partial b_{c2}}{\partial ti_{32}} & 0 & \frac{\partial b_{c2}}{\partial ti_{34}} \\ 0 & 0 & \frac{\partial b_{c3}}{\partial ti_{13}} & \frac{\partial b_{c3}}{\partial ti_{14}} & 0 & 0 & \frac{\partial b_{c3}}{\partial ti_{23}} & \frac{\partial b_{c3}}{\partial ti_{24}} & 0 & 0 & \frac{\partial b_{c3}}{\partial ti_{33}} & \frac{\partial b_{c3}}{\partial ti_{34}} \\ 0 & 0 & 0 & \frac{\partial c_c}{\partial ti_{14}} & 0 & 0 & 0 & \frac{\partial c_c}{\partial ti_{24}} & 0 & 0 & 0 & \frac{\partial c_c}{\partial ti_{34}} \end{pmatrix}, \tag{D.22}$$

where:

$$\frac{\partial a_{c1}}{\partial ti_{11}} = 2a_{w1}ti_{11} + 2a_{w4}ti_{21} + 2a_{w6}ti_{31} \qquad \frac{\partial a_{c1}}{\partial ti_{21}} = 2a_{w4}ti_{11} + 2a_{w2}ti_{21} + 2a_{w5}ti_{31}$$

$$\frac{\partial a_{c1}}{\partial ti_{31}} = 2a_{w6}ti_{11} + 2a_{w5}ti_{21} + 2a_{w3}ti_{31} \qquad \frac{\partial a_{c2}}{\partial ti_{12}} = 2a_{w1}ti_{12} + 2a_{w4}ti_{22} + 2a_{w6}ti_{32}$$

$$\frac{\partial a_{c2}}{\partial ti_{22}} = 2a_{w4}ti_{12} + 2a_{w2}ti_{22} + 2a_{w5}ti_{32} \qquad \frac{\partial a_{c2}}{\partial ti_{32}} = 2a_{w6}ti_{12} + 2a_{w5}ti_{22} + 2a_{w3}ti_{32}$$

$$\frac{\partial a_{c3}}{\partial ti_{13}} = 2a_{w1}ti_{13} + 2a_{w4}ti_{23} + 2a_{w6}ti_{33} \qquad \frac{\partial a_{c3}}{\partial ti_{23}} = 2a_{w4}ti_{13} + 2a_{w2}ti_{23} + 2a_{w5}ti_{33}$$

$$\frac{\partial a_{c3}}{\partial ti_{33}} = 2a_{w6}ti_{13} + 2a_{w5}ti_{23} + 2a_{w3}ti_{33} \qquad \frac{\partial a_{c4}}{\partial ti_{11}} = a_{w1}ti_{12} + a_{w4}ti_{22} + a_{w6}ti_{32}$$

$$\frac{\partial a_{c4}}{\partial ti_{12}} = a_{w1}ti_{11} + a_{w4}ti_{21} + a_{w6}ti_{31} \qquad \frac{\partial a_{c4}}{\partial ti_{21}} = a_{w4}ti_{12} + a_{w2}ti_{22} + a_{w5}ti_{32}$$

$$\frac{\partial a_{c4}}{\partial ti_{22}} = a_{w4}ti_{11} + a_{w2}ti_{21} + a_{w5}ti_{31} \qquad \frac{\partial a_{c4}}{\partial ti_{31}} = a_{w6}ti_{12} + a_{w5}ti_{22} + a_{w3}ti_{32}$$

$$\frac{\partial a_{c4}}{\partial ti_{32}} = a_{w6}ti_{11} + a_{w5}ti_{21} + a_{w3}ti_{31} \qquad \frac{\partial a_{c5}}{\partial ti_{12}} = a_{w1}ti_{13} + a_{w4}ti_{23} + a_{w6}ti_{33}$$

$$
\begin{aligned}
\frac{\partial a_{c5}}{\partial ti_{13}} &= a_{w1}ti_{12} + a_{w4}ti_{22} + a_{w6}ti_{32} & \frac{\partial a_{c5}}{\partial ti_{22}} &= a_{w4}ti_{13} + a_{w2}ti_{23} + a_{w5}ti_{33} \\
\frac{\partial a_{c5}}{\partial ti_{23}} &= a_{w4}ti_{12} + a_{w2}ti_{22} + a_{w5}ti_{32} & \frac{\partial a_{c5}}{\partial ti_{32}} &= a_{w6}ti_{13} + a_{w5}ti_{23} + a_{w3}ti_{33} \\
\frac{\partial a_{c5}}{\partial ti_{33}} &= a_{w6}ti_{12} + a_{w5}ti_{22} + a_{w3}ti_{32} & \frac{\partial a_{c6}}{\partial ti_{11}} &= a_{w1}ti_{13} + a_{w4}ti_{23} + a_{w6}ti_{33} \\
\frac{\partial a_{c6}}{\partial ti_{13}} &= a_{w1}ti_{11} + a_{w4}ti_{21} + a_{w6}ti_{31} & \frac{\partial a_{c6}}{\partial ti_{21}} &= a_{w4}ti_{13} + a_{w2}ti_{23} + a_{w5}ti_{33} \\
\frac{\partial a_{c6}}{\partial ti_{23}} &= a_{w4}ti_{11} + a_{w2}ti_{21} + a_{w5}ti_{31} & \frac{\partial a_{c6}}{\partial ti_{31}} &= a_{w6}ti_{13} + a_{w5}ti_{23} + a_{w3}ti_{33} \\
\frac{\partial a_{c6}}{\partial ti_{33}} &= a_{w6}ti_{11} + a_{w5}ti_{21} + a_{w3}ti_{31} & \frac{\partial b_{c1}}{\partial ti_{11}} &= a_{w1}ti_{14} + a_{w4}ti_{24} + a_{w6}ti_{34} + b_{w1} \\
\frac{\partial b_{c1}}{\partial ti_{14}} &= a_{w1}ti_{11} + a_{w4}ti_{21} + a_{w6}ti_{31} & \frac{\partial b_{c1}}{\partial ti_{21}} &= a_{w4}ti_{14} + a_{w2}ti_{24} + a_{w5}ti_{34} + b_{w2} \\
\frac{\partial b_{c1}}{\partial ti_{24}} &= a_{w4}ti_{11} + a_{w2}ti_{21} + a_{w5}ti_{31} & \frac{\partial b_{c1}}{\partial ti_{31}} &= a_{w6}ti_{14} + a_{w5}ti_{24} + a_{w5}ti_{34} + b_{w3} \\
\frac{\partial b_{c1}}{\partial ti_{34}} &= a_{w6}ti_{11} + a_{w5}ti_{21} + a_{w3}ti_{31} & \frac{\partial b_{c2}}{\partial ti_{12}} &= a_{w1}ti_{14} + a_{w4}ti_{24} + a_{w6}ti_{34} + b_{w1} \\
\frac{\partial b_{c2}}{\partial ti_{14}} &= a_{w1}ti_{12} + a_{w4}ti_{22} + a_{w6}ti_{32} & \frac{\partial b_{c2}}{\partial ti_{22}} &= a_{w4}ti_{14} + a_{w2}ti_{24} + a_{w5}ti_{34} + b_{w2} \\
\frac{\partial b_{c2}}{\partial ti_{24}} &= a_{w4}ti_{12} + a_{w2}ti_{22} + a_{w5}ti_{32} & \frac{\partial b_{c2}}{\partial ti_{32}} &= a_{w6}ti_{14} + a_{w5}ti_{24} + a_{w3}ti_{34} + b_{w3} \\
\frac{\partial b_{c2}}{\partial ti_{34}} &= a_{w6}ti_{12} + a_{w5}ti_{22} + a_{w3}ti_{32} & \frac{\partial b_{c3}}{\partial ti_{13}} &= a_{w1}ti_{14} + a_{w4}ti_{24} + a_{w6}ti_{34} + b_{w1} \\
\frac{\partial b_{c3}}{\partial ti_{14}} &= a_{w1}ti_{13} + a_{w4}ti_{23} + a_{w6}ti_{33} & \frac{\partial b_{c3}}{\partial ti_{23}} &= a_{w4}ti_{14} + a_{w2}ti_{24} + a_{w5}ti_{34} + b_{w2} \\
\frac{\partial b_{c3}}{\partial ti_{24}} &= a_{w4}ti_{13} + a_{w2}ti_{23} + a_{w5}ti_{33} & \frac{\partial b_{c3}}{\partial ti_{33}} &= a_{w6}ti_{14} + a_{w5}ti_{24} + a_{w3}ti_{34} + b_{w3} \\
\frac{\partial b_{c3}}{\partial ti_{34}} &= a_{w6}ti_{13} + a_{w5}ti_{23} + a_{w3}ti_{33} & & \\
\frac{\partial c_c}{\partial ti_{14}} &= 2a_{w1}ti_{14} + 2a_{w4}ti_{24} + 2a_{w6}ti_{34} + 2b_{w1} & & \\
\frac{\partial c_c}{\partial ti_{24}} &= 2a_{w4}ti_{14} + 2a_{w2}ti_{24} + 2a_{w5}ti_{34} + 2b_{w2} & & \\
\frac{\partial c_c}{\partial ti_{34}} &= 2a_{w6}ti_{14} + 2a_{w5}ti_{24} + 2a_{w3}ti_{34} + 2b_{w3} & &
\end{aligned}
\tag{D.23}
$$

D.4 Jacobian matrix $J_{T_iT_{Mi}}$

$J_{T_iT_{Mi}}$ corresponds to a 12x12 matrix with partial derivatives of the inverse modelview matrix $\mathbf{T_i}$ with respect to the inverse of the transformation matrix $\mathbf{T_M}$. Using Equation 2.4, matrix $\mathbf{T}_i$ is obtained by calculating:

$$\mathbf{T_i} = (\mathbf{T_0}\mathbf{T_M})^{-\mathbf{1}} = \mathbf{T_{Mi}}\mathbf{T_{0i}} \tag{D.24}$$

where

$$\mathbf{T}_{0i} = \begin{pmatrix} t0i_{11} & t0i_{12} & t0i_{13} & t0i_{14} \\ t0i_{21} & t0i_{22} & t0i_{23} & t0i_{24} \\ t0i_{31} & t0i_{32} & t0i_{33} & t0i_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{D.25}$$

and

$$\begin{aligned} \mathbf{T}_{\mathrm{Mi}} &= \begin{pmatrix} tm_{11} & tm_{12} & tm_{13} & tm_{14} \\ tm_{21} & tm_{22} & tm_{23} & tm_{24} \\ tm_{31} & tm_{32} & tm_{33} & tm_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} tmi_{11} & tmi_{12} & tmi_{13} & tmi_{14} \\ tmi_{21} & tmi_{22} & tmi_{23} & tmi_{24} \\ tmi_{31} & tmi_{32} & tmi_{33} & tmi_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} tm_{11} & tm_{21} & tm_{31} & -tm_{11}tm_{14} - tm_{21}tm_{24} - tm_{31}t_{34} \\ tm_{12} & tm_{22} & tm_{32} & -tm_{12}tm_{14} - tm_{22}tm_{24} - tm_{32}t_{34} \\ tm_{13} & tm_{23} & tm_{33} & -tm_{13}tm_{14} - tm_{23}tm_{24} - tm_{33}t_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}. \end{aligned} \tag{D.26}$$

The final expressions for $\mathbf{T}_i$ correspond to:

$$\begin{aligned} ti_{11} &= tmi_{11}t0i_{11} + tmi_{12}t0i_{21} + tmi_{13}t0i_{31}, \\ ti_{12} &= tmi_{11}t0i_{12} + tmi_{12}t0i_{22} + tmi_{13}t0i_{32}, \\ ti_{13} &= tmi_{11}t0i_{13} + tmi_{12}t0i_{23} + tmi_{13}t0i_{33}, \\ ti_{14} &= tmi_{11}t0i_{14} + tmi_{12}t0i_{24} + tmi_{13}t0i_{34} + tmi_{14}, \end{aligned}$$

$$
\begin{aligned}
ti_{21} &= tmi_{21}t0i_{11} + tmi_{22}t0i_{21} + tmi_{23}t0i_{31},\\
ti_{22} &= tmi_{21}t0i_{12} + tmi_{22}t0i_{22} + tmi_{23}t0i_{32},\\
ti_{23} &= tmi_{21}t0i_{13} + tmi_{22}t0i_{23} + tmi_{23}t0i_{33},\\
ti_{24} &= tmi_{21}t0i_{14} + tmi_{22}t0i_{24} + tmi_{23}t0i_{34} + tmi_{24},\\
ti_{31} &= tmi_{31}t0i_{11} + tmi_{32}t0i_{21} + tmi_{33}t0i_{31},\\
ti_{32} &= tmi_{31}t0i_{12} + tmi_{32}t0i_{22} + tmi_{33}t0i_{32},\\
ti_{33} &= tmi_{31}t0i_{13} + tmi_{32}t0i_{23} + tmi_{33}t0i_{33},\\
ti_{34} &= tmi_{31}t0i_{14} + tmi_{32}t0i_{24} + tmi_{33}t0i_{34} + tmi_{34}.
\end{aligned}
\tag{D.27}
$$

Then, $J_{T_iT_{Mi}}$ can be calculated as:

$$
\begin{aligned}
J_{T_iT_{Mi}} &= \begin{pmatrix}
\frac{\partial ti_{11}}{\partial tmi_{11}} & \cdots & \frac{\partial ti_{11}}{\partial tmi_{34}} \\
\vdots & \ddots & \vdots \\
\frac{\partial ti_{33}}{\partial tmi_{11}} & \cdots & \frac{\partial ti_{33}}{\partial tmi_{34}} \\
\frac{\partial ti_{34}}{\partial tmi_{11}} & \cdots & \frac{\partial ti_{34}}{\partial tmi_{34}}
\end{pmatrix} \\
&= \begin{pmatrix}
t0i_{11} & t0i_{21} & t0i_{31} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
t0i_{12} & t0i_{22} & t0i_{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
t0i_{13} & t0i_{23} & t0i_{33} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
t0i_{14} & t0i_{24} & t0i_{34} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & t0i_{11} & t0i_{21} & t0i_{31} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & t0i_{12} & t0i_{22} & t0i_{32} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & t0i_{13} & t0i_{23} & t0i_{33} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & t0i_{14} & t0i_{24} & t0i_{34} & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t0i_{11} & t0i_{21} & t0i_{31} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t0i_{12} & t0i_{22} & t0i_{32} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t0i_{13} & t0i_{23} & t0i_{33} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t0i_{14} & t0i_{24} & t0i_{34} & 1
\end{pmatrix}.
\end{aligned}
\tag{D.28}
$$

D.5 Jacobian matrix $J_{T_{Mi}s}$

$J_{T_{Mi}\mathbf{s}}$ corresponds to a 12x6 matrix representing the partial derivatives of $\mathbf{T}_{\mathrm{Mi}}$, which is calculated using the following expression:

$$J_{T_{Mi}s} = J_{T_{Mi}T_M} J_{T_M\mathbf{s}}, \tag{D.29}$$

where $J_{T_{Mi}T_M}$ is a 12x12 matrix representing the partial derivatives of T_{Mi} with respect to T_M, calculated using Equation D.26:

$$J_{T_{Mi}T_M} = \begin{pmatrix} \frac{\partial tmi_{11}}{\partial tm_{11}} & \frac{\partial tmi_{11}}{\partial tm_{12}} & \cdots & \frac{\partial tmi_{11}}{\partial tm_{34}} \\ \frac{\partial tmi_{12}}{\partial tm_{11}} & \frac{\partial tmi_{12}}{\partial tm_{12}} & \cdots & \frac{\partial tmi_{12}}{\partial tm_{34}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial tmi_{34}}{\partial tm_{11}} & \frac{\partial tmi_{34}}{\partial tm_{12}} & \cdots & \frac{\partial tmi_{34}}{\partial tm_{34}} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -tm_{14} & 0 & 0 & -tm_{11} & -tm_{24} & 0 & 0 & -tm_{21} & -tm_{34} & 0 & 0 & -tm_{31} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -tm_{14} & 0 & -tm_{12} & 0 & -tm_{24} & 0 & -tm_{22} & 0 & -tm_{34} & 0 & -tm_{32} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -tm_{14} & -tm_{13} & 0 & 0 & -tm_{24} & -tm_{23} & 0 & 0 & -tm_{34} & -tm_{33} \end{pmatrix}. \tag{D.30}$$

and $J_{T_M\mathbf{s}}$ is a 12x6 matrix of partial derivatives of $\mathbf{T_M}$ with respect to the pose parameters vector s, represented by:

$$J_{T_Ms} = \begin{pmatrix} \frac{\partial tm_{11}}{\partial w_x} & \frac{\partial tm_{11}}{\partial w_y} & \cdots & \frac{\partial tm_{11}}{\partial t_z} \\ \frac{\partial tm_{12}}{\partial w_x} & \frac{\partial tm_{12}}{\partial w_y} & \cdots & \frac{\partial tm_{12}}{\partial t_z} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial tm_{34}}{\partial w_x} & \frac{\partial tm_{34}}{\partial w_y} & \cdots & \frac{\partial tm_{34}}{\partial t_z} \end{pmatrix} = \begin{pmatrix} & & & \frac{\partial r_{11}}{\partial t_x} & \frac{\partial r_{11}}{\partial t_y} & \frac{\partial r_{11}}{\partial t_z} \\ & \frac{\partial \mathbf{R}}{\partial \mathbf{W}} & & \vdots & \vdots & \vdots \\ & & & \frac{\partial r_{33}}{\partial t_x} & \frac{\partial r_{33}}{\partial t_y} & \frac{\partial r_{33}}{\partial t_z} \\ \frac{\partial t_x}{\partial w_x} & \frac{\partial t_x}{\partial w_y} & \frac{\partial t_x}{\partial w_z} & \frac{\partial t_x}{\partial t_x} & \frac{\partial t_x}{\partial t_y} & \frac{\partial t_x}{\partial t_z} \\ \frac{\partial t_y}{\partial w_x} & \frac{\partial t_y}{\partial w_y} & \frac{\partial t_y}{\partial w_z} & \frac{\partial t_y}{\partial t_x} & \frac{\partial t_y}{\partial t_y} & \frac{\partial t_y}{\partial t_z} \\ \frac{\partial t_z}{\partial w_x} & \frac{\partial t_z}{\partial w_y} & \frac{\partial t_z}{\partial w_z} & \frac{\partial t_z}{\partial t_x} & \frac{\partial t_z}{\partial t_y} & \frac{\partial t_z}{\partial t_z} \end{pmatrix}. \tag{D.31}$$

To calculate $\frac{\partial \mathbf{R}}{\partial \mathbf{W}}$, first the Jacobian matrix relating the parameters of matrix $\mathbf{R}$ in Equation 2.3 with respect to the the vector $\mathbf{Q} = (r_x\, r_y\, r_z\, r_a); r_a = \theta$ is calculated as:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{Q}} = \begin{pmatrix} \frac{\partial r_{11}}{\partial r_x} & \frac{\partial r_{11}}{\partial r_y} & \frac{\partial r_{11}}{\partial r_z} & \frac{\partial r_{11}}{\partial r_a} \\ \frac{\partial r_{21}}{\partial r_x} & \frac{\partial r_{21}}{\partial r_y} & \frac{\partial r_{21}}{\partial r_z} & \frac{\partial r_{21}}{\partial r_a} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_{33}}{\partial r_x} & \frac{\partial r_{33}}{\partial r_y} & \frac{\partial r_{33}}{\partial r_z} & \frac{\partial r_{33}}{\partial r_a} \end{pmatrix}$$

$$= \begin{pmatrix} 2(1-\cos\theta)r_x & 0 & 0 & (r_x^2-1)\sin\theta \\ r_y(1-\cos\theta) & r_x(1-\cos\theta) & -\sin\theta & r_x r_y \sin\theta - r_z\cos\theta \\ r_z(1-\cos\theta) & \sin\theta & r_x(1-\cos\theta) & r_x r_z \sin\theta + r_y\cos\theta \\ r_y(1-\cos\theta) & r_x(1-\cos\theta) & \sin\theta & r_x r_y \sin\theta + r_z\cos\theta \\ 0 & 2(1-\cos\theta)r_y & 0 & (r_y^2-1)\sin\theta \\ -\sin\theta & r_z(1-\cos\theta) & r_y(1-\cos\theta) & r_y r_z \sin\theta - r_x\cos\theta \\ r_z(1-\cos\theta) & -\sin\theta & r_x(1-\cos\theta) & r_x r_z \sin\theta - r_y\cos\theta \\ \sin\theta & r_z(1-\cos\theta) & r_y(1-\cos\theta) & r_y r_z \sin\theta + r_x\cos\theta \\ 0 & 0 & 2(1-\cos\theta)r_z & (r_z^2-1)\sin\theta \end{pmatrix}. \tag{D.32}$$

The Jacobian relating the parameters of matrix $\mathbf{R}$ in Equation2.3 and represented by the vector $\mathbf{Q} = (r_x\, r_y\, r_z\, r_a); r_a = \theta$ with respect to the parameters of the vector $\mathbf{W}$ results in:

$$\frac{\partial \mathbf{Q}}{\partial \mathbf{W}} = \begin{pmatrix} \frac{\partial r_x}{\partial w_x} & \frac{\partial r_x}{\partial w_y} & \frac{\partial r_x}{\partial w_z} \\ \frac{\partial r_y}{\partial w_x} & \frac{\partial r_y}{\partial w_y} & \frac{\partial r_y}{\partial w_z} \\ \frac{\partial r_z}{\partial w_x} & \frac{\partial r_z}{\partial w_y} & \frac{\partial r_z}{\partial w_z} \\ \frac{\partial r_a}{\partial w_x} & \frac{\partial r_a}{\partial w_y} & \frac{\partial r_a}{\partial w_z} \end{pmatrix} = \begin{pmatrix} \frac{1}{\theta}(1-x^2) & \frac{-xy}{\theta} & \frac{-xz}{\theta} \\ \frac{-xy}{\theta} & \frac{1}{\theta}(1-y^2) & \frac{-yz}{\theta} \\ \frac{-xz}{\theta} & \frac{-yz}{\theta} & \frac{1}{\theta}(1-z^2) \\ r_x & r_y & r_z \end{pmatrix}. \tag{D.33}$$

Finally, $\frac{\partial \mathbf{R}}{\partial \mathbf{W}}$ is calculated as:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{W}} = \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \frac{\partial \mathbf{Q}}{\partial \mathbf{W}}, \tag{D.34}$$

using the following expressions:

$$\frac{\partial r_{11}}{\partial w_x} = 2r_x(1-r_x^2)\frac{(1-\cos\theta)}{\theta} + r_x(r_x^2-1)\sin\theta$$
$$\frac{\partial r_{11}}{\partial w_y} = -2r_x^2 r_y\frac{(1-\cos\theta)}{\theta} + r_y(r_x^2-1)\sin\theta$$
$$\frac{\partial r_{11}}{\partial w_z} = -2r_x^2 r_z\frac{(1-\cos\theta)}{\theta} + r_z(r_x^2-1)\sin\theta$$
$$\frac{\partial r_{12}}{\partial w_x} = \frac{(1-\cos\theta)}{\theta}r_y(1-r_x^2) - \frac{(1-\cos\theta)}{\theta}r_x^2 r_y + \frac{\sin\theta}{\theta}r_x r_z + r_x^2 r_y\sin\theta - r_x r_z\cos\theta$$
$$\frac{\partial r_{12}}{\partial w_y} = -r_x r_y^2\frac{(1-\cos\theta)}{\theta} + \frac{(1-\cos\theta)}{\theta}r_x(1-r_y^2) + \frac{\sin\theta}{\theta}r_y r_z + r_x r_y^2\sin\theta - r_y r_z\cos\theta$$
$$\frac{\partial r_{12}}{\partial w_z} = -r_x r_y r_z\frac{(1-\cos\theta)}{\theta} - r_x r_y r_z\frac{(1-\cos\theta)}{\theta} - \frac{\sin\theta}{\theta}(1-r_z^2) + r_x r_y r_z\sin\theta - r_z^2\cos\theta$$
$$\frac{\partial r_{13}}{\partial w_x} = \frac{(1-\cos\theta)}{\theta}r_z(1-r_x^2) - r_x r_y\frac{\sin\theta}{\theta} - r_x^2 r_z\frac{(1-\cos\theta)}{\theta} + r_x^2 r_z\sin\theta + r_x r_y\cos\theta$$
$$\frac{\partial r_{13}}{\partial w_y} = -r_x r_y r_z\frac{(1-\cos\theta)}{\theta} + \frac{\sin\theta}{\theta}(1-r_y^2) - r_x r_y r_z\frac{(1-\cos\theta)}{\theta} + r_x r_y r_z\sin\theta + r_y^2\cos\theta$$
$$\frac{\partial r_{13}}{\partial w_z} = -r_x r_z^2\frac{(1-\cos\theta)}{\theta} - r_y r_z\frac{\sin\theta}{\theta} + \frac{(1-\cos\theta)}{\theta}r_x(1-r_z^2) + r_x r_z^2\sin\theta + r_y r_z\cos\theta$$
$$\frac{\partial r_{21}}{\partial w_x} = \frac{(1-\cos\theta)}{\theta}r_y(1-r_x^2) - r_x^2 r_y\frac{(1-\cos\theta)}{\theta} - r_x r_z\frac{\sin\theta}{\theta} + r_x^2 r_y\sin\theta + r_x r_z\cos\theta$$
$$\frac{\partial r_{21}}{\partial w_y} = -r_x r_y^2\frac{(1-\cos\theta)}{\theta} + \frac{(1-\cos\theta)}{\theta}r_x(1-r_y^2) - r_y r_z\frac{\sin\theta}{\theta} + r_x r_y^2\sin\theta + r_y r_z\cos\theta$$
$$\frac{\partial r_{21}}{\partial w_z} = -r_x r_y r_z\frac{(1-\cos\theta)}{\theta} - r_x r_y r_z\frac{(1-\cos\theta)}{\theta} + \frac{\sin\theta}{\theta}(1-r_z^2) + r_x r_y r_z\sin\theta + r_z^2\cos\theta$$
$$\frac{\partial r_{22}}{\partial w_x} = -2r_x r_y^2\frac{(1-\cos\theta)}{\theta} + r_x(r_y^2-1)\sin\theta$$
$$\frac{\partial r_{22}}{\partial w_y} = 2r_y(1-r_y^2)\frac{(1-\cos\theta)}{\theta} + r_y(r_y^2-1)\sin\theta$$
$$\frac{\partial r_{22}}{\partial w_z} = -2r_y^2 r_z\frac{(1-\cos\theta)}{\theta} + r_z(r_y^2-1)\sin\theta$$
$$\frac{\partial r_{23}}{\partial w_x} = -\frac{\sin\theta}{\theta}(1-r_x^2) - r_x r_y r_z\frac{(1-\cos\theta)}{\theta} - r_x r_y r_z\frac{(1-\cos\theta)}{\theta} + r_x r_y r_z\sin\theta - r_x^2\cos\theta$$
$$\frac{\partial r_{23}}{\partial w_y} = r_x r_y\frac{\sin\theta}{\theta} + \frac{(1-\cos\theta)}{\theta}r_z(1-r_y^2) - r_y^2 r_z\frac{(1-\cos\theta)}{\theta} + r_y^2 r_z\sin\theta - r_x r_y\cos\theta$$
$$\frac{\partial r_{23}}{\partial w_z} = r_x r_z\frac{\sin\theta}{\theta} + -r_y r_z^2\frac{(1-\cos\theta)}{\theta} + \frac{(1-\cos\theta)}{\theta}r_y(1-r_z^2) + r_y r_z^2\sin\theta - r_x r_z\cos\theta$$

$$\frac{\partial r_{31}}{\partial w_x} = \frac{(1-\cos\theta)}{\theta} r_z(1-r_x^2) + r_x r_y \frac{\sin\theta}{\theta} - r_x^2 r_z \frac{(1-\cos\theta)}{\theta} + r_x^2 r_z \sin\theta - r_x r_y \cos\theta$$

$$\frac{\partial r_{31}}{\partial w_y} = -r_x r_y r_z \frac{(1-\cos\theta)}{\theta} - \frac{\sin\theta}{\theta}(1-r_y^2) - r_x r_y r_z \frac{(1-\cos\theta)}{\theta} + r_x r_y r_z \sin\theta - r_y^2 \cos\theta$$

$$\frac{\partial r_{31}}{\partial w_z} = -r_x r_z^2 \frac{(1-\cos\theta)}{\theta} + r_y r_z \frac{\sin\theta}{\theta} + \frac{(1-\cos\theta)}{\theta} r_x(1-r_z^2) + r_x r_z^2 \sin\theta - r_y r_z \cos\theta$$

$$\frac{\partial r_{32}}{\partial w_x} = \frac{\sin\theta}{\theta}(1-r_x^2) - r_x r_y r_z \frac{(1-\cos\theta)}{\theta} - r_x r_y r_z \frac{(1-\cos\theta)}{\theta} + r_x r_y r_z \sin\theta + r_x^2 \cos\theta$$

$$\frac{\partial r_{32}}{\partial w_y} = -r_x r_y \frac{\sin\theta}{\theta} + \frac{(1-\cos\theta)}{\theta} r_z(1-r_y^2) - r_y^2 r_z \frac{(1-\cos\theta)}{\theta} + r_y^2 r_z \sin\theta - r_x r_y \cos\theta$$

$$\frac{\partial r_{32}}{\partial w_z} = -r_x r_z \frac{\sin\theta}{\theta} + -r_y r_z^2 \frac{(1-\cos\theta)}{\theta} + \frac{(1-\cos\theta)}{\theta} r_y(1-r_z^2) + r_y r_z^2 \sin\theta + r_x r_z \cos\theta$$

$$\frac{\partial r_{33}}{\partial w_x} = -2 r_x r_z^2 \frac{(1-\cos\theta)}{\theta} + r_x(r_z^2 - 1)\sin\theta$$

$$\frac{\partial r_{33}}{\partial w_y} = -2 r_y r_z^2 \frac{(1-\cos\theta)}{\theta} + r_y(r_z^2 - 1)\sin\theta$$

$$\frac{\partial r_{33}}{\partial w_z} = 2 r_z(1-r_z^2) \frac{(1-\cos\theta)}{\theta} + r_z(r_z^2 - 1)\sin\theta \tag{D.35}$$

Finally, the Jacobian matrix is calculated for $\mathbf{W} = (0, 0, 0)$. In this case,

$$\theta = \sqrt{0^2 + 0^2 + 0^2} = 0$$

and

$$|w_i| \leq \sqrt{w_1^2 + w_2^2 + w_3^2};\ i = 1, 2, 3.$$

Then,

$$\frac{|w_i|}{\sqrt{w_1^2 + w_2^2 + w_3^2}} \leq 1;\ i = 1, 2, 3.$$

Therefore:

$$|r_x| \leq 1, |r_y| \leq 1, |r_z| \leq 1.$$

Furthermore, considering:

$$\lim_{x \to 0} \frac{\sin\theta}{\theta} = 1 \text{ and } \lim_{x \to 0} \frac{1-\cos\theta}{\theta} = 0,$$

the Jacobian matrix results in:

$$J_{T_M\mathbf{s}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{D.36}$$

Bibliography

[1] Sony. EyePet. http://www.eyepet.com/, 2009. Accessed in November, 2012.

[2] Sony. The eye of judgment. http://us.playstation.com/games-and-media/games/the-eye-of-judgment-ps3.html. Accessed in November, 2012.

[3] Nintendo. AR cards. http://www.nintendo.com/3ds/ar-cards. Accessed in November, 2012.

[4] I. Barakonyi, T. Psik, and D. Schmalstieg. Agents that talk and hit back: Animated agents in augmented reality. In *Proceedings of The 3rd IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR04)*, pages 141–150, 2004.

[5] H. Alvarez, I. Aguinaga, and D. Borro. Providing guidance for maintenance operations using automatic markerless augmented reality system. In *Proceedings of the 10th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR11)*, pages 181–190, 2011.

[6] S. J. Henderson and S. Feiner. Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. In *Proceedings of The 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR09)*, pages 135–144, 2009.

[7] Zs. Szalavari, D. Schmalstieg, A. Fuhrmann, and M. Gervautz. Studierstube - an environment for collaboration in augmented reality. *Journal of the Virtual Reality Society*, 3(1):37–48, 1998.

[8] H. Kaufmann, K. Steinbugl, A. Dunser, and J. Gluck. Improving spatial abilities by geometry education in augmented reality - application and evaluation design. In *7th Virtual Reality International Conference (VRIC - Laval Virtual 2005)*, pages 25–34, 2005.

[9] B. Schwald, H. Seibert, and T. Weller. A flexible tracking concept applied to medical scenarios using an ar window. In *Proceedings of The 1st International Symposium on Mixed and Augmented Reality (ISMAR02)*, 2002.

[10] C. Bichlmeier, F. Wimmer, S. M. Heining, and N. Navab. Contextual anatomic mimesis: Hybrid in-situ visualization method for improving multi-sensory depth perception in medical augmented reality. In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR07)*, pages 129–138, 2007.

[11] Y. Yamaguchi, T. Nakagawa, K. Akaho, M. Honda, H. Kato, and S. Nishida. AR-Navi: An in-vehicle navigation system using video-based augmented reality technology. In *Symposium on Human Interface 2007*, volume 4558, pages 1139–1147. Springer-Verlag Berlin Heidelberg, 2007.

[12] Pioneer. Cyber-Navi. http://pioneer.jp/carrozzeria/cybernavi/, 2012. Accessed in November, 2012.

[13] Wikitude GmbH. What is Wikitude? http://www.wikitude.com/tour/wikitude-world-browser. Accessed in November, 2012.

[14] Metaio Inc. Junaio. http://www.junaio.com/home/. Accessed in November, 2012.

[15] R. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

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

[17] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proceedings of The 2nd IEEE International Workshop on Augmented Reality*, pages 85–94, 1999.

[18] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, 2002.

[19] 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*, 12(4):615–628, 2006.

[20] G. Klein and D. Murray. Parallel Tracking and Mapping for small AR workspaces. In *Proceedings of The 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR07)*, pages 225–234, 2007.

[21] S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Proceedings of IEEE Virtual Reality 1999*, pages 260–267, 1999.

[22] L. Naimark and E. Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality (ISMAR02)*, pages 27–36, 2002.

[23] M. Ribo, P. Lang, H. Ganster, M. Brandner, C. Stock, and A. Pinz. Hybrid tracking for outdoor augmented reality applications. *IEEE Computer Graphics and Applications*, 22(6):54–63, 2002.

[24] G. Reitmayr and T. Drummond. Going out: Robust model-based tracking for outdoor augmented reality. In *Proceedings of the Fifth International Symposium on Mixed and Augmented Reality (ISMAR05)*, 2006.

[25] V. Teichrieb, J. P. S. M. Lima, E. L. Apolinario, T. S. M. C. de Farias, M. A. S. Bueno, J. Kelner, and I. H. F. Santos. A survey of online monoc-

ular markerless augmented reality. *International Journal of Modeling and Simulation for the Petroleum Industry*, 1(1), 2007.

[26] Borko Furht (Ed.). *Handbook of Augmented Reality*. Springer, 2011.

[27] 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*, pages 315–320, 2001.

[28] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, 1991.

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

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

[31] 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.

[32] J. Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *Proceedings of Asia Pacific Computer-Human Interaction*, pages 63–68, 1998.

[33] D. Wagner. *Handheld Augmented Reality*. PhD thesis, Graz University of Technology, 2007.

[34] D. Wagner and D. Schmalstieg. ARToolKitPlus for pose tracking on mobile devices. In *Proceedings of the 12th Computer Vision Winter Workshop (CVWW07)*, pages 139–146, 2007.

[35] J. Park, S. You, and U. Neuman. Natural feature tracking for extendible robust augmented realities. In *Proceedings of the International Workshop on Augmented Reality (IWAR98)*, pages 209–217, 1998.

[36] K. W. Chia, A. D. Cheok, and S. J. D. Prince. Online 6DOF augmented reality registration from natural features. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality (ISMAR02)*, pages 305–313, 2002.

[37] Y. Park, V. Lepetit, and W. Woo. Texture-less object tracking with online training using an RGB-D camera. In *Proceedings of the 10th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR2011)*, pages 121–126, 2011.

[38] K. Hirose and H. Saito. Fast line description for line-based SLAM. In *Proceedings of the British Machine Vision Conference 2012*, pages 83.1–83.11, 2012.

[39] J. Rekimoto and Y. Ayatsuka. CyberCode: designing augmented reality environments with visual tags. In *Proceedings of the 2000 ACM Conference on Designing Augmented Reality Environments (DARE2000)*, pages 1–10, 2000.

[40] M. Fiala. ARTag, a fiducial marker system using digital techniques. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05)*, volume 2, pages 590–596, 2005.

[41] R. Bencina and M. Kaltenbrunner. The design and evolution of fiducials for the reacTIVision system. In *Proceedings of the 3rd International Conference on Generative Systems in the Eletronic Arts*, 2005.

[42] J. Koehler, A. Pagani, and D. Stricker. Robust detection and identification of partially occluded circular markers. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP2010)*, pages 387–392, 2010.

[43] E. Constanza and J. Robinson. A region adjacency tree approach to the detection and design of fiducials. In *Proceedings of Video, Vision and Graphics (VVG2003)*, pages 63–69, 2003.

[44] J. Koehler, A. Pagani, and D. Stricker. Detection and identification techniques for markers used in computer vision. In *Visualization of Large and*

Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG Workshop), pages 36–44, 2010.

[45] H. Nishino. A 6DOF fiducial tracking method based on topological region adjacency and angle information for tangible interaction. In *Proceedings of the 4th International Conference on Tangible, Embedded and Embodied Interaction*, pages 253–256, 2010.

[46] X. Zhang, S. Fronz, and N. Navab. Visual marker detection and decoding in ar systems: A comparative study. In *Proceedings of 1st the International Symposium on Mixed and Augmented Reality (ISMAR02)*, pages 97–108, 2002.

[47] J. P. Lima, F. Simoes, L. Figueiredo, and J. Kelner. Model based markerless tracking applied to augmented reality. *SBC Journal on 3D Interactive Systems*, 1:2–15, 2010.

[48] S. B. Gokturk, J.-Y. Bouguet, and R. Grzeszczuk. A data-driven model for monocular face tracking. In *Proceedings of the 8th IEEE International Conference on Computer Vision*, volume 2, pages 701–708, 2001.

[49] H. Li, P. Roivainen, and R. Forchheimer. 3D motion estimation in model-based facial image coding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6), 1993.

[50] S. Basu, I. Essa, and A. Pentland. Motion regularization for model-based head-tracking. In *Proceedings of the 13th IEEE Internation Conference in Pattern Recognition (ICPR96)*, pages 611–616, 1996.

[51] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[52] F. Jurie and M. Dhome. Real-time 3D template matching. In *Proceedings of The 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR01)*, volume 1, pages 791–796, 2001.

[53] F. Jurie and M. Dhome. Real-time robust template matching. In *Proceedings of the 13th British Machine Vision Conference (BMVC02)*, pages 123–132, 2002.

[54] C. Harris and M. Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, pages 147–151, 1988.

[55] S. M. Smith and J. M. Brady. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[56] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of the 9th European Conference on Computer Vision (ECCV06)*, pages 430–443, 2006.

[57] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[58] H. Bay, T. Tuytelaars, and L. van Gool. SURF: Speeded-Up Robust Features. In *Proceedings of the 9th European Conference on Computer Vision (ECCV06)*, pages 404–417, 2006.

[59] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[60] M. Haag and H.-H. Nagel. Combination of edge element and optical flow estimates for 3D model-based vehicle tracking in traffic image sequences. *International Journal of Computer Vision*, 35(3):295–319, 1999.

[61] T. Brox, B. Rosenhahn, D. Cremers, and H.-P. Seidel. High accuracy optical flow serves 3D pose tracking: Exploiting contour and flow based constraints. In *Proceedings of the 9th European Conference on Computer Vision (ECCV2006)*, pages 98–111, 2006.

[62] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3D camera tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, pages 48–57, 2004.

[63] L. Vacchetti, V. Lepetit, and P. Fua. Fusing online and offline information for stable 3d tracking in real-time. In *Proceedings of The International Conference on Computer Vision and Pattern Recognition (CVPR03)*, pages 241–248, 2003.

[64] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, pages 1508–1511, 2005.

[65] C. Choi and H. I. Christensen. Real-time model-based tracking using edge and keypoint features for robotic manipulation. In *Proceedings of The IEEE International Conference on Robotics and Automation*, pages 4048–4055, 2010.

[66] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[67] M. Pollefeys and L. Van Gool. From images to 3D models. *Communications of the ACM*, 45(7):50–55, 2002.

[68] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008.

[69] R. Koch, K. Koeser, B. Streckel, and J. F. Evers-Senne. Markerless image-based 3D tracking for real-time augmented reality applications. In *Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, 2005.

[70] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV03)*, volume 2, pages 1403–1410, 2003.

[71] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6), 2007.

[72] E. Eade and T. Drummond. Scalable monocular SLAM. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR06)*, pages 469–476, 2006.

[73] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using fastslam. In *IEEE International Conference on Robotics and Automation (ICRA2003)*, pages 1985–1991, 2003.

[74] G. Klein and D. Murray. Parallel Tracking and Mapping on a camera phone. In *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR09)*, pages 83–86, 2009.

[75] M. Tamaazousti, V. Gay-Bellile, S. N. Collette, and S. Bourgeois. Real-time accurate localization in a partially known environment: Application to Augmented Reality on textureless 3D objects. In *Proceedings of the International workshop on AR/MR registration, tracking and benchmarking (TrakMark2011)*, 2011.

[76] R. A. Newcombe, S. J. Lovegrove, and A. J. Davidson. DTAM: Dense Tracking and Mapping in real-time. In *Proceedings of the 2011 IEEE International Conference on Computer Vision (DTAM2011)*, pages 2320–2327, 2011.

[77] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davidson, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the 10th IEEE and ACMI nternational Symposium on Mixed and Augmented Reality (ISMAR 2011)*, pages 127–136, 2011.

[78] G. Bleser, H. Wuest, and D. Stricker. Online camera pose estimation in partially known and dynamic scenes. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR06)*, pages 56–65, 2006.

[79] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4), 2006.

[80] D. G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, 1992.

[81] D. B. Gennery. Visual tracking of known three-dimensional objects. *International Journal of Computer Vision*, 7(1):243–270, 1992.

[82] D. Koller, K. Daniilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, 1993.

[83] M. Armstrong and A. Zisserman. Robust object tracking. In *Proceedings of Asian Conference on Computer Vision*, pages 58–62, 1995.

[84] C. Harris and C. Stennettt. RAPID: a video rate object tracker. In *Proceedings of the 1st British Machine Vision Conference*, pages 73–77, 1990.

[85] R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 82:35–45, 1960.

[86] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[87] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2D-3D model-based approach. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, volume 1, pages 262–268, 1999.

[88] R. Cipolla and A. Blake. Surface shape from the deformation of apparent contours. *International Journal of Computer Vision*, pages 83–112, 1992.

[89] 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*, 28(2):302–315, 2006.

[90] S.D. Ma. Conics-based stereo, motion estimation and pose determination. *International Journal of Computer Vision*, 10(1):7–25, 1993.

[91] T. Joshi, N. Ahuja, and J. Ponce. Structure and motion estimation from dynamic silhouettes under perspective projection. *International Journal of Computer Vision*, 31(1):31–50, 1999.

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

[93] G. Li, Y. Tsin, and Y. Genc. Exploiting occluding contours for real-time 3D tracking: A unified approach. In *Proceedings of the 11th IEEE International Conference on Computer Vision*, 2007.

[94] P. Azad, D. Munch, T. Asfour, and R. Dillmann. 6-DOF model-based tracking of arbitrarily shaped 3D objects. In *IEEE International Conference on Robotics and Automation*, pages 5204–5209, 2011.

[95] V. Kyrki and D. Kragic. Tracking rigid objects using integration of model-based and model-free cues. *Machine Vision and Applications*, 22(2):323–335, 2011.

[96] B. Stenger, P.R.S. Mendonca, and R. Cipolla. Model-based 3D tracking of an articulated hand. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 310–315, 2001.

[97] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[98] R. Plaenkers and P. Fua. Model-based silhouette extraction for accurate people tracking. In *In Proceedings of the 7th European Conference on Computer Vision*, pages 325–339, 2002.

[99] N. Khan. Silhouette-based 2D-3D pose estimation using implicit algebraic surfaces. Master's thesis, Saarland University, 2007.

[100] D. P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001.

[101] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. *Recent Advances in Remeshing of Surfaces*. Springer Berlin Heidelberg, 2008.

[102] C. Barton. Poly reducer. http://wiki.blender.org/index.php/Extensions:2.4/Py/Scripts/Mesh/Mesh_poly_reduce, 2006. Accessed in June, 2010.

[103] M. Garland. *Quadric-Based Polygonal Surface Simplification.* PhD thesis, Carnegie Mellon University, 1999.

[104] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998. Available at: http://vcg.iei.pi.cnr.it/software.php.

[105] S. Petitjean. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys*, 2(34):169–201, 2002.

[106] M. A. Oikawa. Apparent contour tracking of rigid curved objects based on quadrics approximation of the surface. Master's thesis, Nara Institute of Science and Technology, 2010.

[107] M. Rouhani and A.D. Sappa. A novel approach to geometric fitting of implicit quadrics. In *IEEE International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 121–132, 2009.

[108] F. Dunn and I. Parberry. *3D Math Primer for Graphics and Game Development.* Wordware Publishing, Inc., 2002.

[109] V. H Mederos, J.C.E Sarlabous, and P.B. Sanchez. A new algorithm to compute the euclidean distance from a point to a conic. *Revista Investigacion Operacional*, 23(2), 2002.

[110] G. Taubin. Distance approximation for rasterizing implicit curves. *ACM Transactions on Graphics*, 13(1):3–42, 1994.

[111] K. Kumagai. Robust model-based tracking for measurable DOF change. Master's thesis, Nara Institute of Science and Technology, 2012.

[112] 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.

[113] 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.

[114] J. J. Broek, W. Sleijffers, I. Horvath, and A. F. Lennings. Using physical models in design. In *Proceedings of The Third International Conference on Computer Aided Industrial Design and Computer Aided Conceptual Design*, 2000.

[115] S. H. Choi and A. M. M. Chan. A virtual prototyping system for rapid product development. *Computer-Aided Design*, 36(5):401–412, 2004.

[116] G. G. Wang. Definition and review of virtual prototyping. *Journal of Computing and Information Science in Engineering*, 2(3), 2002.

[117] G. R. Bennett. The application of virtual prototyping in the development of complex aerospace products. *Aircraft Engineering and Aerospace Technology*, 69(1):19–25, 1997.

[118] A. Kulkarni, A. Kapoor, M. Iyer, and V. Kosse. Virtual prototyping used as validation tool in automotive design. In *Proceeedings of The 19th International Congress on Modelling and Simulation*, pages 419–425, 2011.

[119] M. Bordegoni and C. Rizzi. *Innovation in Product Design - From CAD to Virtual Prototyping*. Springer, 1st edition, 2011.

[120] 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.

[121] P. S. Dunston, X. Wang, M. Billinghurst, and B. Hampson. Mixed reality benefits for design perception. In *Proceedings of The 19th International Symposium on Automation and Robotics in Construction*, pages 191–196, 2002.

[122] 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*, pages 106–109, 2005.

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

[124] O. Korkalo, M. Aittala, and S. Siltanen. Light-weight marker hiding for augmented reality. In *Proceedings of The IEEE International Symposium on Mixed and Augmented Reality 2010*, pages 247–248, 2010.

[125] N. Kawai, M. Yamasaki, T. Sato, and N. Yokoya. AR marker hiding based on image inpainting and reflection of illumination changes. In *Proceedings of The IEEE International Symposium on Mixed and Augmented Reality 2012*, pages 293–294, 2012.

[126] J. Herling and W. Broll. Pixmix: A real-time approach to high-quality diminished reality. In *Proceedings of The IEEE International Symposium on Mixed and Augmented Reality 2012*, pages 141–150, 2012.

[127] S. R. Porter, M. R. Marner, R. T. Smith, J. E. Zucco, and B. H. Thomas. Spatial augmented reality for interactive rapid prototyping. In *Proceedings of the 20th International Conference on Artificial Inteligence and Telexistence*, pages 110–117, 2010.

[128] V. Gay-Bellile, S. Bourgeois, M. Tamaazousti, and S. N. Collette. A mobile markerless Augmented Reality system for the automotive field. In *Workshop on Tracking Methods and Applications*, 2012.

[129] Canon Global. Canon launches new MR (Mixed Reality) system, contributing to shorter development times during product design. http://www.canon.com/news/2012/jun18e.html, 2012. Accessed in January, 2013.

[130] T. Tullis and B. Albert. *Measuring the User Experience: Collecting, Analyzing and Presenting Usability Metrics*. Morgan Kaufmann, 2008.