UNIVERSITY OF NAVARRA

SCHOOL OF ENGINEERING

DONOSTIA-SAN SEBASTIÁN

Study of Augmented Reality Methods for Real Time Recognition and Tracking of Untextured 3D Models in Monocular Images

DISSERTATION
submitted for the
Degree of Doctor of Philosophy
of the University of Navarra by

Hugo Álvarez Ponga

under the supervision of

Diego Borro Yagüez

Dec, 2011

A mi familia.

Agradecimientos

Esta tesis me ha aportado numerosos conocimientos técnicos, y lo que es aún más importante, ha confirmado mi suerte al darme la posibilidad de descubrir el grupo humano tan increíble que tengo a mi alrededor. De hecho, las siguientes palabras van dedicadas a todas aquellas personas que me han ayudado. Aun sabiendo que la deuda contraída perdurará, gracias, gracias y gracias.

En primer lugar agradezco al Gobierno Vasco la concesión de una de sus becas doctorales, más concretamente la financiación obtenida a través del *Programa de Formación de Personal Investigador del Departamento de Educación, Universidades e Investigación.*

Doy las gracias a Alejo Avello, Jordi Viñolas y Luis Matey por apostar en mí y dejarme realizar los estudios de doctorado en el área de simulación del CEIT. Esta gratitud es extensible a la Universidad de Navarra, y especialmente al TECNUN, por la formación profesional y académica ofrecida, así como el personal de administración y servicios, quien ha hecho más cómodo mi trabajo.

Tanto en mi etapa universitaria en la Facultad de Informática de la UPV/EHU como en mi etapa de doctorando en el CEIT he recibido muchos y buenos consejos por parte de Alex Garcia Alonso. Gracias por todos ellos, especialmente aquellos tan directos y simples como ¡publicar, publicar, publicar!

Habré tomado decisiones erróneas durante la tesis, pero tengo claro cuál ha sido uno de mis mayores aciertos: la elección de Diego Borro como director de tesis. Gracias por el esfuerzo y tiempo que me has dedicado, por transmitirme ese punto de confianza cuando los resultados no acompañaban. Ha sido muy enriquecedor el poder haber trabajado contigo, tanto profesionalmente como personalmente. ¡Gracias!

También me gustaría dar las gracias a mis *socios de AR*, sois insustituibles. Gracias Jairo por toda tu ayuda, especialmente en los duros inicios (¡forza Altza connection!); y gracias Ibai por motivarme y hacerme reír durante la última etapa.

Tampoco podría haber escogido mejores compañeros de trabajo. Gracias a todos vosotros por haber conseguido que disfrute el día a día: Nerea A., Ainara B., Ainara P., Iñigo, Tomasz, Julian, Mike, Álvaro S., Ane, Javi M., Pablo, Ainitze, Jorge, Iñaki, Manolo, Imanol P., Jorge Juan, Emilio, Yaiza (por esas miradas a través de la mampara), Alvaro B., Fran (por tu naturalidad), Sergio, Alberto, Iker (por tus interesantes comentarios), Aiert, Ibon y Aitor C. (por ofrecerme modelos 3D de calidad), Gaizka (por esas rutas moteras), Maite (por tus observaciones femeninas), Carlos (por tu gran apoyo técnico), Josune (por tus buenos consejos), Alba (por contagiarme tu buen humor), Goretti (por esos ánimos y gestos desinteresados, sabes que tengo amigos), Imanol H. y Pedro (por amenizar las sobremesas, el ajedrez es un deporte), Aitor A. (por esas representaciones gráficas, no fake), Borja (por esos debates $a = b * c/f$, me encantan), Luis (Kun) A., Ilaria (por mejorar mi italiano), Alex V. (por tus sinceras reflexiones, visualizas como nadie), Gorka (por crear a Meteo), Javi B., Nere E., Luis U., Ignacio M., Oskar, Aitor R., Maider, Denis, Dimas.

No me olvido de mis valiosos amigos, quienes han sido capaces de alegrar mis ratos libres y soportar mis numerosas ausencias: Aitor, Dani, David (Boni), Diego P., Igone, Iker G., Javi F., Javi P., Jony, Odin, y Sergio L., por citar algunos. ¡Gracias!

Y por supuesto, tengo mucho que agradecer a mi familia, no solo por su apoyo durante la tesis, sino a lo largo de toda mi vida. Todo lo que he conseguido ha sido gracias a vosotros. Gracias a mis padres Eduardo y Goretti, a mi hermana Inge, y a mi "hermano" Ander.

Si de algo estoy orgulloso, es de poder haber compartido esta experiencia con todos vosotros (más alguno que seguramente se me haya olvidado). ¡Gracias!

Abstract

The main challenge of an augmented reality system is to obtain perfect alignment between real and virtual objects in order to create the illusion that both worlds coexist. To that end, the position and orientation of the observer has to be determined in order to configure a virtual camera that displays the virtual objects in their corresponding position. This problem is known as *tracking*, and although there are many alternatives to address it by using different sensors, tracking based on optical sensors is the most popular solution. However, optical tracking is not a solved problem.

This thesis presents a study of the existing optical tracking methods and provides some improvements for some of them, particularly for those that are real time. More precisely, *monocular optical marker tracking* and *model-based monocular optical markerless tracking* are discussed in detail. The proposed improvements are focused on industrial environments, which is a difficult challenge due to the lack of texture in these scenes.

Monocular optical marker tracking methods do not support occlusions, so this thesis proposes two alternatives: (1) a new tracking method based on temporal coherence, and (2) a new marker design. Both solutions are robust against occlusions and do not require more environment adaptation. Similarly, the response of model-based monocular optical markerless tracking methods is jeopardized in untextured scenes, so this thesis proposes a 3D object recognition method that uses geometric properties instead of texture to initialize the tracking, as well as a markerless tracking method that uses multiple visual cues to update the tracking.

Additionally, the details of the augmented reality system that has been developed to help in disassembly operations are given throughout the thesis. This serves as a tool to validate the proposed methods and it also shows their real world applicability.

Resumen

El principal desafío de un sistema de realidad aumentada consiste en alinear correctamente los objetos reales y virtuales, creando la ilusión de que ambos mundos coexisten. Para ello es necesario calcular la posición y orientación del observador que permita configurar la cámara virtual que renderiza los objetos virtuales en su posición exacta. Este problema es conocido como *tracking*, y aunque existen varias alternativas para su resolución, usando diferentes sensores, el tracking óptico es la solución más popular. No obstante, el tracking óptico es un problema no resuelto.

Esta tesis presenta un estudio de los métodos de tracking óptico existentes y propone mejoras en algunos de ellos, especialmente en aquellos que son tiempo real. Más concretamente, se analizan en profundidad los métodos de *tracking óptico monocular con marcadores* y los métodos de *tracking óptico monocular basado en el modelo* (sin marcadores). Las mejoras se han propuesto teniendo en cuenta las características de los entornos industriales, que carecen de textura.

El tracking óptico monocular con marcadores no soporta oclusiones, por lo que este trabajo propone dos soluciones: (1) un nuevo método de tracking basado en coherencia temporal, y (2) un nuevo diseño de marcador. Ambas soluciones consiguen mayor robustez ante oclusiones sin necesidad de adecuar más el entorno. Asimismo, el tracking óptico monocular basado en el modelo no funciona correctamente en escenas poco texturadas, por lo que esta tesis propone un método de reconocimiento 3D que inicializa el tracking usando características geométricas en vez de textura, y un método de tracking que combina múltiples características visuales para actualizarse.

Este documento también detalla el sistema de realidad aumentada desarrollado para la ayuda en operaciones de desmontaje. Este sirve como herramienta de validación, además de ser un ejemplo de aplicabilidad real.

Contents

List of Figures

List of Tables

List of Algorithms

Part I

Introduction

Chapter 1
Introduction

The key to ultimate success
is the determination to progress day by day
Edmar Mednis

Some ideas from this chapter have been published in:

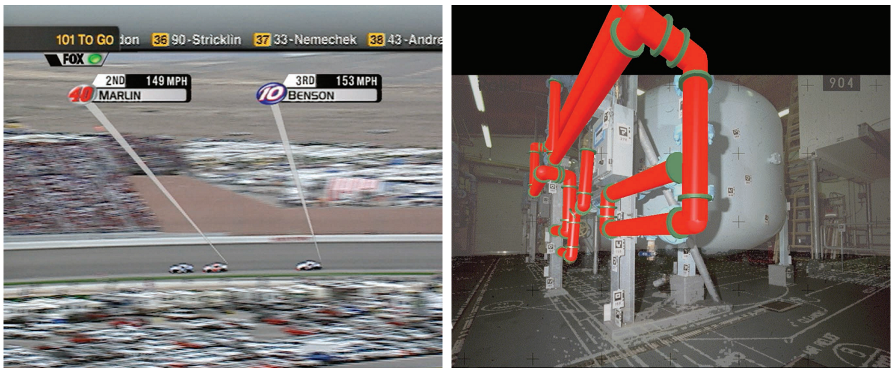
Basogain, X., Olabe, M., Etxebarri, A., Izkara, J. L., Garrido, R., and Álvarez, H. "Towards the augmented reality in wearable personal assistants". In II Jornadas sobre Realidad Virtual y Entornos Virtuales (JOREVIR'08). Albacete, Spain. June, 2008.

Puerto, M., Gil, J., Álvarez, H., and Sánchez, E. "Influence of user grasping position on haptic rendering". IEEE/ASME Transactions on Mechatronics, N. 99, pp. 1–9. 2011.

1.1 Augmented Reality

Augmented Reality (AR) is a technology that enriches the way in which users experience the real world with additional virtual information. This augmented data could refer to simple 2D annotations or more sophisticated 3D objects (Figure 1.1).

In contrast to Virtual Reality (VR), where the user is completely immersed in a synthetic world, AR consist on adding virtual objects to the real world. The main goal of AR is to create the sensation that virtual objects belong to the real world, offering perfect alignment between virtual



(a) 2D augmentation. (b) 3D augmentation.

Figure 1.1: Examples of augmented reality (Azuma et al., 2001).

and real objects. The virtual-reality continuum proposed by Milgram (Milgram et al., 1995) emphasizes the differences between AR and VR (Figure 1.2).

A common AR system requires a display device to render an image in which the virtual objects are overlaid with their real counterparts. This device can be as simple as a computer monitor or television, or as a complex as a head mounted display (HMD). (Azuma, 1997) and (Cawood and Fiala, 2008) classify display devices into two categories: *Video See Through - Magic Mirror* and *Optical See Through - Magic Lens*. The first group situates the user in front of a projection screen and does not allow any direct view of the real world (Figure 1.3), while the second alternative uses transparent displays so that the user can look directly through them to see the real world (Figure 1.4).

The main challenge of an augmented reality system is to obtain a robust and accurate *registration*. The registration problem is based on finding a perfect alignment between real and virtual objects since it is essential to create the illusion that virtual and real worlds coexist. This requirement can be clearly seen in the example of 2D annotations presented above (Figure 1.1(a)), where misalignment between the text and car positions can induce an error of interpretation. To solve this problem, the position and orientation of the observer has to be determined. Using this information a virtual camera can be configured, indicating the exact location where the virtual objects should be drawn in the image. Another challenge for AR

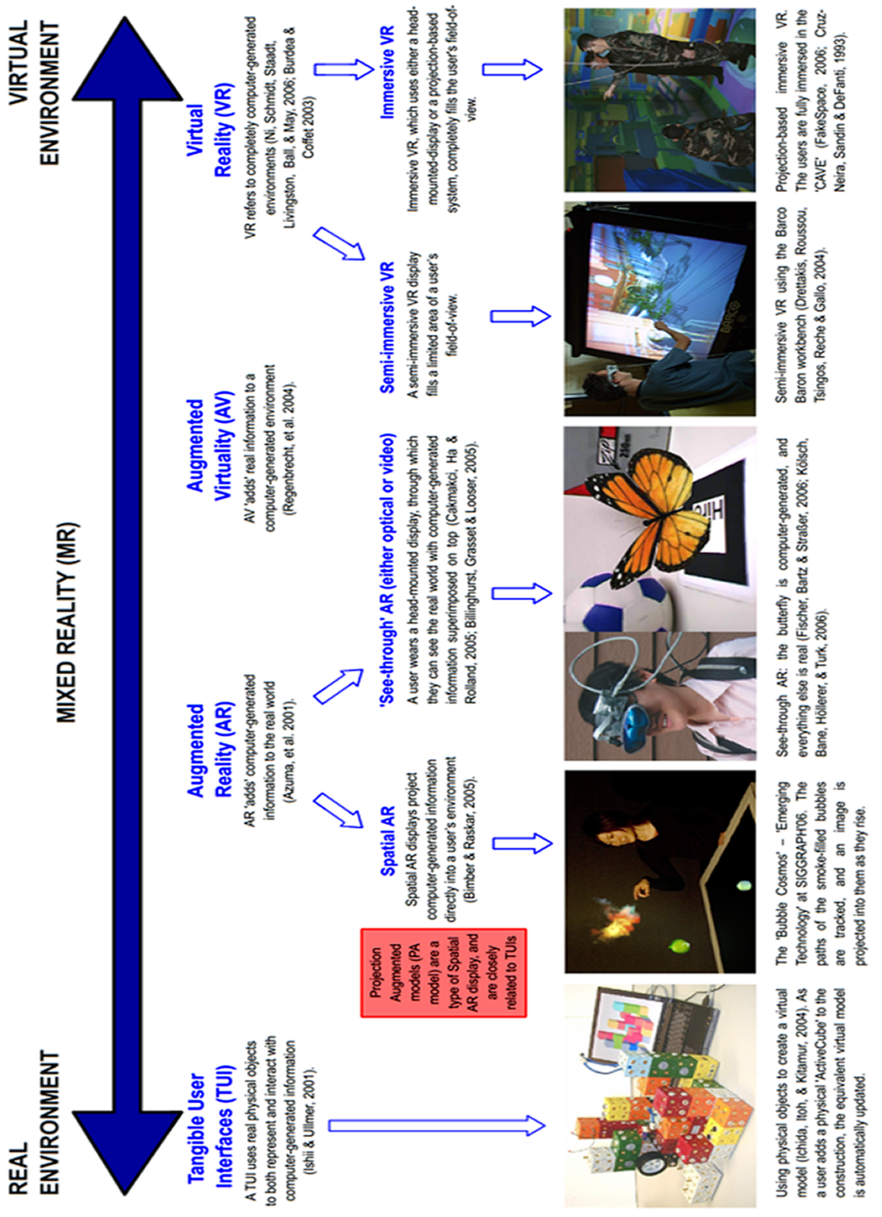
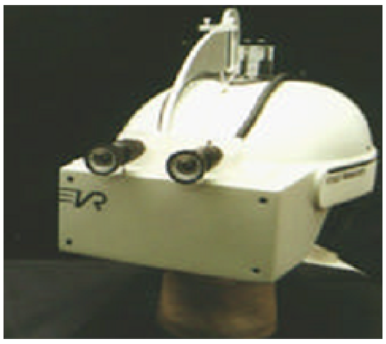
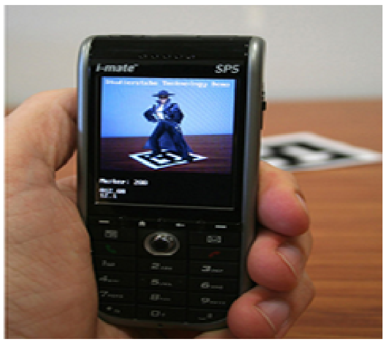

VIRTUAL ENVIRONMENT
MIXED REALITY (MR)
REAL ENVIRONMENT
Tangible User Interfaces (TUI)
A TUI uses real physical objects to both represent and interact with computer-generated information (Ishii & Ullmer, 2001).
Augmented Reality (AR)
AR 'adds' computer-generated information to the real world (Azuma, et al. 2001).
Augmented Virtuality (AV)
AV 'adds' real information to a computer-generated environment (Regenbrecht, et al. 2004).
Virtual Reality (VR)
VR refers to completely computer-generated environments (Ni, Schmidt, Staadt, Livingston, Ball, & May, 2006; Burdea & Coffet 2003)
Projection Augmented models (PA model) are a type of Spatial AR display, and are closely related to TUIs
Spatial AR
Spatial AR displays project computer-generated information directly into a user's environment (Bimber & Raskar, 2005).
'See-through' AR (either optical or video)
A user wears a head-mounted display, through which they can see the real world with computer-generated information superimposed on top (Cakmakci, Ha & Rolland, 2005; Billinghurst, Grasset & Looser, 2005).
Semi-immersive VR
A semi-immersive VR display fills a limited area of a user's field-of-view.
Immersive VR
Immersive VR, which uses either a head-mounted-display or a projection-based system, completely fills the user's field-of-view.
Using physical objects to create a virtual model (Ichida, Itoh, & Kitamur, 2004). As a user adds a physical 'ActiveCube' to the construction, the equivalent virtual model is automatically updated.
The 'Bubble Cosmos' – 'Emerging Technology' at SIGGRAPH'06. The paths of the smoke-filled bubbles are tracked, and an image is projected into them as they rise.
See-through AR: the butterfly is computer-generated, and everything else is real (Fischer, Bartz & Straßer, 2006; Kölsch, Bane, Höllerer, & Turk, 2006).
Semi-immersive VR using the Barco Baron workbench (Drettakis, Roussou, Tsingos, Reche & Gallo, 2004).
Projection-based immersive VR. The users are fully immersed in the 'CAVE' (FakeSpace, 2006; Cruz-Neira, Sandin & DeFanti, 1993).

Figure 1.2: Detailed Milgram Reality-Virtuality continuum (image licensed under the Creative Commons Attribution-ShareAlike 3.0 License).

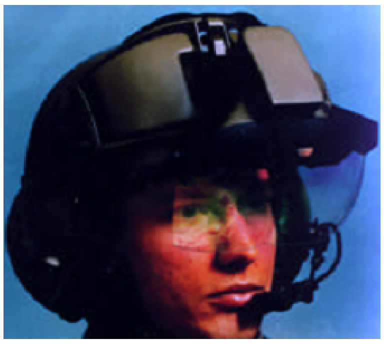


(a) HMD (Azuma, 1997).

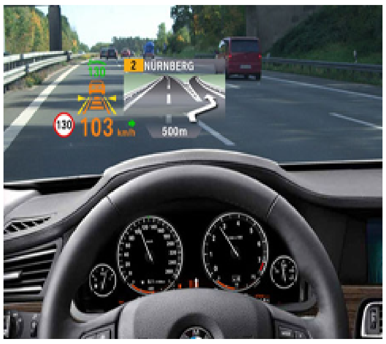


(b) Smartphone (Wagner and Schmalstieg, 2007).

Figure 1.3: *Video See Through - Magic Mirror* examples.



(a) HMD (Azuma, 1997).



(b) BMW head up display (©BMW).

Figure 1.4: *Optical See Through - Magic Lens* examples.

systems is the problem of finding the parameters that define the camera, which is referred to as *tracking*. It requires the extraction of the 6 degrees of freedom (DOF) that represent the user's viewpoint, 3-DOF for the orientation and 3-DOF for the translation. There are many alternatives to address this problem, which differ in the type of sensors they use (Rolland et al., 2001): inertial sensors combine accelerometers and gyroscopes to estimate the translation and rotations respectively; ultrasound sensors rely on the delay times of ultrasonic pulses to infer position and orientation; GPS receivers use the signals emitted by a set of satellites to triangulate its position; magnetic sensors measure the magnetic fields to deduce the viewpoint parameters; and optical sensors process the image of the scene captured by a camera to obtain its corresponding 6-DOF.

The first work associated with augmented reality dates from 1968 (Sutherland, 1968), when Ivan Sutherland built a prototype with a HMD to render 2D perspective images that created the illusion of visualizing 3D dimensional data (Figure 1.5). This was the starting point for many of the AR applications that exist today, which have extended along many different fields, such as medicine (Blum et al., 2009), cultural heritage (Izkara et al., 2008) or education (Juan et al., 2010), to name a few.

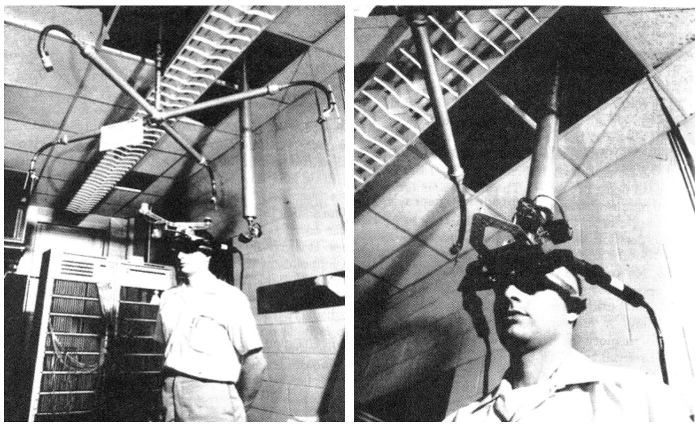


Figure 1.5: Mechanical (left) and ultrasonic (right) head tracking sensors used by Sutherland (Sutherland, 1968).

Recently, due to the meteoric evolution of the market for mobile devices, augmented reality is increasingly present in everyday life. For example, (Basogain et al., 2008) describes an AR platform that tries to develop a Wearable Personal Assistant, using mobile devices as tools that provide user-support for daily activities. Moreover, nowadays these devices have several integrated sensors, which provide all necessary hardware for tracking. Layar and Wikitude are applications that demonstrate this ability; they use a hybrid tracking system that combines a GPS and compass-integrated sensors to compute the position and orientation of the device respectively. This, together with a geolocated online database, makes it possible to overlay virtual information related to the places that are being recorded by the camera sensor (Figure 1.6(a)). Similarly, the game industry has adapted some of its products to satisfy this emerging demand, as can be seen in the release of augmented reality games like Invizimals, developed by

Novarama for the PSP platform (Figure 1.6(b)). In this case, only optical tracking is used to recognize some special patterns placed in the scene (markers), whose geometric properties are known and allow the recovery of the position and orientation of the camera. This optical tracking method was made popular in 1999, when Hirokazu Kato developed the ARToolkit (Kato and Billinghurst, 1999), a widely known marker-based monocular optical tracking.

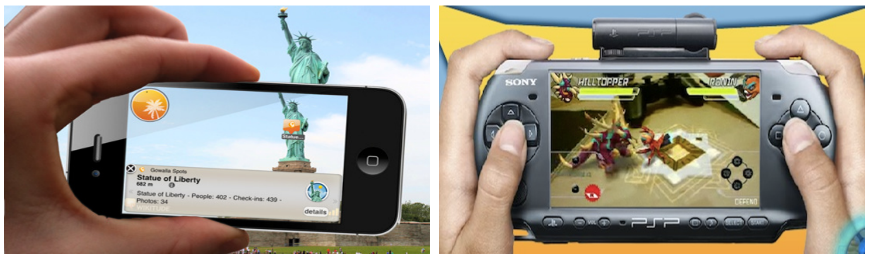
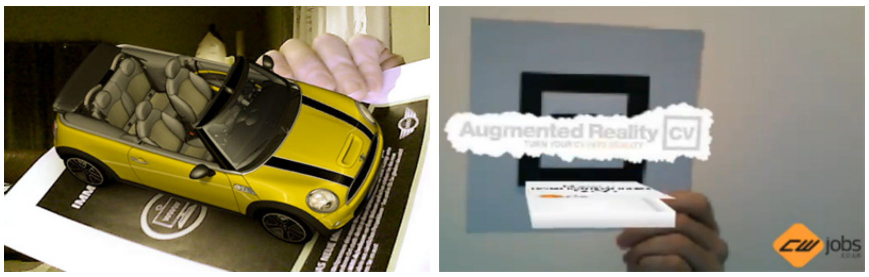

(a) Wikitude screenshot. (b) Invizimals screenshot (©Sony Corp.).

Figure 1.6: Examples of AR applications for mobile devices.

The current success of augmented reality in the marketing and publicity fields also highlights its growth in popularity. Many companies have started using AR to make different and novel advertisements. This is the case for various automotive companies, which overlay virtual information about their products when some predefined markers are detected in the image captured by the camera sensor (Figure 1.7(a)). As mentioned above, the camera parameters are extracted using marker-based optical tracking. Usually, these markers are simple pieces of paper, and therefore they have low manufacturing costs and are easily integrated with everyday items like magazines or newspapers. Continuing with this idea, the company CWjobs proposes a CV based on AR (Figure 1.7(b)), displaying virtual information related to the skills and experience of applicants when these markers are detected.

Augmented reality solutions can also be found for industrial environments. For example, (Hakkarainen et al., 2008) proposes the use of a marker-based optical tracking to assist with assembly tasks (Figure 1.8(a)) by visualizing what the next part and placement should be. Nonetheless, not all optical tracking solutions are based on markers. Some applications extract the camera position and orientation via computer vision techniques



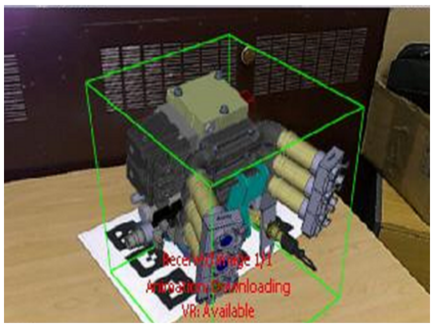
(a) Mini advertisement (©Mini). (b) CV based on AR (©CWjobs).

Figure 1.7: Examples of AR for marketing and publicity.

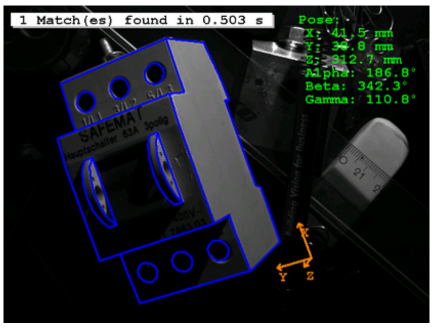
that process the image captured by the camera sensor without any special hardware or external markers. Usually, these techniques try to make correspondences between some visual cue templates and those detected in the camera image, obtaining the camera parameters that best fits the transformation of correspondences. Multiple visual cues have been used to validate the quality of the correspondences. (Ulrich et al., 2009) recognize industrial objects by detecting the presence of shapes with similar contour geometries in the image, which can be used to perform several tasks, e.g., render virtual data associated with the object (Figure 1.8(b)), pick and place operations or quality control. Sometimes, however, due to the meaningful characteristic features that the target model has, the recognition of the object is based on appearance. On this basis, (De Crescenzio et al., 2011) recognize some components of an airplane by comparing the similarity of their texture patterns. An AR application that support technicians in aircraft maintenance and repair operations validates its functionality (Figure 1.8(c)). Additionally, (Puerto et al., 2011) perform a colour-based identification of the user's grasping position to analyse its influence on haptic rendering (Figure 1.8(d)).

1.2 Motivation

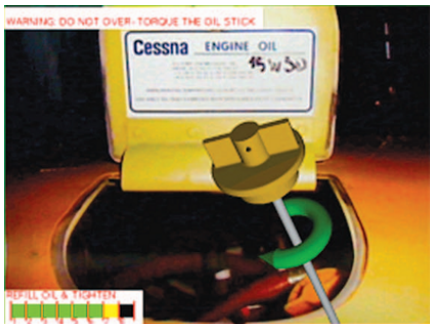
Tracking based on optical sensors is the most popular solution due to the fact that it requires minimal environmental adaptation and its low cost. It does not need to add bulky machines to the scene or force the user to use heavy devices, it only uses a camera to capture images of the scene,



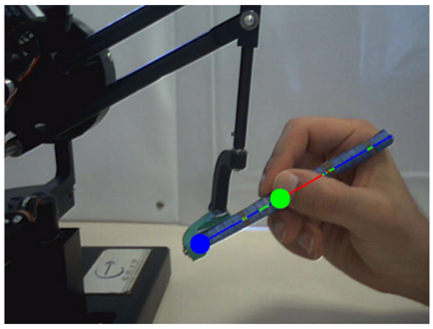
(a) Augmented assembly (Hakkarainen et al., 2008).



(b) Augmented data related to the recognized object (Ulrich et al., 2009).



(c) Augmented instruction for maintenance task (De Crescenzio et al., 2011).



(d) Augmented view of the user grasping position (Puerto et al., 2011).

Figure 1.8: Examples of AR for industrial environments.

a computer to process the images, and a screen to overlay the virtual information. Despite this, optical tracking is not a solved problem.

As stated before, the use of markers to solve the optical tracking problem is a common alternative in many fields . They provide accurate camera parameters, and they also require low computational resources. Nevertheless, besides having to add markers to the scene, the other shortcoming they suffer from is that they do not support occlusions. The tracking fails even when the marker is slightly occluded, making impossible for virtual information to be displayed. This failure produces an undesirable effect on users, who lose their sense of realism. The constant appearance and disappearance of virtual objects jeopardizes the effectiveness of AR technology. Similarly, in AR applications that are oriented to industrial environments it is very likely that the marker is occluded. Because the technicians use their hands to do the corresponding tasks, it is easy for the hands to occlude a part of the marker. The use of mobile devices also enhances the need for the treatment of occlusions. These devices are usually light, and as a consequence, they are moved easily, which increases the likelihood of occlusion. The probability of putting the marker partially outside the camera's field of view increases with the rapid sudden movements that usually occur when manipulating handheld devices. Because of that, it would be very interesting to offer a solution that obtains more robustness against occlusions and does not require any extra scene modification to achieve it.

As noted above, augmented reality has an important role in industrial environments. In addition, from the point of view of usability, AR offers multiple benefits in assembly tasks (Maad, 2010). This statement alleges that virtual information is interpreted more easily than documentation based on paper. Thus, AR guidance reduces errors in assembly sequences, as it is clear how to proceed and where the next component is supposed to be placed. Furthermore, AR animation favours the identification of relations between the different components due to the enriched visual perception of their parameters, such as texture, material or colour. As a consequence of the improved comprehension of the assembly task, performance improvements are also obtained because technicians are able to perform the same task in less time. This improvement in efficiency can also be explained by the stimulation of motivation, since the enjoyment of the interactive experience in AR animation might increase the motivation and interest of technicians. It is noticeable that all these advantages are analogous for disassembly tasks.

Due to all the advantages mentioned above, many researchers have addressed the problem of building an AR system for guidance in assembly/disassembly tasks. In most cases marker-based optical tracking is used to recover the 6 DOF of the camera, but the environment adaptation that the marker tracking systems require is not always possible. Thus, it would be beneficial to have an AR guidance system that uses optical tracking that dispenses with markers. Due to this reasoning, solutions that use multiple cameras (stereoscopic vision) should be discarded, as the main objective is to minimize environment adaptation. In fact, monocular optical tracking based on computer vision techniques already exist for industrial environments, as mentioned earlier. However, existing approaches have two main drawbacks. First, many of these techniques are designed for environments rich in texture, where the presence of different texture patterns favours the distinctiveness of appearance, and consequently, tracking is simplified. On the other hand, these methods need an extensive user intervention to obtain some of the visual cue templates required for tracking, as well as the specification of the assembly/disassembly sequence. Considering all these problems, it would be useful to build an AR system for guidance that uses monocular optical 3D tracking that is not based on markers, that minimizes user intervention by building all the necessary data automatically, and that is a valid solution for untextured environments. It is noteworthy that this last requirement is critical, as many industrial objects have a homogeneous outer surface that does not provide much information, and consequently is a difficult challenge.

The realism of AR increases when correct illumination is estimated or when highly detailed models are rendered (Quintana et al., 2010). Similarly, the effectiveness of AR increases when virtual elements are added in real time. As stated by (Russ, 1999), the meaning of real time varies with the application's characteristics; i.e., for some situations, such as video processing, very short exposures and high rates are needed, while for others, such as remote sensing, a single frame is taken over long periods. The real time definition that is used in this dissertation is that accepted for video acquisition, namely, 1/30 second per full frame. This enables the computer to refresh the augmented data very quickly, constantly obtaining a valid alignment between virtual and real elements, even when the camera is moving. Thus, it would be important to execute AR methods near the limits of real time.

1.3 Contributions

The goal of this thesis is the improvement of the existing monocular optical tracking solutions for augmented reality, with a focus on industrial environments. Using standard hardware components, such as a low cost webcam and a common computer equipped with a simple monitor, the main challenge that this dissertation addresses is the calculation in real time of the 6 DOF that define the position and orientation of the camera that is used to overlay the virtual information. In order to achieve this objective and provide a valid solution for most contexts, a new marker-based method as well as novel tracking alternatives based on computer vision techniques are proposed. Furthermore, the characteristics of industrial environments have been considered when designing these methods, and as a result, they can handle untextured scenes. Apart from that, an AR system for guidance in disassembly tasks has been developed to validate the quality and capacity of the proposed monocular optical tracking based on computer vision techniques. This AR system uses techniques that deduce automatically the disassembly sequence, offering a complete framework. The main contributions can be classified as follows:

- *Two marker-based tracking methods for the treatment of occlusions.*

  Both methods are able to update the camera parameters when the marker is partially occluded. The first one only updates 4 DOF of the camera, which is enough for some AR applications. As compensation for this shortcoming, the computational cost is low, making it ideal for mobile platforms. The second method consists of a new marker design that enables the extraction of the 6 DOF of the camera despite the marker occlusion. It offers new ways of developing novel interfaces. Additionally, it is based on texture patches that are customizable, which is a desirable property for marketing and publicity purposes.

- *A complete optical tracker that uses computer vision techniques to obtain the camera parameters.*

  First, a 3D object recognition method initializes the 6 DOF of the camera despite the difficult conditions of industrial objects (i.e., homogeneous outer surface). It uses the geometric constraints of the target model to calculate the camera parameters. These constraints are automatically extracted during a preprocessing stage. Once the camera position and orientation are initialized (a problem that is known as *first camera pose* in AR), a 3D optical tracker processes incoming camera images to update the 6 DOF of the camera. Using computer vision techniques and temporal coherence constraints, it measures the displacement of some image features between consecutive frames to apply the same motion to the camera parameters. The proposed method uses a well known computer vision technique called SIFT (Lowe, 2004), which obtains correspondences between different features of two images. As the literature demonstrates, it is one of the most robust techniques, but it consumes too many resources. Because of that, a simplified version of SIFT has been implemented, which offers a balance between robustness and computational cost.

- *An AR system for guidance in disassembly tasks.*

  The system only requires a single untextured 3D triangle mesh of each component that belongs to the model that is going to be disassembled. A path planning module (Aguinaga et al., 2008) is used to automatically compute the disassembly sequence, finding collision-free trajectories. Moreover, this module has been integrated with the optical tracker mentioned above, building a complete framework that is characterized by its ability to generate all the data automatically, minimizing user intervention, and offering assistance in disassembly operations.

1.4 Thesis Outline

This dissertation is organized in 7 chapters. Chapter 1 introduced the augmented reality technology, as well as the motivation and contributions of this work within that area of knowledge. Chapter 2 presents some preliminary concepts that are needed to understand subsequent chapters. A classification of different optical tracking approaches is offered among other ideas. Chapter 3 discusses monocular optical tracking based on markers, including proposed solutions to overcome occlusions. Chapter 4 deals with the 3D recognition of untextured industrial models, which serves as a initialization method for the camera parameters (first camera pose). Chapter 5 describes the monocular optical 3D tracking based on computer vision techniques that has been implemented to update the camera parameters. Chapter 6 presents the AR system that has been built to provide guidance in disassembly tasks. Finally, Chapter 7 enumerates the conclusions of this thesis and proposes some future research lines.

Some appendices also appear at the end of this document to explain some technical concepts in more detail. Appendix A focuses on the simplifications that have been applied to the original SIFT algorithm, and Appendix B provides the mathematical background to compute the camera pose from a 3D planar structure. Moreover, Appendix C shows a hierarchy of 2D transformations, and Appendix D includes a part of the documentation used in the usability experiments of Chapter 6.

Chapter 2

Background

Not infrequently... the theoretical is a synonym of the stereotyped. For the "theoretical" in chess is nothing more than that which can be found in the textbooks and to which players try to conform because they cannot think up anything better or equal, anything original
Mikhail Chigorin

The position and orientation of the camera must be determined in order to obtain perfect alignment between real and virtual objects and increase the effectiveness of augmented reality. Because of that, this chapter describes the mathematical tools that are necessary to understand the camera behaviour. The state of the art in camera tracking is also presented, addressing the problem of finding the parameters of the camera from an image or multiple images.

Some ideas introduced in this chapter can be found in:

Barandiarán, J., Álvarez, H., and Borro, D. "Edge-based markerless 3d tracking of rigid objects". In International Conference on Artificial Reality and Telexistence (ICAT'07), pp. 282–283. Esbjerg, Denmark. November, 2007.

Sánchez, J. R., Álvarez, H., and Borro, D. Gft: Gpu fast triangulation of 3d points (ISBN: 3-642-15909-5), volume 6374 of Lecture Notes in Computer Science, Computer Vision and Graphics, pp. 235–242. Springer-Verlag Berlin Heidelberg. 2010.

Sánchez, J. R., Álvarez, H., and Borro, D. "Towards real time 3d *tracking and reconstruction on a gpu using monte carlo simulations". In International Symposium on Mixed and Augmented Reality (ISMAR'10), pp. 185–192. Seoul, Korea. October, 2010.*

2.1 Camera Geometry

When the real world is shown through an image captured by a camera a 2D representation of the 3D world is perceived. The geometry of the camera (lens and sensor) must be known to apply this 3D-2D transformation. Furthermore, the accuracy of this projection process is critical, as it is the responsible of misleading our mind into believing that this is a window from which the 3D world is seen.

The camera lens is an optical device through which the light hits the internal sensor of the camera to form the image. There are several ways to explain this procedure, but the *pinhole model* is the simplest and widely accepted representation used in computer vision applications. It assumes the following statements: (1) that the wave propagation of light can be modelled as a straight rays; (2) that the lens can be substituted by an infinitesimally small aperture (a single point called *center of projection*); and (3) that the camera sensor can be represented by a planar surface called *image plane*, which is in front of the aperture hole. Thus, rays of light leaving the 3D object pass through the center of projection to form an inverted 2D image of the object in the image plane (see Figure 2.1).

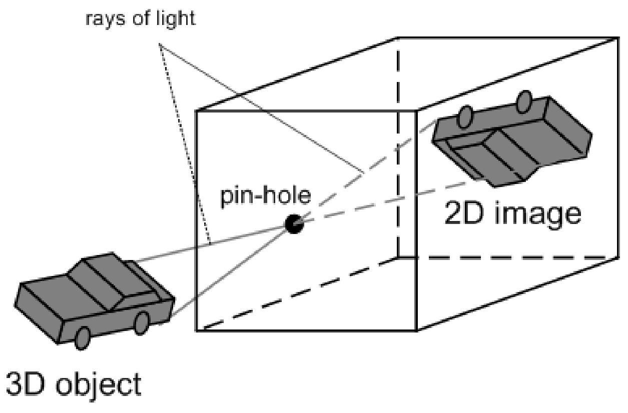


Figure 2.1: Pinhole camera model.

This way of relating a 3D world point with a 2D image point is known as *perspective projection*, and it explains the formation of images in a pinhole camera. As it is shown in Figure 2.2, the camera coordinate system is defined such that the XY plane is aligned with the image plane and the Z axis coincides with the *optical axis*, which is the axis passing through the center of projection, also called *optical center*. The intersection of the optical axis and the image plane is a point called *principal point*. In order to avoid the image appears inverted, an equivalent geometry configuration is used, where the optical center is moved behind the image plane. For this reason, the image plane is located at (0,0,f), where f is a non-zero distance, referred as *focal length*. Furthermore, for the moment, it is assumed that the camera and world coordinate systems are aligned to facilitate comprehension.

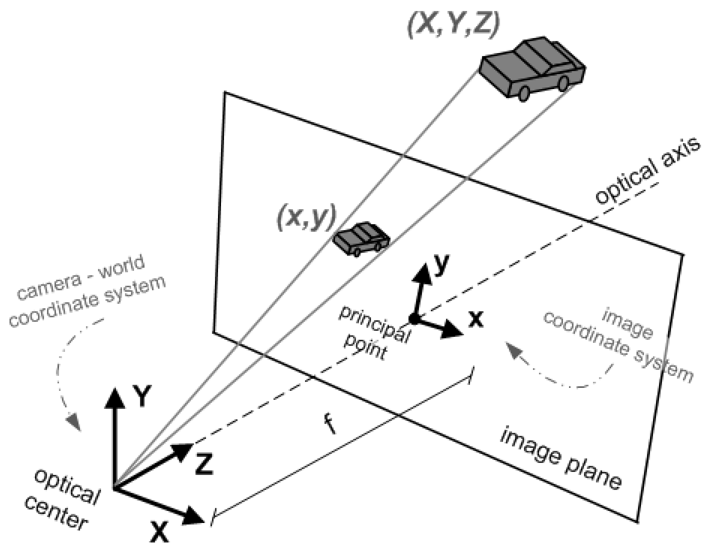


Figure 2.2: Perspective projection model simplified : camera and world coordinate systems are aligned.

According to the assumptions made above, the projection of a 3D point can be obtained by similar triangles:

$$x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z}, \tag{2.1}$$

where (X, Y, Z) are the 3D world coordinates of a point, and (x, y) are their image coordinates.

For a point with (X, Y, Z) cartesian coordinates, its homogeneous coordinates are given by (kX, kY, kZ, k), where k is an arbitrary constant that is non-zero. Using this notation, Equation 2.1 can be linearized, obtaining the following matrix form:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} \sim \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.2}$$

Nonetheless, this is an ideal camera, where several factors have not taken into account. In general, the image coordinate system is centred on the top left corner of the image (Figure 2.3), so the pixel coordinates of the principal point are not (0,0), but (p_x,p_y).

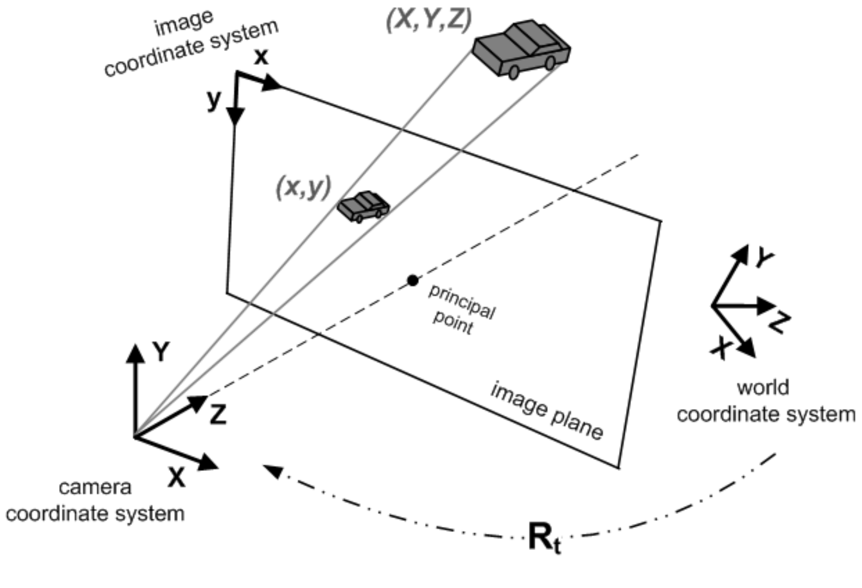


Figure 2.3: Perspective projection model.

Similarly, the camera sensor can result in a non-square pixels that are incorrectly positioned respect to the lens (Figure 2.4). The non-square size (s_x,s_y) produces two different focal lengths, one for each axis ($f_x = \frac{f}{s_x}, f_y = \frac{f}{s_y}$), while the error in the alignment between the sensor and lens is expressed as ($s = tan\alpha * \frac{f}{s_y}$), called the *skew* parameter.

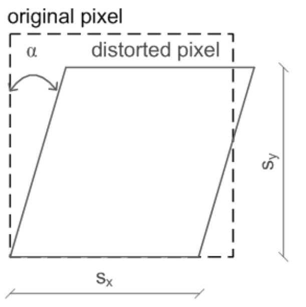


Figure 2.4: Distortion suffered by a pixel of the camera sensor.

Based on these changes the new projection coordinates are given by:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim KP_N \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.3}$$

where P_N is the projection matrix of the normalized camera, and K describes the characteristics of the camera.

There are other effects associated to the imperfection of lens that modify the final image coordinates. These effects are related to the tangential and radial distortions. The first one is an image defect caused by errors of lens centration, while the second one causes straight lines to curve, which results in image magnification. To correct this magnification, the image can be warped using the non-linear Brown's distortion model (Brown, 1966):

$$x_u = p_x + L(r)(x - p_x), \quad y_u = p_y + L(r)(y - p_y), \tag{2.4}$$

where (x,y) is an image point, (x_u,y_u) is the undistorted image point, $r^2 = (x - p_x)^2 + (y - p_y)^2$, and $L(r)$ is a distortion factor that can be approximated by a Taylor expansion $L(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + ...$, being $\{k_1, k_2, k_3, ...\}$ the coefficients for radial correction.

Equation 2.3 assumes that the camera and world coordinate systems are aligned, but this is not the most realistic example. Considering that the world coordinate system is fixed, the alignment is lost when the camera is moved around the scene[1]. Thus, an Euclidean transformation is required to align these coordinate systems and apply the pinhole model (Figure 2.3).

Applying this coordinate transfer to Equation 2.3, the 3D-2D projection pipeline results in:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim K P_N R_t \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \sim K P_N \begin{bmatrix} \mathbf{R} & \vec{t} \\ \vec{0}_3^\top & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

where R is a 3x3 rotation matrix and $\vec{t}$ is a 3x1 translation vector. Both represent the alignment between camera and world coordinate systems.

The matrix K is called the camera *intrinsic parameters*, while the matrix that describes the position and orientation of the camera (R_t) is called the camera *extrinsic parameters*. Furthermore, the P_N matrix is usually omitted for clarity in the notation, and consequently, this simplification is also used in the rest of the thesis. A more detailed explanation about the camera geometry can be found in (Faugeras et al., 2001; Hartley and Zisserman, 2004).

2.2 Camera Calibration

If the camera intrinsic parameters are known, then it is said that the camera is calibrated. Thus, camera calibration is the process of determining the values of the matrix K.

An object with a known geometric configuration is usually used to calibrate the camera. This object receives the name of *calibration pattern*, for which the location of several 3D control points is known. Given an image of the calibration pattern, the correspondence between these 3D points and their image projections is determined, which provides a set of equations that are used to find out the camera intrinsic parameters:

[1]The camera coordinate system could have been considered as fixed and the world coordinate system as mobile, which is referred as the duality of the observer.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.6}$$

where (X, Y, Z) are the 3D coordinates of the control points and (x, y) their corresponding image position. P is a 3x4 matrix that codifies both intrinsic and extrinsic parameters of the camera. Only six 3D-2D correspondences are needed to obtain P, which is decomposed in $P \sim K * R_t$ using the QR theorem (Flaquer et al., 2004). QR theorem states that a non-singular matrix P can be factored as the product of an upper triangular matrix K and an orthogonal matrix R_t, so the values of K are accessible.

As it is shown in Figure 2.5, calibration patterns are constructed so that the control points are easily identifiable in the image, facilitating the collection of correspondences. Examples of this calibration method can be found in (Tsai, 1987; Zhang, 2000).

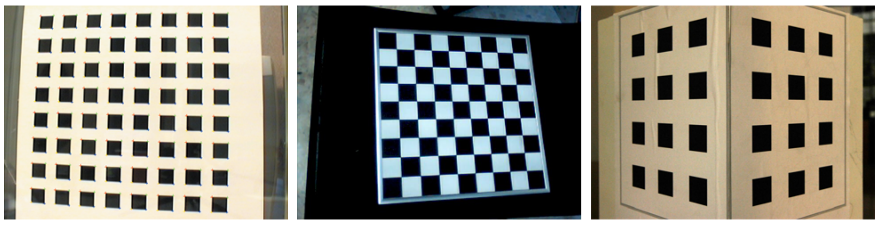

(a) (Zhang, 2000) pattern. (b) Chessboard. (c) Multi-planar pattern.

Figure 2.5: Examples of different calibration patterns.

Sometimes, however, is not possible to exploit the information provided by a calibration pattern. This is the case of images that have already been taken in an arbitrary scene and in which no calibration pattern appears. Examples of these images are those available in image libraries or extracted from a video sequence, which come without calibration data. Self calibration methods are used in order to deduce the internal camera parameters in these situations. For that purpose, they combine several images of the scene that are taken from different points of view (*multiview geometry*). Additionally, they assume some constraints like the existence of parallelism and orthogonality in the scene; or introduce some simplifications such as zero-skew ($s = 0$) or constant aspect ratio ($s_x = s_y$), which make easier the extraction of K. More details about these methods can be found in (Hemayed, 2003), where a survey on self calibration techniques is presented.

Camera intrinsic parameters are fixed provided that the focus is not changed, the zoom is not applied, and the image resolution is not altered. Therefore, the calibration process is only executed each time the camera configuration is modified.

2.3 Visual Cues

Visual cues are defined as those measures taken from an image (an array of colour pixels), which are used to perform multiple tasks, including optical tracking. Different types of visual cues can be detected in an image (Figure 2.6), such as image points that are very distinguishable from their neighborhood (*features*), areas with abrupt intensity changes (*edges*), or regions with homogeneous intensity levels (*blobs*). In addition, there are more complex structures derived from these visual cues, such as lines (straight edges), circles (blobs with circular shape) or junctions (a point where two straight edges meet).

Features are a very common alternative because of its simplicity, and they are used by many methods explained in this dissertation. For that reason, an overview of this type of image measurement is provided in the following paragraphs.

2.3.1 Features

Features, also called *corners*, *keypoints* or *interest points*, are a low level image measurements used for image analysis, particularly for tracking. They can be defined as salient points of the image, which are very distinguishable from their local neighbourhood. In its simple form, they are represented by pixel coordinates (sometimes with subpixel accuracy).

The process of determining the presence of features in an image is called *feature extraction* or *feature detection*. The quality of a feature detector is related to its ability to detect the same corner in different images (repeatability), being invariant to viewing conditions. It should be tolerant to as many transformations as possible, including camera rotations,

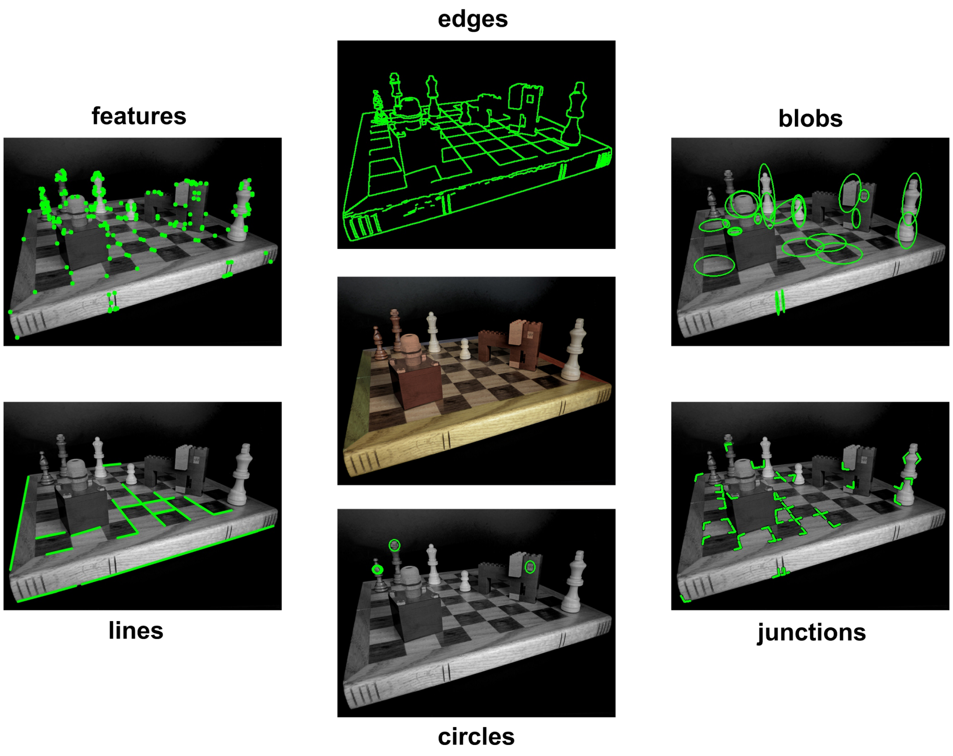


Figure 2.6: Different types of visual cues.

translations, scales, or illumination changes. Accuracy and performance are also an important issues when selecting an appropriate feature detector.

A more complete representation of a feature is given by the image region of its vicinity. A patch centered at the interest point is used to extract different image properties such as pixel intensities, colour, texture, edges etc, which are stored in a vector called *feature descriptor*. Each descriptor describes the appearance or shape of its corresponding feature, so a similarity measure of feature descriptors is used to get correspondences of a corner that is in multiple images. A good feature descriptor should satisfy the following properties:

- **Highly distinctive:** Two different features should have two different descriptors, i.e., the probability of a mismatch is low.
- **Robustness:** The descriptor of a feature should remain unchanged

despite the transformations applied to the image, i.e, the descriptor of a feature is preserved after rotations, translations, scales or illumination changes.

- **Good performance:** The computational cost of building and comparing feature descriptors should be low, being able to run in real time systems. The euclidean distance between two descriptors is usually enough to measure their similarity.

There are many feature detectors and descriptors algorithms. (Mikolajczyk and Schmid, 2004) describes several feature detectors, while (Mikolajczyk and Schmid, 2005) presents a comparison of different feature descriptors. Likewise, (Gauglitz et al., 2011) offers an evaluation of some feature detectors and descriptors oriented to optical tracking. A brief explanation of the popular FAST interest point detector (Rosten and Drummond, 2006) and the widely used SIFT feature descriptor (Lowe, 2004) is provided below, as they have been used in this thesis.

2.3.1.1 FAST

FAST (Features from Accelerated Segment Test) is a feature detector that classifies a pixel p as a corner if there are n pixel values in a discretized circle centered on p (denoted by $p \rightarrow x$) that are all brighter than the intensity of p (I_p) plus a threshold t, or darker than $I_p - t$ (Equation 2.7).

$$\begin{aligned} S_{bright} &= \{x \mid I_{p\rightarrow x} \geq (I_p + t)\}\,, \\ S_{dark} &= \{x \mid I_{p\rightarrow x} \leq (I_p - t)\}\,, \\ p \in Features &\Leftrightarrow (|S_{bright}| \geq n) \parallel (|S_{dark}| \geq n)\,. \end{aligned} \tag{2.7}$$

A Bresenham circle of radius 3 is used, so 16 pixels around the candidate p are considered (Figure 2.7); n is usually set to 9 (FAST-9) or 12 (FAST-12). A high-speed test is also used to discard non-corners with the minimum amount of checks: *at least three of the four pixels located at 1, 5, 9 and 13 positions (called compass directions) must be brighter or darker than I_p to continue testing the remaining locations.*

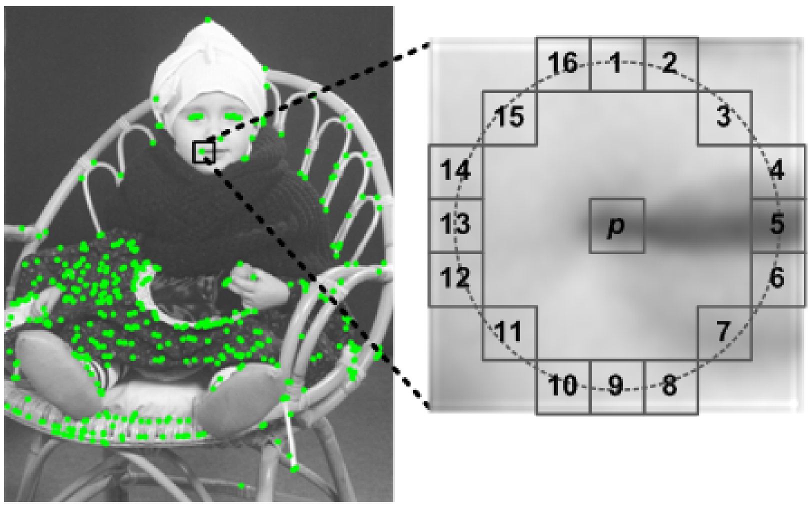


Figure 2.7: FAST feature detection (left) and enlarged image patch of a detected corner (right).

The response (quality) of each feature p is given by the intensity contrast between I_p and its surrounding pixels:

$$max\left(\sum_{x\in S_{bright}}(|I_{p\to x}-I_p|-t),\ \sum_{x\in S_{dark}}(|I_{p\to x}-I_p|-t)\right).$$

This score function is used to apply a non-maximal suppression, which gets stable features by removing corners that have and adjacent corner with higher response.

FAST obtains a good balance between repeatability and performance (it process high resolution images in few milliseconds), which is the reason of its popularity in real time applications. Additionally, the set of final features can be categorized as *bright* or *dark* at no extra cost, which is useful, since bright features do not need to be compared with dark features in post-processing steps such as matching.

2.3.1.2 SIFT

SIFT (Scale Invariant Feature Transform) includes both a feature detector and a feature descriptor. Corners are located using the *Difference of*

Gaussians (DOG) function, which is invariant to scale and orientation. An input image is convolved with different Gaussian kernels at multiple scales, and adjacent Gaussian images are subtracted to produce DOG. Corners are related to pixels that are locally maxima or minima, i.e., sample points whose value is larger or smaller than their 26 neighbors, 8 in the current scale and 18 in the upper and lower scales (Figure 2.8).

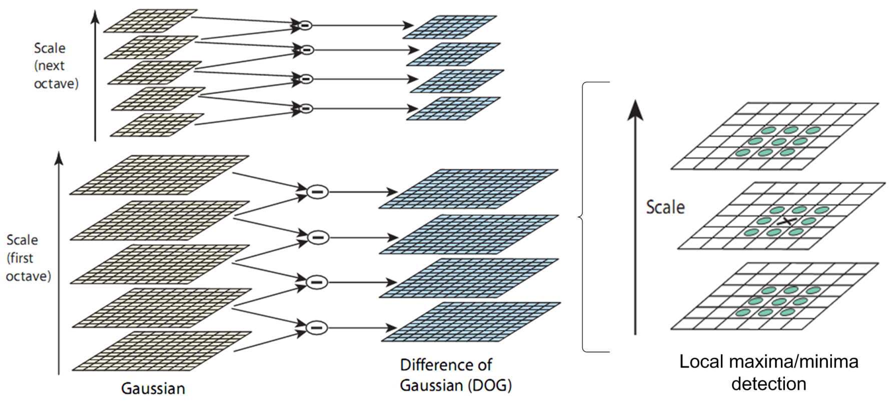


Figure 2.8: Corner location using DOG (Lowe, 2004).

The scale in which the corner is detected is stored, as it determines the size of the local image region used to build the feature descriptor (scale invariance)(Figure 2.9). Moreover, as this technique offers *regions* rather than points, it is considered as a *blob detector*.

Each located corner is represented by a descriptor (SIFT descriptor), which is extracted relative to the dominant orientation of the corner to achieve invariance to rotations. The image gradients of the surrounding pixels are calculated, whose orientations are used to build an orientation histogram. This histogram is discretized in 36 orientations (10 degrees per orientation), and each sample that is added to the histogram is weighted by its gradient magnitude and by a Gaussian-weight that gets more importance to central samples. The dominant orientation of a corner is the highest peak of the histogram. Nonetheless, if multiple peaks with similar magnitude are detected, then the corner is replicated with different dominant orientations.

The coordinates and gradient orientations of the pixels that belong to the local image patch of the feature are rotated according to the corresponding dominant orientation (rotation invariance). Furthermore,

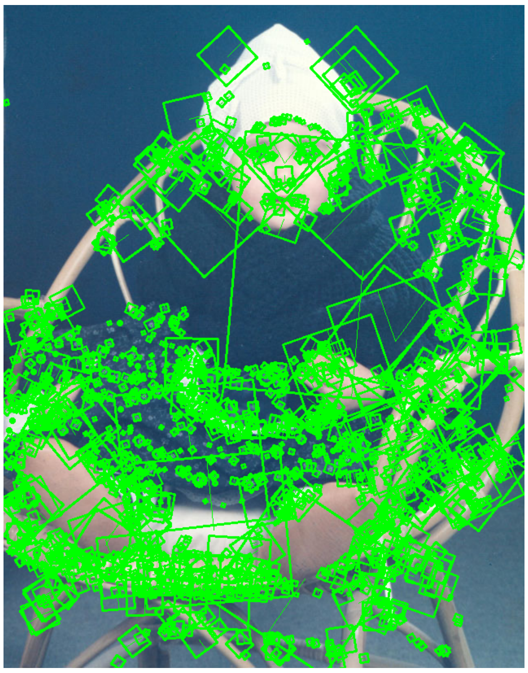


Figure 2.9: Feature detection using DOG. Each rectangle represents the scale and orientation of a feature.

this patch is subdivided in $n * n$ subregions, and each subregion is characterized by an orientation histogram of b bins (Figure 2.10), where n and b are application dependent parameters defined by the user. Finally, all these histograms are concatenated into a single histogram, which is normalized to unit length to reduce the effects of illumination. $n = 4$ and $b = 8$ is a typical parameterisation, which gives a vector of $4 * 4 * 8 = 128$ elements for each feature.

SIFT is a very robust technique for object recognition. Features extracted in some reference images are matched to those features detected in the current image by using similarity between SIFT descriptors. However, SIFT has high computational cost for real time tasks, as it takes hundred of milliseconds to process an image of 640x480 resolution. Speeded-Up Robust Features (SURF) method (Bay et al., 2008) is similar to SIFT, which uses efficient tools such as integral images to minimize computational cost, but still not real time. Due to this limitation, a simplified version of SIFT has been implemented for this dissertation, called *simplified-SIFT*

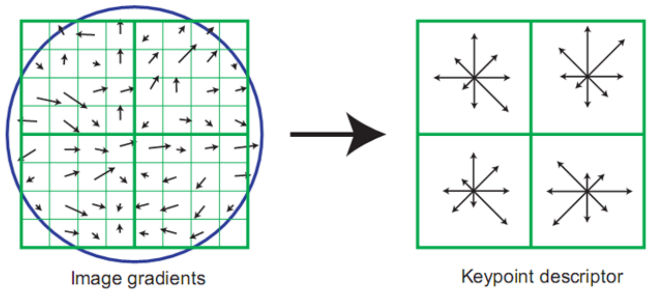


Figure 2.10: SIFT descriptor (Lowe, 2004).

(see Appendix A). In this implementation the expensive DOG operator is replaced by the FAST detector, and some parallel techniques are applied in order to reduce the computational time.

2.3.1.3 Optical Flow

Apart from the location and description, the motion is another property that can be useful when dealing with features. Thus, *optical flow* is a technique that estimates the motion of a feature between two consecutive frames, taken at times t and $t + \Delta t$. For a feature with intensity $I(x, y, t)$ that is moved by Δx, Δy and Δt between two images, optical flow satisfies the following constraint:

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x, y, t)+\frac{\delta I}{\delta x}\Delta x+\frac{\delta I}{\delta y}\Delta y+\frac{\delta I}{\delta t}\Delta t+H.O.T., \quad (2.8)$$

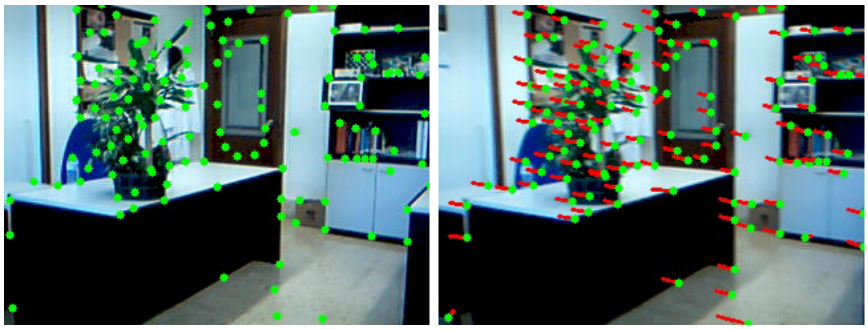
where $\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y} and \frac{\delta I}{\delta t}$ are partial derivatives of I, and $H.O.T.$ are the higher order terms of Taylor series. Assuming a small movement, $H.O.T.$ can be disregarded, resulting in $\frac{\delta I}{\delta x}\Delta x+\frac{\delta I}{\delta y}\Delta y+\frac{\delta I}{\delta t}\Delta t = 0$, which is a single equation with two unknowns (Δx and Δy, corresponding to the feature motion). This problem is known as *aperture problem.*

Lucas-Kanade method (Lucas and Kanade, 1981) is an optical flow algorithm that assumes that the flow remains constant in the local neighbourhood of each feature. Therefore, the aperture problem is solved by adding the constraints of pixels that are in the local neighbourhood. It provides the following least squares solution:

$$\epsilon(\Delta x, \Delta y) = \sum_{u,v \in W} \left[I(u,v) - J(u+\Delta x, v+\Delta y)\right]^2, \tag{2.9}$$

where I and J are consecutive images, W is the size that defines the local neighbourhood, and ϵ is the residual function to be minimized. Notice that Equation 2.9 corresponds to the sum of squared differences (SSD) of the intensities of two image patches.

Large values of W allows large motions, but increases the instability of the estimated flow. Because of that, a pyramidal implementation of Lukas-Kanade algorithm is proposed in (Bouguet, 2000), which offers a trade off between accuracy and robustness. First, optical flow is estimated at the lowest resolution image, and then, this result is propagated to the next resolution image as an initial guess. This procedure is repeated until the highest resolution is reached, which coincides with the original image. Note that W is fixed for all resolutions, so large motions computed at low resolutions are refined by the accurate estimations of high resolutions. An example of this method can be shown in Figure 2.11, which is configured with two pyramidal reductions and $W = 10$ pixels.



(a) Detected features. (b) Flow of features.

Figure 2.11: Optical flow for two images of a sequence.

Optical flow estimations cannot be used indefinitely without any correction, as the errors in the estimations are integrated over the time, i.e., optical flow prone to drift. Additionally, it works better against smooth movements due to the assumption of small movement between images. Local regions with constant image intensity also jeopardizes the quality of the optical flow, as it produces degenerate solutions. Indeed, that is why features are selected, as they obtain optimal optical flow estimations. For more information about the usage of features with the Lucas-Kanade algorithm, the interest reader can refer to (Tomasi and Kanade, 1991), where the Kanade-Lucas-Tomasi (KLT) feature tracker is described.

2.4 Camera Tracking

Camera tracking is the process that extracts the position and orientation (jointly called *pose*) of the camera relative to a global coordinate system (usually cited as world coordinate system). As introduced earlier, several sensors can be combined to address this task, but in this dissertation only optical tracking (also called *visual tracking*) is studied. Thus, given an image, the camera tracking finds out the camera extrinsic parameters (R_t) that best align the camera and world coordinate systems.

The source of information for optical tracking is(are) the image(s) of the scene captured by the camera(s), which emphasizes the need for image processing. Moreover, computer vision is a field of study and research that focuses on interpreting the world that is seen in one or more images. Because of that, computer vision is used to calculate the camera pose, recognizing some visual cues in the image(s) captured by the camera(s).

Given an input image, the image positions of some visual cues are detected and matched with their corresponding 3D locations to extract the camera pose. It can be expressed mathematically assuming the pinhole camera model, solving $\vec{m}_i = P\vec{M}_i$ for a set of $\vec{m}_i \leftrightarrow \vec{M}_i$ correspondences, where $\vec{m}_i$ is the 2D image position of the visual cue i, $\vec{M}_i$ are their corresponding 3D coordinates, and $P = KR_t$. The Direct Linear Transformation (DLT) algorithm (Hartley and Zisserman, 2004) solves that linear equation in case that the camera is not calibrated. Besides this technique, the Perspective-to-Point (PnP) methods (Lepetit et al., 2009) are also used when K is known. Nevertheless, all these linear methods lack precision when the measurements $\vec{m}_i$ are inexact (generally termed *noise*), so it is preferable to use a non-linear minimization of the reprojection error, i.e., the squared distance between $\vec{m}_i$ and the projection of $\vec{M}_i$:

$$\underset{R_t}{\operatorname{argmin}} \sum_i \left\| \vec{m}_i - KR_t\vec{M}_i \right\|. \tag{2.10}$$

The non-linear least-squares Levenberg-Marquadt[2] (LM) algorithm (Madsen et al., 1999) is extensively used to solve Equation 2.10. It is an iterative process that converges into the local minima combining the Gauss Newton method with the gradient descent approach. Moreover, it requires a starting point, so the estimation computed by a linear method (DLT-PnP) is used to initialize the final solution.

There are many ways to get $\vec{m}_i \leftrightarrow \vec{M}_i$ correspondences, resulting in different optical tracking methods. A possible classification of the existing optical tracking methods is shown in Figure 2.12.

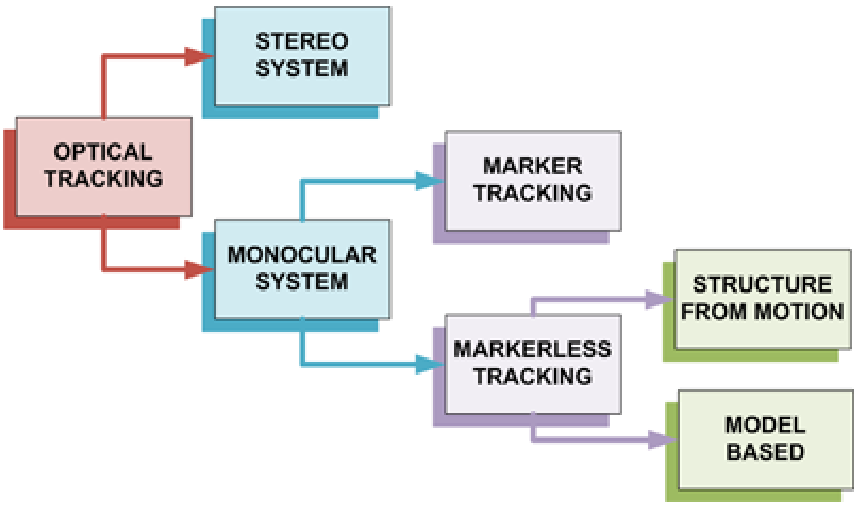


Figure 2.12: Classification of the optical tracking methods.

2.4.1 Stereo System

A stereo system processes several images of the scene at the same time, which are taken from different points of view by a set of cameras located strategically. In its simplest form, it is similar to the biological stereo vision of the human eyes, where two images of the same scene are captured from two different and known locations (left and right eye). Thereby, the 3D information of an object that appears in both images is extracted by triangulating its 2D image positions. This is how humans perceive the depth

[2]A widely used open source implementation of this method can be found in *http://www.ics.forth.gr/ lourakis/levmar/*.

of objects, and this also explains why is hard to estimate the distance at which an object is when the vision of one eye is lost. A depth image map can be obtained after triangulating all point correspondences, like the one presented in Figure 2.13.

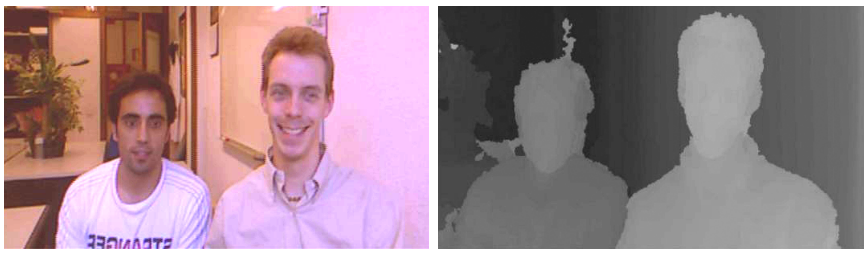


Figure 2.13: Depth image (right) for a given scene (left).

In order to perform the triangulation, points that are the projections of the same point in the 3D space must be identified in two or more views (*correspondence problem*). Generally, the image appearance of the local vicinity of each point is used to match points along a sequence of images. Additionally, these correspondences are constrained by the *epipolar geometry* (Hartley and Zisserman, 2004) (see Figure 2.14). Given a point in the left image ($\vec{x}_0$), the epipolar geometry states that its correspondence in the right image ($\vec{x}_1$) belongs to a straight line ($\vec{l}_1$). The line that belongs to the right image ($\vec{l}_1$) is the projection of the ray formed by the left optical center ($\vec{C}_0$) and the image point $\vec{x}_0$, so it is called *epipolar line* associated to $\vec{x}_0$. This is analogous for points of the right image and their left correspondences.

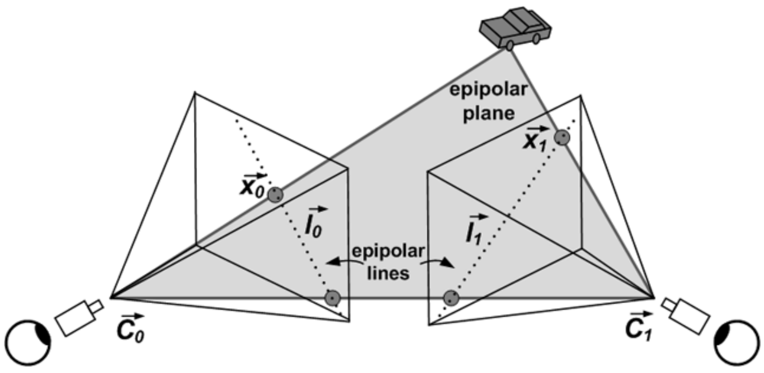


Figure 2.14: Two camera geometry.

The extraction of the 3D information, together with the known camera calibration parameters, facilitates the process of relating visual cues with their 3D values, i.e., helps to solve Equation 2.10. The disadvantage of a stereo system is that it requires bulky and expensive hardware. For further reading on stereo systems, please refer to (Brown et al., 2003).

2.4.2 Monocular System

Unlike a stereo system, a monocular system is composed of a single camera, which captures one image of the scene each time. Therefore, the knowledge about the scene should be extended to cope with this loss of information. There are many ways to represent this prior knowledge (Lepetit and Fua, 2005), which can be classified as *marker* and *markerless* tracking.

2.4.2.1 Marker Tracking

A marker tracking system adds a known and easily identifiable patterns (called *markers*, *fiducials* or *landmarks*) to the scene. Although there are different patterns, black squares printed on a white sheet (Cawood and Fiala, 2008; Kato and Billinghurst, 1999; Wagner and Schmalstieg, 2007; Zhang et al., 2002) are a widely used markers due to their good performance and low cost of manufacture. Thus, in case that multiple of these markers are detected, each one is distinguished by the unique identifier that is codified in its center. Likewise, assuming that a single marker has been added to the scene, its 2D image position ($\vec{m}_i$) is determined by finding black square shapes in the camera image. Moreover, the world coordinate system is centered on the marker (indeed, the world coordinate system is usually centered on the target object), so its 3D coordinates ($\vec{M}_i$) are known. Considering that the marker lies on a plane and the camera is calibrated (K is known), the camera pose is recovered from four $\vec{m}_i \leftrightarrow \vec{M}_i$ correspondences that do not form triplets of collinear points (see Appendix B). More precisely, the correspondences of the four corners of the marker are used to compute the camera extrinsic parameters (Figure 2.15). This is a very fast and accurate technique to build AR applications, which is even executed on mobile platforms (Schmalstieg and Wagner, 2007). As a counterpart, it requires environment adaptation, which is not always possible.

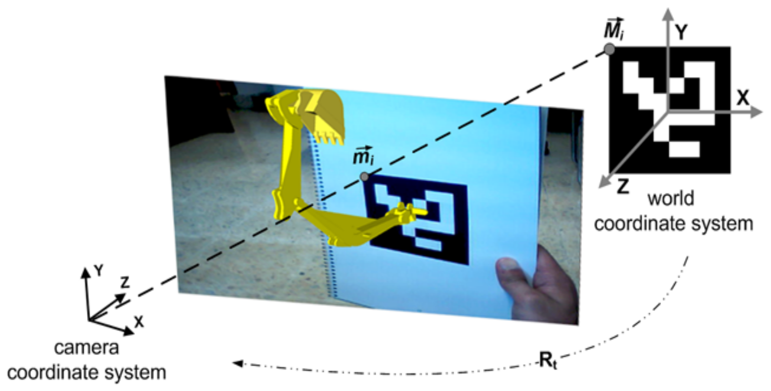


Figure 2.15: Marker tracking system overview. A square marker of the ARToolkitPlus library (Wagner and Schmalstieg, 2007) is shown. A virtual yellow shovel is superimposed in the image.

2.4.2.2 Markerless Tracking

A markerless tracking system does not add artificial markers to the scene, it takes advantage of the visual cues that are naturally in the scene. Depending on whether the scene geometry is known or not, the markerless tracking is divided into two groups (Teichrieb et al., 2007):

1. Structure From Motion

   In Structure From Motion (SFM) approaches the camera movement is estimated while the 3D reconstruction of the scene is performed (Longuet-Higgins, 1981). They estimate both R_t and $\vec{M}_i$. Some visual cues (usually features) are tracked throughout a sequence of images, and their corresponding 2D positions are stored (Figure 2.16). Thus, given a minimum set of two images with its corresponding visual cue positions, the camera pose and the structure of the scene are recovered. For that purpose, the multiple view geometry theory is used (Hartley and Zisserman, 2004), similar to that exposed for stereo systems (Section 2.4.1). It is noteworthy that these solutions only require the data stored in previous frames.

   In addition, two refinements of the SFM algorithm exist to avoid errors due to noisy measurements (Sánchez, 2010): *batch optimisations* and *recursive estimations.*

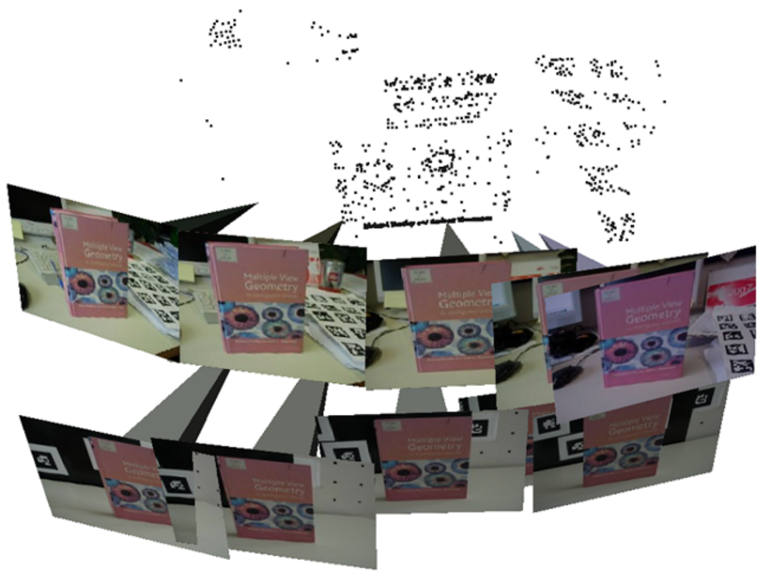


Figure 2.16: 3D scene reconstruction from multiple image views. (Courtesy of (Zach et al.)).[3]

Batch optimisations minimize a cost function that refers to the difference between the projections of an unknown 3D scene points and their known image measurements. They use the *Bundle Adjustment* (BA) technique (Triggs et al., 2000) to jointly optimise the 3D structure and the motion parameters. In their first implementations, the entire video sequence was used for optimisation (Hartley, 1994), making them impractical for real time. Recently, (Klein and Murray, 2007) uses a local BA over the last five selected keyframes and parallel techniques (one thread computes the mapping and other thread performs the tracking) to meet real time requirements.

Recursive estimations are probabilistic methods that have been extensively used in the robotics community in order to address the *Simultaneous Localization and Mapping* (SLAM) problem. They compute an online reconstruction of the scene using recursive Bayesian estimators, formulating the problem as a

[3]C. Zach, A. Irschara, and H. Bischof. What can missing correspondences tell us about 3D structure and motion? *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8. June, 2008.

state-space model. The state-space model is described by a state model, which is associated to the transition over the time of the 3D structure and motion parameters; and a observation model, which is related to the measurements that determine the transition. One of the first successful implementations using this technique is described in (Davison, 2003). Moreover, (Sánchez et al., 2010c) presents an efficient implementation based on GPU. In this approach all the calculus are performed by the GPU pipeline, the 3D reconstruction (Sánchez et al., 2010a; Sánchez et al., 2010b) as well as the camera tracking, making it feasible for real time.

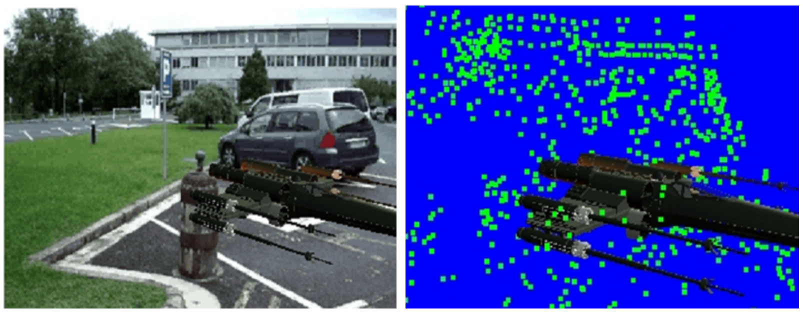


Figure 2.17: SLAM execution, courtesy of (Sánchez et al., 2010c). The original image (left) and the 3D scene reconstruction (right) are shown. A virtual X-Wing (from *Star Wars* movie) is embedded in the real scene.

2. Model-Based

   Model-based techniques store the knowledge about the scene in a 3D model, which is available before the camera tracking begins. The 3D model could be represented by its simple 3D geometry (Drummond and Cipolla, 2002), or by a more detailed description that includes the geometry and the texture of its surface (Vacchetti et al., 2004). In both cases some visual cues that belong to the 3D model are tracked throughout a sequence of images to estimate the camera extrinsic parameters. Depending on the prior knowledge about these visual cues two different techniques are distinguished: *frame-to-frame tracking* and *tracking by detection.*

   Frame-To-Frame tracking, also known as *recursive tracking* or *incremental tracking*, uses the previous pose to estimate the current one. More precisely, the current 2D image locations of the target visual cues are estimated ($\vec{m}'_i$). Some techniques (Barandiarán et al., 2007) combine the previous camera with a predictor[4](LaViola, 2003; Salih and Malik, 2011) to get an estimation of the current camera pose, which is used to project the 3D visual cues and obtain $\vec{m}'_i$. Other approaches, however, use the previous 2D image positions of the visual cues to provide $\vec{m}'_i$ as a function of intensity differences between two consecutive frames (Bleser et al., 2005). A local search is also performed in the vicinity of each $\vec{m}'_i$ to find the correct position of the visual cue in the current image ($\vec{m}_i$). This search is based on the similarity (shape, texture, etc.) between the reference visual cues and those candidates detected in the current image. Once $\vec{m}_i$ is estimated for enough visual cues, the camera pose is computed solving Equation 2.10. Due to its recursive nature, this method suffers from drift (error accumulation) and is sensible to fast camera movements. It also requires an initial pose to start the recursive process, which is obtained manually or using a *tracking by detection* method.

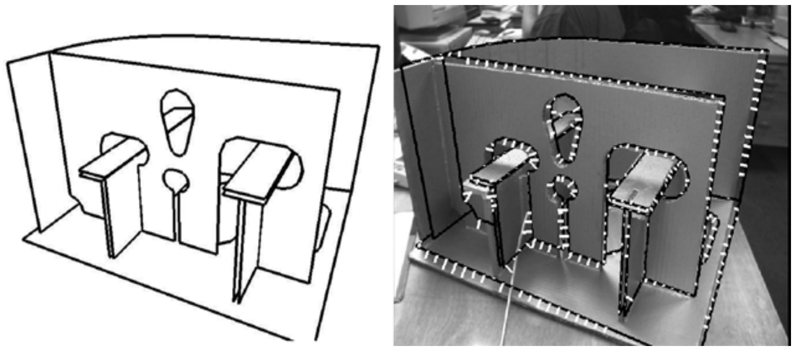


Figure 2.18: 3D wire-frame (left) used to perform edge-based markerless 3D tracking (right) (Drummond and Cipolla, 2002). A local search (white lines) in the vicinity of the previous pose (black wire-frame) is done to find the current camera pose.

[4]A predictor stores the camera pose of previous frames to feed a transition model and provide an estimate of the current camera pose according to the trajectory followed by the camera.

Tracking by detection, sometimes called *3D object recognition*, faces the challenge of computing the camera pose without previous information, so it is used for automatic initialization and recovery from failure. It tries to match some reference visual cues with those detected in the entire image, without limiting the search to a local area imposed by the previous state. Multiple 2D views of the 3D model (*keyframes*) are taken from different positions and orientations during an offline training phase to build a database of 3D visual cues (Figure 2.19). Each 3D visual cue is characterized by a set of 2D views ($\vec{m}'_i$), which try to simulate the online conditions of the 3D visual cue and improve the matching quality between reference ($\vec{m}'_i$) and detected ($\vec{m}_i$) visual cues. Some authors (Rothganger et al., 2006) use the appearance (texture) of the visual cues to establish $\vec{m}'_i \leftrightarrow \vec{m}_i$ correspondences, while others use shape similarity (Wiedemann et al., 2008) to determine positive matches. Considering that each $\vec{m}'_i$ is a 2D view of a 3D visual cue $\vec{M}_i$, Equation 2.3 is solved as long as enough $\vec{m}'_i \leftrightarrow \vec{m}_i$ correspondences are computed.

2.5 Discussion

Multiple methods have been presented in this chapter to recover the position and rotation of the camera from an image. Some of them use multiple cameras to simplify the problem, but they require bulky and expensive hardware. Other methods, however, rely on a single camera (monocular systems) and more sophisticated computer vision techniques. Thus, certain solutions add markers to the scene, obtaining a fast and accurate camera pose at the expense of environment adaptation, which is not always possible. Other alternatives solve both the camera motion and the structure of the scene, for which correspondences of some visual cues are determined throughout a sequence of images. They only need some previous frames as an input, but have high computational cost compared to other solutions. Additionally, they are oriented to scenes rich in texture. Another option that has been proposed stores (before tracking occurs) the knowledge about the scene in a 3D model, which is matched to the visual cues detected in the image to estimate the camera extrinsic parameters. Although the main drawback of this technique is the generation of the 3D

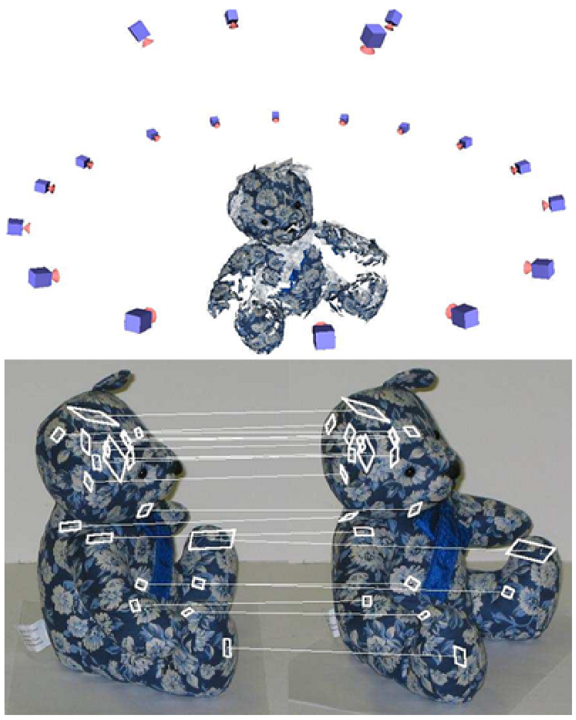


Figure 2.19: 3D object recognition based on appearance (Rothganger et al., 2006). Some keyframes are generate during an offline phase (top) to match reference features (bottom-left) with those detected in the current image (bottom-right).

model, it can handle textureless scenes and offers a robust response in a reasonable amount of time.

The goal that is pursued determines the selection of an appropriate tracking method. Thus, this dissertation focuses on methods oriented to industrial environments, which are characterized by the absence of texture. Moreover, it is oriented to monocular systems, the use of standard hardware components, and real time solutions. Considering these requirements marker tracking systems and model-based markerless tracking methods are the best choices. Both methods are complementary, using the model-based alternative when the environment adaptation is not an option, and using the marker tracking system when the 3D model is not available.

Part II

Proposal

Chapter 3

Marker Tracking

The hallmark of the artist is simplicity
LARRY EVANS

A synthesis of this chapter has been published in:

Álvarez, H. and Borro, D. "Cálculo de la pose de la cámara ante oclusiones de un marcador". In Proceedings of the XVIII Conferencia Española de Computación Gráfica (CEIG'08), pp. 123–132. Barcelona, Spain. September, 2008.

Álvarez, H. and Borro, D. "A novel approach to achieve robustness against marker occlusion". In International Conference on Computer Vision Theory and Applications (VISAPP'09), pp. 478–483. Lisboa, Portugal. February, 2009.

Álvarez, H., Leizea, I., and Borro, D. "A new marker design for a robust marker tracking system against occlusions". Submitted to Computer Animation and Virtual Worlds, 2011.

3.1 Introduction

Marker tracking systems are characterized by adding artificial landmarks (also called *markers* or *fiducials*) to the scene. There are different types of markers (Figure 3.1), such as circular (de Ipiña et al., 2002), planar (Zhang et al., 2002), or based on colour-coded (Mohring et al., 2004). Nonetheless, all of them share the same property: they are easy to detect in the image.

Thus, the information that they provide is extracted and used to calculate the camera pose. Although they are robust, accurate and real time tracking system, their main inconvenience is the environment adaptation, which is not always possible. Moreover, marker occlusion is another shortcoming, as the system fails even if the marker is only slightly occluded, producing an undesirable effect on users, who lose the sense of realism.

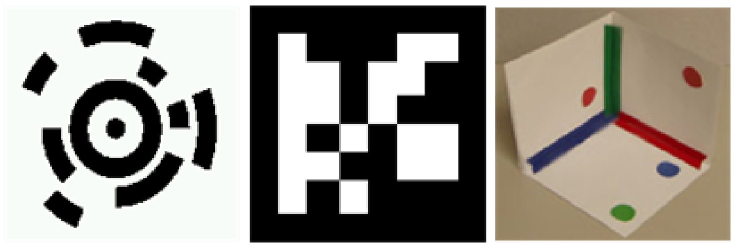


Figure 3.1: Different types of markers.

3.1.1 ARToolkitPlus

ARToolkitPlus (Wagner and Schmalstieg, 2007) is a widely used non-commercial real time marker tracking system that uses black square markers (Figure 3.1, middle) to compute the camera pose. It is an evolution of the ARToolkit library originally developed by Hirokazu Kato (Kato and Billinghurst, 1999), which is considered one of the most influential software libraries in the growth of augmented reality technology. These improvements include the use of BCH markers (Bose, Chaudhuri, Hocquenghem), the use of digital encoding, segmentation based on dynamic thresholds, or the ability to run in mobile devices.

ARToolkitPlus executes sequentially several steps to extract the 6 DOF of the camera (Figure 3.2). Although these steps are not a standard, they are very similar to those executed by other systems, and consequently, they are used here to explain how a common marker tracking system works. These steps are known as ARToolkitPlus *pipeline* and process the input camera image to obtain the camera pose:

1. **Segmentation:** The input grey image is transformed to a binary image using a threshold. Grey levels that are above the threshold are considered as white, and those that are below as black. Note that the

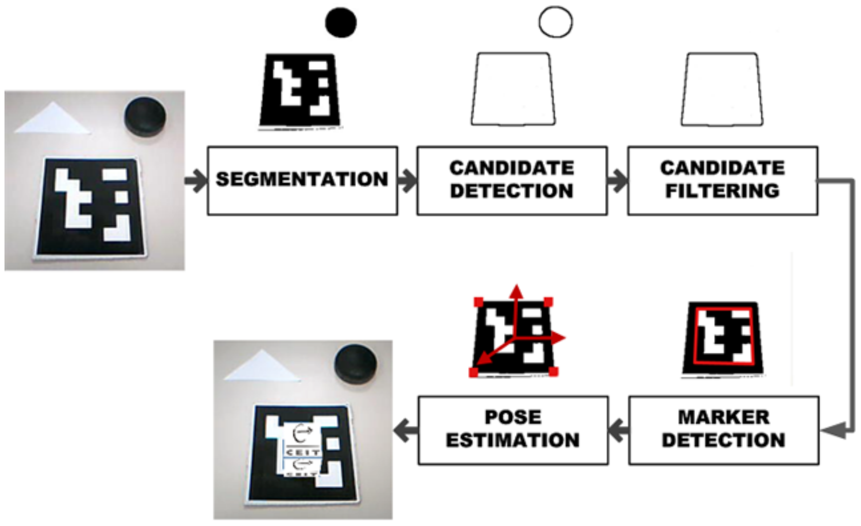


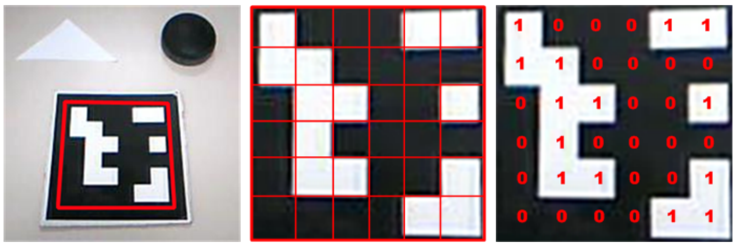
Figure 3.2: ARToolkitPlus pipeline.

threshold is adapted dynamically by using the mean grey value of the marker if it was previously detected, and a random value otherwise.

2. **Candidate Detection:** Since the marker is black, this step groups each set of contiguous black pixels to form a marker candidate.

3. **Candidate Filtering:** Based on the square shape of the marker, those candidates detected in the previous step that do not have a square shape or have a small area are discarded. To determine the shape of a candidate, the number of corners along its boundary are counted, and only those that have 4 corners are considered as squares.

4. **Marker Detection:** Remaining square candidates are unprojected to a normalized surface, making a perspective correction (Figure 3.3). More precisely, the central area of each candidate is unprojected, since this is the region that codifies the marker identifier. This normalized surface is divided into a grid and each cell is interpreted as a single bit so that each marker generates an array of bits that identifies it uniquely. Therefore, a candidate is discarded unless its identifier is equal to one of the marker identifiers. BCH markers use blocks of 6x6 pixels (6x6 bits), 12 bits to codify the marker identifier ($2^{12} = 4096$ different markers), and 24 bits of redundancy for error correction.



(a) Interior area. (b) Unprojected area. (c) Marker identifier.

Figure 3.3: Perspective correction of a BCH marker.

5. **Pose Estimation:** Once the presence of a marker has been identified in the image, the camera pose is obtained using the DLT algorithm (see Appendix B). 3D marker coordinates are known and fixed, since the world coordinate system is centered on the marker. Additionally, 2D image positions of the four corners of the marker are accessible (Candidate Filtering step), so they are matched with their corresponding 3D coordinates. Using these four 2D-3D matches, which do not form triplets of collinear points, the camera pose is recovered uniquely.

3.1.2 Marker Occlusion

Partial occlusion of the marker causes tracking failure, since none of the candidates detected in the image are considered as a marker. Even a small occlusion of the marker is often enough to fail. ARToolkitPlus, for example, fails even when the marker is occluded approximately 2%. This has a negative impact on users, as the virtual information cannot be aligned properly, and consequently, the sense of realism is lost.

Partial occlusion of the marker involves the following problems, or a combination of them:

- **Change in shape:** When the marker is occluded with a bright object, some black pixels are considered as white after the Segmentation step (Figure 3.4, middle). Similarly, when the marker is occluded with a dark object, some black pixels are added around the marker (Figure 3.4, right). Both cases involve an erroneous filtering of candidates because the square shape is distorted.

- **Unknown identifier:** The marker identifier cannot be determined if the interior area of the marker is partially occluded; it is not possible to know which marker is in the field of view of the camera.

- **Few 2D-3D matches:** Sometimes one corner of the marker is not visible due to the occlusion (Figure 3.4), so its 2D image position is unknown. In these cases, one of the four 2D-3D matches that are required to update the camera pose (Pose Estimation step) is lost.

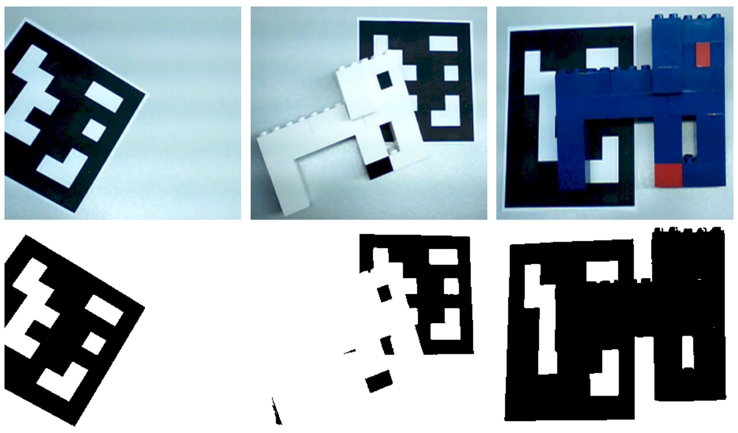


Figure 3.4: Segmentation (bottom) for different occlusions (top).

3.1.2.1 Previous Works

Some authors place multiple markers in the scene to solve the marker occlusion problem (Kato and Billinghurst, 1999). Thus, they increase the probability of finding one visible marker at the cost of more environment adaptation. Continuing with this idea (Tateno, 2007) uses multiple layers of markers. The main disadvantage of this approach is the accuracy waste after switching between layers due to scale change. Moreover, these markers require larger sizes to get similar recognition rates to that of a common single marker.

ARTag is a marker tracking system (Fiala, 2005a) that uses edge segmentation for the identification of target patterns. It is more robust than ARToolkitPlus against bad illumination conditions, and it is capable of closing broken contours (Fiala, 2005b). Nonetheless, it only supports small occlusions, such as putting the fingertip on the marker.

(Wagner et al., 2008) presents two new marker designs. Although not designed for occlusions, they support some restrictive partial occlusion of their interior area, as they use the frame of the marker to codify their digital identification. An incremental tracking of features is also described to handle marker occlusions. It assumes that the marker lies on a textured plane, from which features are detected. As a drawback, this technique is a scene dependent, and according to the authors, it is limited to a few seconds due to drift.

In (Malik et al., 2002) an incremental tracking of features is also applied. In this case, only features that belong to the marker are detected and tracked. A correct spatial configuration of features is required to guarantee a minimum number of features and to obtain an accurate homography[1] between two consecutive frames. As a result, it is not a valid method for all markers, since it depends on the patterns that are inside the marker. It suffers the problem of drift as well.

(Marimon et al., 2007) uses a particle filter to obtain the 6 DOF of the camera when the ARToolkit library fails. It minimizes the reprojection error between the four corners of the marker and those features detected in the current image. The number of particles should be large enough in order to adapt to sudden movements. However, the computational cost of this operation is too high, so the robustness is jeopardized by the number of particles, i.e., by the real time requirement.

3.1.2.2 Proposed methods

Following the argumentation presented above, two new methods have been developed to address the limitation of the marker tracking systems against occlusions: *Occlusion-OBB* and *Occlusion-patches*. Both of them use computer vision techniques to update the camera pose in real time when the marker is partially occluded. Instead of using multiple markers so that there is always one marker visible, these proposals offer more

[1]A projective transformation that maps points on one plane to points on another plane. In computer vision, assuming the pinhole camera model, a homography relates two images of the same 3D plane. See Appendix C.

robustness using a single marker. The environment adaptation has not been jeopardized in favour of robustness against occlusions.

As an overview, Occlusion-OBB uses temporal coherence to track the *oriented bounding box* (OBB, Figure 3.5(b)) of the marker and update the camera pose. It is a valid solution for any marker tracking system that uses parallelogram markers. It is very fast and can be executed on mobile devices, but only updates 4 DOF of the camera. Occlusion-patches, meanwhile, uses a new marker design that provides more information when the marker is partially occluded. It places customizable texture patches along the frame of the marker to have more visible features (with known 3D coordinates) during the marker occlusion, which facilitates the calculation of the camera pose. It is a valid solution for markers that do not take advantage of their frame to codify information. It obtains the 6 DOF of the camera, but compared to Occlusion-OBB requires more computational cost.

It is noteworthy that ARToolkitPlus has been used to prove the validity of the proposed approaches, as it is a widely used non-commercial marker tracking system that uses parallelogram markers that do not take advantage of their frame to codify information.

3.2 Occlusion-OBB

Occlusion-OBB is a method that has been proposed to update the camera pose despite partial occlusion of the marker. It tracks the bounding boxes (Figure 3.5) of the marker using temporal coherence assumptions, without requiring too much image processing.

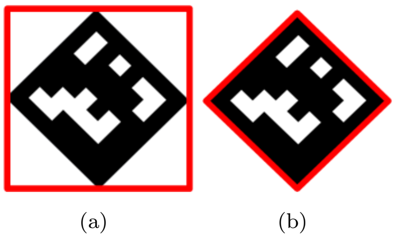


Figure 3.5: Different types of bounding boxes. (a) Axis Aligned Bounding Box (AABB). (b) Oriented Bounding Box (OBB).

Due to its low computational cost, Occlusion-OBB is oriented to mobile devices, whose popularity has increased in recent years. As these devices have limited processing and memory capabilities, it is necessary to assume some simplifications. Indeed, only 4 DOF of the camera pose are calculated.

Figure 3.6 shows how ARToolkitPlus and Occlusion-OBB interact. The first one updates the camera pose and stores some information when the marker is completely visible. This data, called *occlusion data*, contains information about the last two poses. More precisely, it stores the identifier and the AABB (Figure 3.5(a)) of the marker, and the camera pose of the last two poses. Occlusion-OBB, meanwhile, uses the occlusion data to initialize, and it is executed when ARToolkitPlus fails, i.e., when the marker is partially occluded. Note that ARToolkitPlus is executed first every frame, so it recovers the control as soon as the marker is completely visible.

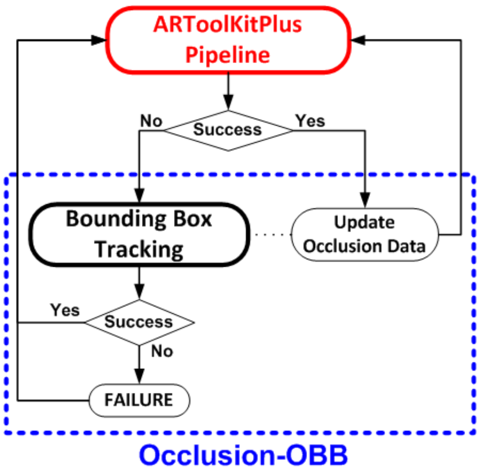


Figure 3.6: Occlusion-OBB overview.

3.2.1 Initialization

Occlusion-OBB requires an initialization step, which is executed every time the marker changes from visible to occluded. This step accesses occlusion data to compute the area of the marker (extracted from the AABB), and activates the new search of the marker (Section 3.2.2). This way, this search is only executed when the marker is occluded, i.e., it is executed on demand.

The trajectory set by the last two frames is extrapolated in order to update the camera pose: $R_t^i = R_t^{i-1} + (R_t^{i-2} - R_t^{i-1})$, where R_t^k is the camera pose of the frame k. This is a good estimation only for the first frame of Occlusion-OBB because the trajectory may change considerably for longer time intervals.

3.2.2 Marker Search

ARToolkitPlus fails against a partial occlusion of the marker because none of the candidates detected are considered as a marker (Section 3.1.1). However, a part of the marker remains visible in the image, which can be exploited to estimate the camera pose.

Following this reasoning, a new marker search is executed, which processes those candidates detected and discarded by the ARToolkitPlus pipeline. This new technique performs a less restrictive search and uses temporal coherence to find the presence of the marker: *assuming that the frame-rate is high, the position and area of the marker will not suffer too much variation between two consecutive frames.* As a result, the new search focuses on candidates that are close to the last position of the marker and whose area has not changed too much (Algorithm 3.1).

Algorithm 3.1 Marker search executed by Occlusion-OBB.

```
1: $selectedCandidate \leftarrow NULL$;
2: for all $c_i$ in CANDIDATES do
3:     if $dist(c_i, Marker) <$ SEARCH_WINDOW then
4:         if $dist(c_i, Marker) < dist(selectedCandidate, Marker)$ then
5:             if $|area(Marker) - area(c_i)| \,/\, area(Marker) < 0.25$ then
6:                 $selectedCandidate \leftarrow c_i$;
7:             end if
8:         end if
9:     end if
10: end for
```

Condition 3 of Algorithm 3.1 removes candidates that are not close to the position of the marker in the last frame ($Marker$), where $dist(i, j)$ is the distance between the centers of i and j, and SEARCH_WINDOW is a threshold that sets the maximum search distance. Large search distances increase the probability of selecting an erroneous candidate, but can withstand sudden movements. Good results have been obtained with searches that do not exceed 75 pixels at 320x240 resolution. Condition 4 of Algorithm 3.1 selects the candidate that is closest to $Marker$, provided that the area has not changed more than 25% (condition 5 of Algorithm 3.1). Note that the AABB of each candidate is calculated so that the width and height of the AABB are multiplied to compute $area$.

If no candidate is selected this search is also executed in the two incoming frames to avoid the effects of a possible image blur generated by a fast camera movement. It is noteworthy that this recovery step is only executed twice because temporal coherence cannot be applied for longer periods of time. In case that these additional search attempts do not work Occlusion-OBB fails and waits until ARToolkitPlus works fine again.

Occlusion-OBB continues the processing and estimates the camera pose for the candidate that satisfies all the conditions of Algorithm 3.1. Additionally, as a partial occlusion of the marker generally means that the identifier of the marker cannot be determined (Section 3.1.2), the last identifier stored by ARToolkitPlus is returned, which corresponds to the correct one.

3.2.3 Pose Estimation

The camera pose is estimated once one of the candidates is considered as the target marker. Different procedures are used to calculate translation and rotation values.

3.2.3.1 Translation

2D displacement of the center of the marker (distance between the selected candidate and the previous position of the marker) is used to estimate 3D translation in X (tX') and Y (tY') (Equations 3.1 and 3.2). More specifically, measured pixel translation of the center (Δx, Δy) is extrapolated to 3D translation using precomputed 2D-3D proportions (Section 3.2.4).

$$tX' = X_{2D-3D} * \Delta x, \tag{3.1}$$

$$tY' = Y_{2D-3D} * \Delta y, \tag{3.2}$$

where Δx and Δy are the pixel translation of the center of the marker for x and y image axis, respectively, and X_{2D-3D} and Y_{2D-3D} are the 2D-3D proportions for each axis.

For each candidate its AABB and center is calculated. Indeed, the center of the marker is approximated by the center of the AABB. Both points suffer similar movement, but the computation of the center of the AABB is cheaper, as it does not require any image processing. Note that the AABB only surrounds the visible part of the marker (Figure 3.7), so the dimensions of the AABB vary for different degrees of marker occlusion. Additionally, the center of the AABB is only displaced by $n/2$ pixels in x coordinate when the width of the AABB is modified by n pixels (analogous for y and height). Therefore, using the center of the AABB in Equations 3.1 and 3.2, Δx is multiplied by 2 when the marker is occluded with the vertical image axis, and Δy is multiplied by 2 when the marker is occluded with the horizontal image axis.

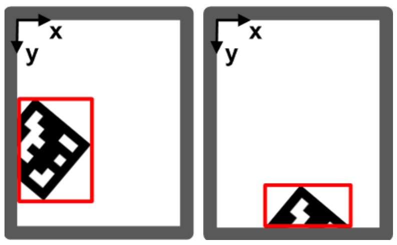


Figure 3.7: Occlusion with vertical (left) and horizontal (right) image axes. Red rectangle represents the AABB of the visible part of the marker.

Z translation modifies the size of an object in the image. Based on this statement, the changes in the dimensions of the AABB are multiplied by a 2D-3D proportion to obtain the 3D Z translation (tZ', Equation 3.3). Similar to X and Y translations, Z translation depends on the image axis in which the marker is occluded. The width of the AABB is analysed when the marker is occluded with the horizontal image axis, and the height of the AABB is analysed when the marker is occluded with the vertical image axis.

$$tZ' = Z_{2D-3D} * diff(AABB_{candidate}, AABB_{marker}), \tag{3.3}$$

where $diff(i, j)$ calculates the difference in dimensions between i and j

according to the occlusion axis (horizontal or vertical).

All this procedure generates ambiguity when the marker is occluded in the corners of the image. In these regions is not possible to distinguish between translations. The dimensions of the AABB can vary due to (X,Y) translations or Z translations. In order to clarify this confusion, a movement in the image corner is interpreted as Z translation when both dimensions of the AABB increase or decrease similarly: $|width(AABB_{candidate})/width(AABB_{marker}) - height(AABB_{candidate})/height(AABB_{marker})| \sim 0$. This is the most generally interpretation, as it is unlikely to make two translations almost identical at the same time in X and Y.

3.2.3.2 Rotation

The new rotation parameters are only updated for Z axis, since there is not enough information for other axes (Section 3.2.6). The rotation in Z axis corresponds to the rotation of the OBB of the marker in the last two consecutive frames. The AABB is used to calculate the OBB, as it is shown in Figure 3.8.

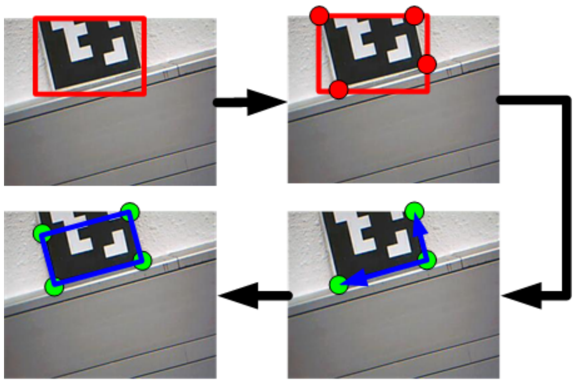


Figure 3.8: OBB computation using the AABB as a starting point.

Starting in each AABB corner the image features that are closest to each AABB corner are chosen by moving in horizontal and vertical directions. This procedure results in 4 points (red points of Figure 3.8), but only 3 of these 4 points belong to the OBB, so the 3 points that generate two perpendicular vectors are selected (green points of Figure 3.8). The last point of the OBB corresponds to the parallelogram rule. This is executed very fast because the image area defined by the AABB is only processed. Additionally, the image that has been already segmented by ARToolkitPlus is analysed.

To calculate the rotation between the OBB of the previous frame (OBB_{prev}) and the OBB of the current frame (OBB_{curr}) the angle formed by the same side of both OBBs is measured. Corners of OBB_{curr} are stored in clockwise and all the possible configurations (permutations) are compared to the point configuration of OBB_{prev} to guarantee that both OBBs have the same point configuration. Once the point configuration of OBB_{curr} that is closest to OBB_{prev} is selected, the first two corners of OBB_{prev} and OBB_{curr} are used to calculate two vectors ($\vec{v_1}$ and $\vec{v_2}$ respectively, Figure 3.9), which represent the same side .

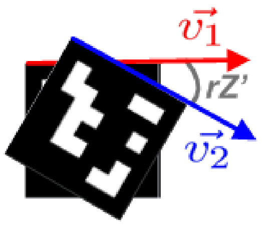


Figure 3.9: Rotation in Z axis (rZ') between two OBBs.

To obtain the rotation angle in Z axis (rZ'), the dot product of $\vec{v_1}$ and $\vec{v_2}$ is calculated (Equation 3.4).

$$rZ' = acos\left(\frac{\vec{v_1} \cdot \vec{v_2}}{\|\vec{v_1}\|\|\vec{v_2}\|}\right). \tag{3.4}$$

3.2.3.3 Pose Update

Once the new translations (tX', tY' and tZ') and rotations (rZ') are known, the previous camera pose ($[R|\vec{t}]$) is transformed into the new camera pose ($[R'|\vec{t'}]$):

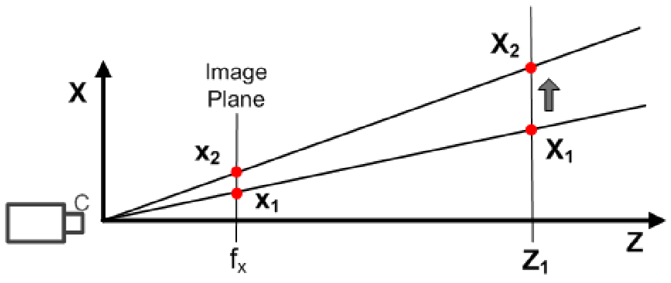
$$t'^T = t^T + (tX', tY', tZ')^T. \tag{3.5}$$

$$R' = \begin{bmatrix} \cos rZ' & -\sin rZ' & 0 \\ \sin rZ' & \cos rZ' & 0 \\ 0 & 0 & 1 \end{bmatrix} * R. \tag{3.6}$$

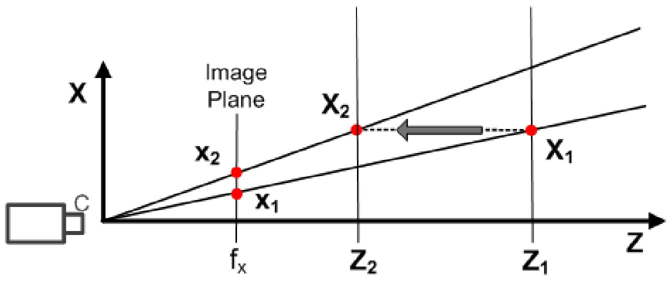
Furthermore, the occlusion data is updated with information concerning the current frame. The candidate that has been selected replaces the data of the marker, ensuring the evolution of the marker appearance for subsequent frames.

3.2.4 Proportion Values

X_{2D-3D} (Equation 3.1), Y_{2D-3D} (Equation 3.2) and Z_{2D-3D} (Equation 3.3) are calculated during an offline step, using the pinhole camera model and the similar triangles rule. Pure translations are assumed to simplify calculations and get an approximated proportions. This implies that when a translation in one axis is performed, the other two translations are set to 0 (Figure 3.10).



(a) Translation in X axis (analogous for Y axis).



(b) Translation in Z axis.

Figure 3.10: Translations in a pinhole camera model.

Assume that f_x and f_y are the focal lengths, (X_1,Y_1,Z_1) and (X_2,Y_2,Z_2) represent 3D coordinates of a point in two consecutive frames, and (x_1,y_1) and (x_2,y_2) are their corresponding projections. X_{2D-3D} and Y_{2D-3D} consider that there is no translation in Z axis ($Z_1 - Z_2 = 0 \rightarrow Z_1 = Z_2$), and are given by:

$$X_{2D-3D} = \frac{(X_2 - X_1)}{(x_2 - x_1)} = \frac{(X_2 - X_1)}{\frac{f_x}{Z_1} * (X_2 - X_1)} = \frac{Z_1}{f_x} = \frac{Z_2}{f_x}. \tag{3.7}$$

$$Y_{2D-3D} = \frac{(Y_2 - Y_1)}{(y_2 - y_1)} = \frac{(Y_2 - Y_1)}{\frac{f_y}{Z_1} * (Y_2 - Y_1)} = \frac{Z_1}{f_y} = \frac{Z_2}{f_y}. \tag{3.8}$$

Similarly, Z_{2D-3D} sets to 0 the translation in X and Y axes, which implies the following relations:

$$X_1 = X_2 \Rightarrow \frac{Z_1 * x_1}{f_x} = \frac{Z_2 * x_2}{f_x} \Rightarrow Z_2 = \frac{x_1}{x_2} * Z_1. \tag{3.9}$$

$$Y_1 = Y_2 \Rightarrow \frac{Z_1 * y_1}{f_y} = \frac{Z_2 * y_2}{f_y} \Rightarrow Z_2 = \frac{y_1}{y_2} * Z_1. \tag{3.10}$$

Additionally Z_{2D-3D} is divided in Z^x_{2D-3D} and Z^y_{2D-3D}, since the image resolution is not necessarily square ($f_x \neq f_y$). A greater/lower number of pixels in one image axis implies that a difference of one pixel is considered as a lower/greater increase in 3D motion. Taking everything in consideration, Z^x_{2D-3D} and Z^y_{2D-3D} are given by:

$$\begin{aligned} Z^x_{2D-3D} &= \tfrac{Z_2 - Z_1}{x_2 - x_1} = \tfrac{\frac{x_1}{x_2} * Z_1 - Z_1}{x_2 - x_1} = \tfrac{\left(\frac{x_1}{x_2} - 1\right) * Z_1}{x_2 - x_1}. \\ Z^y_{2D-3D} &= \tfrac{Z_2 - Z_1}{y_2 - y_1} = \tfrac{\frac{y_1}{y_2} * Z_1 - Z_1}{y_2 - y_1} = \tfrac{\left(\frac{y_1}{y_2} - 1\right) * Z_1}{y_2 - y_1}. \end{aligned} \tag{3.11}$$

The use of Z^x_{2D-3D} or Z^y_{2D-3D} depends on the occlusion axis. i.e., Z^x_{2D-3D} and Z^y_{2D-3D} are selected for occlusions of the marker with the horizontal and vertical image axis respectively.

Equations 3.7, 3.8 and 3.11 vary for different Z values (distance between the camera and the marker). Thus, different proportion values are computed for different Z values during the offline step, and the proportion that is closest to the current Z value is chosen during the online execution.

3.2.5 Experiments and Results

Occlussion-OBB has been executed with two different hardware devices: a PC (P4 3GHz, 2GB RAM) and a PDA (Dell Axim X 50v). Table 3.1 shows the frames per second (fps) that takes an ARToolkitPlus execution, with and without Occlusion-OBB.

	ARToolkitPlus + Occlusion-OBB		
	no occlusion	occlusion	ARToolkitPlus
PC	27-29	27-29	27-29
PDA	6	8	8

Table 3.1: ARToolkitPlus fps, with and without Occlusion-OBB.

When ARToolkitPlus calculates the camera pose (the marker is completely visible, no occlusion), Occlusion-OBB jeopardizes the frame-rate of the PDA execution ($\sim$ 2 frames per second) due to the storage of the occlusion data. On the contrary, ARToolkitPlus fails when the marker is partially occluded, and the camera pose is calculated by Occlusion-OBB. Notice that the pipeline of ARToolkitPlus is not executed completely (Candidate Filtering step fails, Section 3.1.1), so this saving of time is used by Occlusion-OBB, and the total fps remains constant.

The response of Occlusion-OBB against different conditions is presented in Figure 3.11. It is a valid method for both PDA (left column of Figure 3.11) and PC (right column of Figure 3.11). Additionally, top right of Figure 3.11 demonstrates that Occlusion-OBB supports severe occlusions. Right column of Figure 3.11 also indicates that the new marker search explained above (Section 3.2.2) is a robust technique that is not confused by objects with similar shape and colour.

Occlusion-OBB has also been compared to ARTag, which is a commercial marker tracking software that supports tiny occlusions. Occlusion-OBB is more robust, as it is demonstrated in Figure 3.12 by obtaining satisfactory results for those cases in which ARTag fails.

3.2.6 Observations and Limitations

Rotation in Z axis modifies the size of the AABB, so it could be interpreted as translation in Z axis too. In order not to mix both movements a

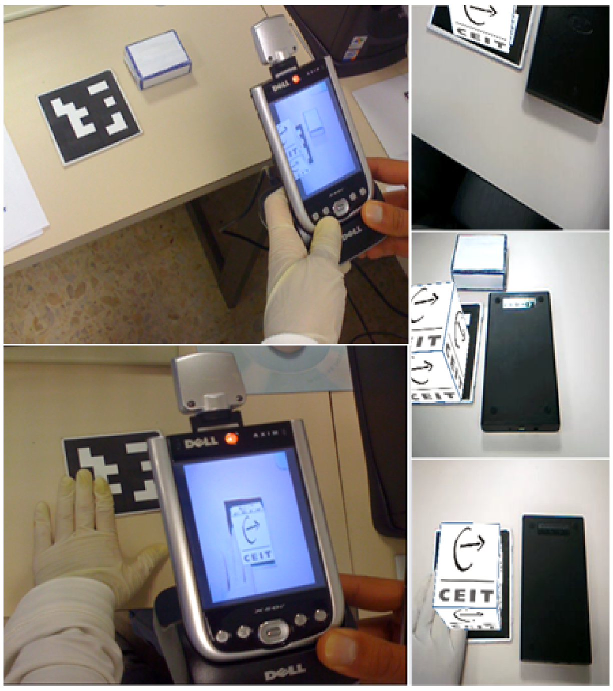
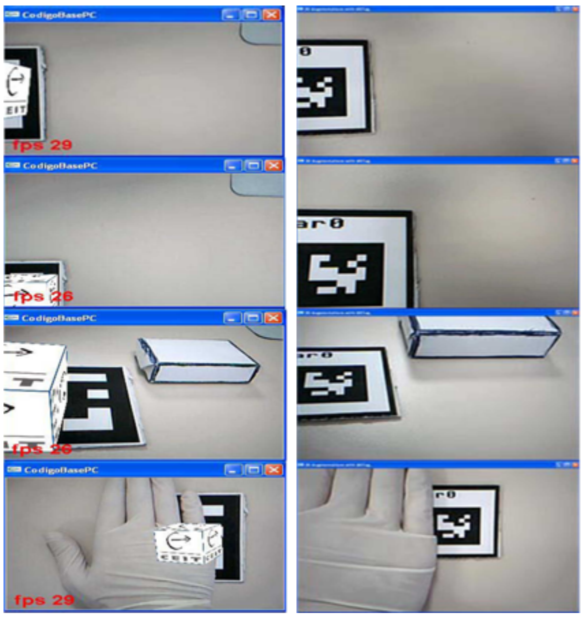


Figure 3.11: Occlusion-OBB against different occlusions.

constraint has been set: *Translation in Z axis is executed when the rotation angle is less than β degrees*, where β is a threshold. Good performance has been obtained with a value between 2 and 3 degrees.

In this approach, the variations of the AABB determine the translation in Z axis (Section 3.2.3). Similarly, they could be used to calculate the rotation in X and Y axes (difference in the width and height of the AABB respectively), provided that the translations in Z axis are forbidden. Nonetheless, translation in Z axis is a more important and usual movement.

As explained in Section 3.1.2, the shape of the marker is distorted after the Segmentation step when the user occludes the marker with black objects. Indeed, the AABB of the marker increases considerably, causing the failure of Occlusion-OBB (Figure 3.13(a)). An initial prototype that uses edge segmentation instead of binary segmentation has been implemented to overcome this problem. As ARTag does, image edges are detected and those closed contours are interpreted as candidates ((Figure 3.13(b))). As a counterpart, it has a high computational cost, so it would not be oriented



(a) Occlusion-OBB. (b) ARTag.

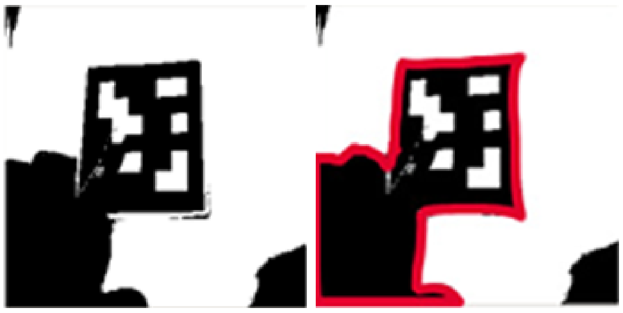
Figure 3.12: Occlusion-OBB vs ARtag.

to mobile platforms.

Although Occlusion-OBB calculates 4 DOF, it is enough for some tasks. Moreover, this technique does not need especial markers or extra prepared environments, so it can be used in scenes that are already prepared for ARToolkitPlus, without installing anything. In fact, it is a fast method that can be used directly by any marker tracking system based on squared markers.

3.3 Occlusion-patches

Occlusion-patches is based on a new marker design that has been proposed to overcome the problem of marker occlusion. It can be adapted to any marker tracking system that uses its central area to codify the digital identification (ARToolkitPlus, for example). It takes advantage of an untapped frame to place some textures that will be tracked during marker


(a) Binary segmentation (left) and Candidate Detection (right).

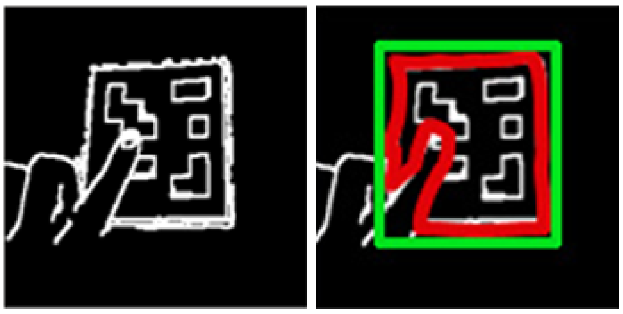

(b) Edge segmentation (left) and Candidate Detection (right).

Figure 3.13: Binary segmentation vs Edge segmentation. Green rectangle indicates the success of the Candidate Detection step.

occlusion. This way, the response of the marker tracking system is not jeopardized when the marker is visible and the camera pose is also updated when the marker is occluded. A robust tracking technique exploits the information that these textures provide to calculate the 6 DOF of the camera in real time. The extra information that this new marker design provides can be used to develop new human-machine interfaces as well.

3.3.1 Justification of the New Design

Partial occlusion of the marker causes tracking failure because few 2D-3D matches are found for the estimation of the camera pose (Section 3.1.2). To overcome this constraint there should be as many features as possible with known 3D coordinates lying on the marker surface. This would increase the probability of finding, at least, four 2D-3D matches during marker occlusion. Because of that, texture patches have been added to the frame of the marker (middle image of Figure 3.14), which is an area not used by the ARToolkitPlus library. Indeed, these textures are customizable, which lets users make their own designs.

Each texture is painted with a different colour (right image of Figure 3.14) to accelerate the matching step. The colour codification lets distinguish the texture that each point belongs to, while different texture patterns guarantee that not all the points look like the same point (see Section 3.3.4.2). It is also noteworthy that the frame of the marker has been slightly expanded (the new marker size is 12% larger) to get reasonable texture sizes and to maintain a thin black outer border. This black outer border benefits the segmentation step of the ARToolkitPlus pipeline (Section 3.1.1), as the square shape is more easily detected despite the presence of the textures.

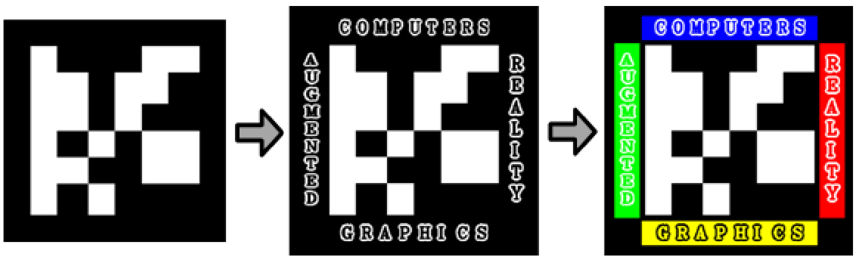


Figure 3.14: Evolution of the new marker design.

The same texture configuration can be used for all markers or a unique set of textures can be designed to build a special marker. This decision only modifies the training phase (Section 3.3.3).

3.3.2 Algorithm Overview

Figure 3.15 shows the interaction between Occlusion-patches and ARToolkitPlus. The data associated to the current state is stored (*Update Occlusion Data*) when the ARToolkitPlus pipeline is executed successfully. This data is used to initialize the Occlusion-patches when ARToolkitPlus fails, and thus, it guarantees that the occlusion state is updated every time the marker is visible.

Occlusion-patches combines two different tracking methods:

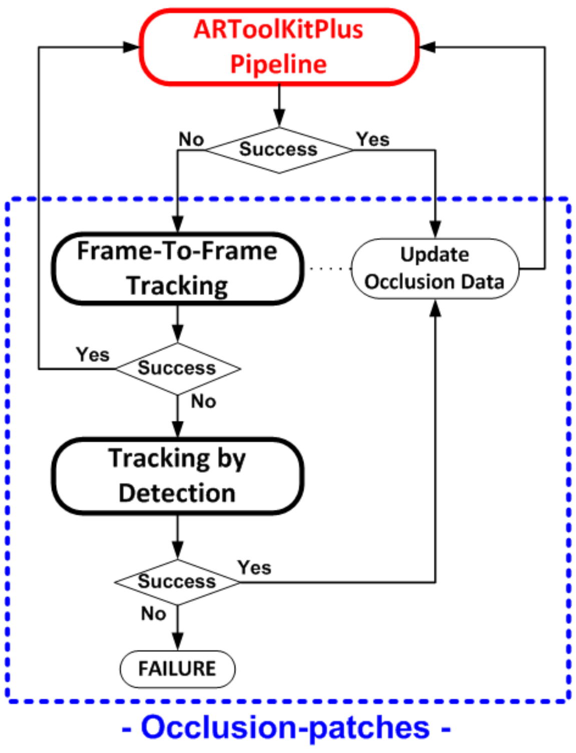


Figure 3.15: Occlusion-patches overview.

frame-to-frame tracking and *tracking by detection*. The first one is based on temporal coherence, which provides a fast and robust tracking against no-strong camera movements. The second tracking method, meanwhile, computes the pose regardless of the movement of the camera, but requires more computational resources. This way, both tracking methods are combined to obtain a robust tracking. The frame-to-frame tracking is used as long as possible, while the tracking by detection is used for failure recovery. Furthermore, the tracking by detection method is based on appearance, so it requires a database of feature descriptors that belong to the marker. This database is computed during an offline phase (Section 3.3.3.3), which is executed only once for each marker design.

In the next section the offline phase of Occlusion-patches is detailed, while in the subsequent sections the online phase is described by explaining how the two tracking methods mentioned above interact.

3.3.3 Offline Phase

The offline phase builds all the data that the tracking by detection method requires to be executed in the online phase. For that purpose, the marker is trained along a set of keyframes. These keyframes are processed, and the 2D features that belong to the surface of the texture patches are back-projected, obtaining their corresponding 3D values. These features should be as distinguishable as possible in order to generate a unique descriptor for each of them. In addition, these 3D points and descriptors are indexed in a database, which is used to get a set of matches during the online phase. All the steps involved in this phase are shown in Figure 3.16.

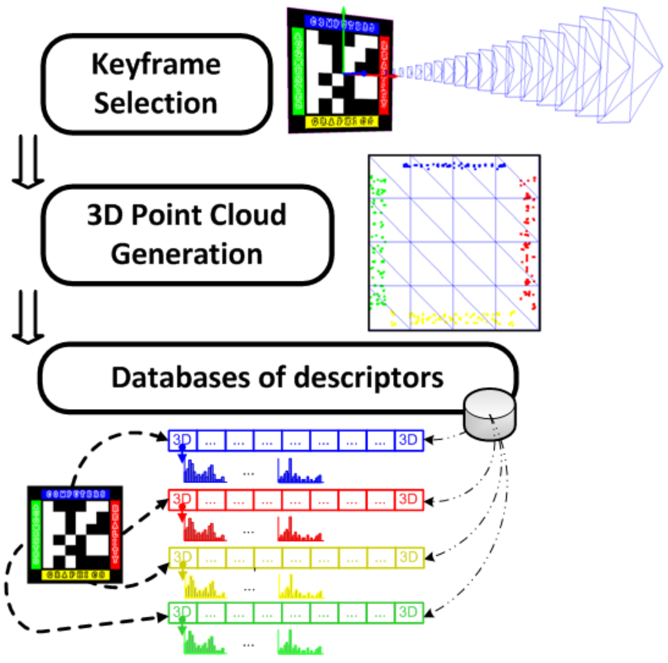


Figure 3.16: Offline phase of Occlusion-patches.

3.3.3.1 Keyframe Selection

The main goal of this step consists of generating a set of keyframes of the marker. Thus, the marker is rendered from different points of view in a virtual scene, i.e., synthetic images are used instead of taking real images of the marker. This procedure lets minimize the user intervention during the training phase.

The number of keyframes depends on the set of movements that the camera covers at runtime, without exceeding the maximum storage of the database. Moreover, due to the real time requirements of the online phase, the FAST operator is used to process the keyframes, which does not provide scale information (see Section 3.3.3.2). Hence, as it is done in (Wagner et al., 2010), the marker is trained along multiple scales to solve the scale ambiguity (see Appendix A). However, very small scales are discarded because of the poor quality of feature detection, which makes difficult to obtain a set of good matches (Section 3.3.6). Additionally, keyframes that put the marker in front of the camera are only considered, without applying rotations. This is a valid assumption for an object that lies on a plane, and it has been successfully applied by other authors (Xu et al., 2008).

3.3.3.2 3D Point Cloud Generation

Once the set of keyframes is obtained, interest points that lie on the surface of the marker are extracted for each keyframe. The FAST operator is used to detect these 2D features since it provides good repeatability and low computational cost. Note that not all the features that lie on the marker are processed, but rather a filter is passed to get those that belong to one of the four textures of the marker. As the position of the textures is fixed and known, a mask is applied to identify the pixels that belong to those areas. Furthermore, the 2D features are divided into four different clusters according to the texture they belong to.

Remaining 2D features are back-projected to obtain their 3D values, which is a necessary step because a set of 2D-3D matches are required at runtime to calculate the camera pose. This way, all the virtual camera poses that define the set of keyframes are used to render the marker with a color codification. In fact, the marker is modelled as a triangle mesh and each triangle is rendered with a unique colour, so the location of each 2D feature points to a unique colour, which is used to index the corresponding facet. Once each 2D feature is matched to its corresponding 3D triangle, its 3D values are given by barycentric coordinates (Gall et al., 2006).

As a result of this procedure four 3D point clouds are created, one for each texture of the marker (Figure 3.16).

3.3.3.3 Databases of Descriptors

This step receives four 3D point clouds (one for each texture of the marker) and computes the descriptors that characterized the appearance of each point. SIFT has been chosen for that purpose, since it has demonstrated high performance compared to other local descriptors (Mikolajczyk and Schmid, 2005). More precisely, a simplified version of SIFT has been used, called *simplified-SIFT* (Appendix A). In summary, besides replacing the expensive Difference of Gaussians feature detector by the lightweight FAST operator, parallel techniques are applied to satisfy real time requirements. Moreover, the parameterisation of simplified-SIFT is application dependent, so an in depth study of these parameters is presented in Section 3.3.6 according to the proposed marker design.

Simplified-SIFT implies that no scale information is obtained during the extraction of features. Therefore, an scale-space like (Wagner et al., 2010) is implemented to avoid scale ambiguity and get better matching results at runtime. As a result, each 3D point is related to multiple descriptors, one for each image that results from applying different Gaussian filters to each keyframe. In fact, 8 Gaussian filters with a standard deviation factor of $1/\sqrt{2}$ the previous one (recursively) have reached a good compromise between storage and computational cost. These scale-space levels, together with the marker scales of Section 3.3.3.1, try to cover the expected scale ranges at runtime.

Once all descriptors of all 3D points are computed, they are indexed in a database using kd-trees[2]. A different database is created for each texture, resulting in four different databases. It is noteworthy that this arrangement offers fast and robust matches during the online phase, since an input feature is only matched with its corresponding texture database. Indeed, the FLANN library (Fast Library for Approximate Nearest Neighbors) integrated in the widely known OpenCV library (Bradski and Kaehler, 2008) has been used to build these databases, which offers an efficient approximated nearest neighbour.

[2]In this thesis Kd-trees are space-partitioning data structures that organize features based on their descriptors (multi dimensional space). They offer an efficient nearest neighbour search.

3.3.4 Online Phase

This phase is executed every time ARToolkitPlus does not detect the marker. It uses the data provided by the last successful execution of ARToolkitPlus to initialize the frame-to-frame tracking (Figure 3.15). This technique works fine while no fast camera movements are applied, otherwise the tracking is lost. Therefore, in case of rapid camera movement the tracking by detection method is called, which uses the databases of descriptors created in the offline phase (see previous section). This second method is based on appearance and supports fast camera movements, so it is used as a recovery tool. If the tracking by detection computes the camera pose successfully, then the system returns to the initialization step of the frame-to-frame tracking, starting the whole process over again. Likewise, if the marker is not detected after a few attempts (*Failure* status of Figure 3.15), then the Occlusion-patches does not update the camera pose and the system waits until ARToolkitPlus is successfully executed. It is noteworthy that the ARToolkitPlus pipeline is executed every time to check the marker visibility, and thus, Occlusion-patches is disabled as soon as ARToolkitPlus detects the marker.

3.3.4.1 Frame-To-Frame Tracking

The first step of the frame-to-frame tracking consist of initializing some control points that will be tracked in subsequent frames. The last successful camera pose and frame that were registered are used to back-project features that lie on the surface of the marker as well as computing their representative descriptors (Figure 3.17). The FAST detector is used to extract these features, while the camera pose is provided by ARToolkitPlus or by tracking by detection after recovering from a failure. The back-projection process is similar to that explained in Section 3.3.3.2, using barycentric coordinates to compute 3D values.

Back-projected features and descriptors are not the same as those generated in the offline phase. These features belong to the entire surface of the marker, i.e., they are not limited to the frame of the marker. In addition, they do not use simplified-SIFT descriptor, but a simple descriptor (called *GREY*) that is based on the grey values of the surrounding patches. It is similar to normalized cross correlation (NCC), but using a Gaussian weight. More precisely, a sparse 11x11 sample grid centered on each feature is used

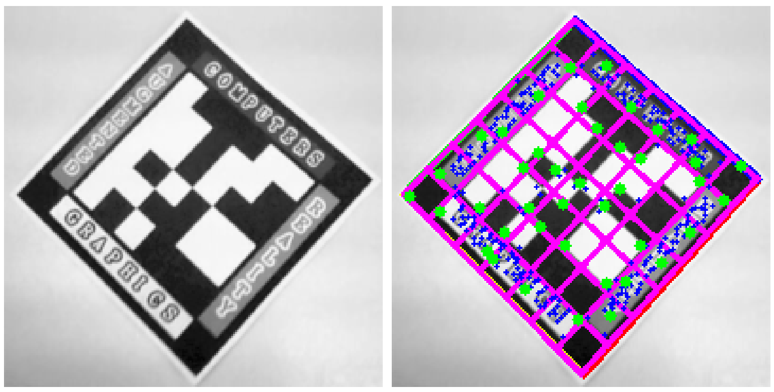


Figure 3.17: Frame-To-Frame tracking initialization. Image of the last successful execution of ARToolkitPlus (left). Feature detection (blue points) and back-projection of the strongest features (green points) that are uniformly distributed (right).

to build a 121 bin descriptor (Figure 3.18). Each bin value corresponds to the grey value of the corresponding sample, which is multiplied by a Gaussian weight that is proportional to the distance to the center. Each descriptor is also normalized to be invariant against illumination changes. This descriptor is not as robust as simplified-SIFT, but it has demonstrated considerable robustness and low computational cost (Section 3.3.6).

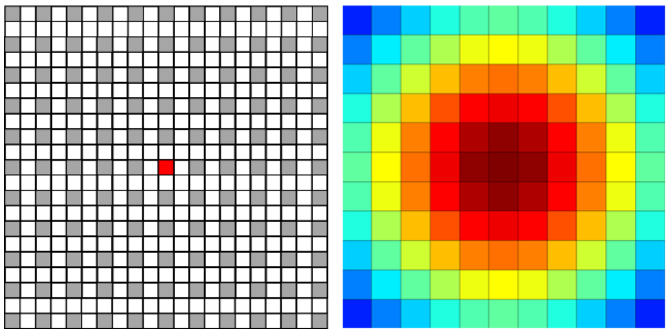


Figure 3.18: Sparse 11x11 sampling grid (left) and gaussian weight (right) used by GREY. Red indicates more weight.

In order to reduce the computational cost, the maximum number of back-projected features has also been limited to 49. The marker is divided into 49 equal cells (7x7 grid), and only the strongest feature is retained for each cell (Figure 3.18). This is a fast technique to limit the maximum number of features and maintain an uniform spatial distribution of features, which favours the response against occlusions because it does not matter which side of the marker is occluded.

This initialization step is executed every time the marker changes from visible to occluded. It guarantees that all features and descriptors are updated with the last visible frame, and thus, they are adapted to possible environmental changes such as different illumination conditions.

Once the initialization step is finished, all back-projected features that are visible in the image are tracked in the subsequent frames. An incremental tracking similar to those presented in (Malik et al., 2002; Wagner et al., 2008) is used. First, the new locations of the features are predicted using a pyramidal optical flow (Section 2.3.1.3). This technique provides a good starting point for the second step, which searches in the vicinity of the predicted locations the presence of points with similar GREY descriptors (Figure 3.19). This way, a feature is considered successfully tracked if a corner is found in a 20x20 window centered on the predicted location, and whose descriptor is highly correlated to the original descriptor (correlation value higher than 0.9). This step removes outlier matches and minimizes drift, which is a typical problem faced by this type of techniques.

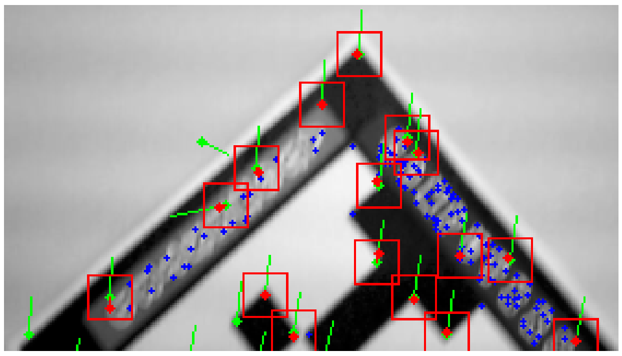


Figure 3.19: Refinement of the frame-to-frame tracking. Green lines indicate optical flow prediction, while red dots correspond to the final location. Red squares define the area for refinement. Features without red square are considered as outliers.

The location of the features that pass the correlation rule is replaced by the location of the corner they are matched to. However, some of these remaining matches can be outliers, so a hypothesize-and-verify technique like RANSAC (Fischler and Bolles, 1981) is used to compute the affine transformation (see Appendix C) that best fits the movement of inlier features between the two consecutive frames. Note that a minimum set of inlier matches (6 for this dissertation) are required to validate the correctness of the current tracking.

The estimated affine transformation provides the new 2D locations of the 3D control points that have been computed during the initialization, so this new set of 2D-3D matches is used to update the camera pose. Moreover, these 3D control points are projected with the new camera pose, and the resulting 2D locations are used as a starting point for the next frame. Additionally, a refinement step similar to that mentioned above is performed. If a corner with similar GREY descriptor appears in the 10x10 vicinity of the projection of each control point, then its position and descriptor is replaced. Otherwise, both position and descriptor are not changed, although they are considered for the next frame as well. This last refinement step ensures that descriptors evolve and improves the quality of the optical flow for the next frame. Should be noted that the simple projection of a 3D point may fall in the middle of a homogeneous image area if a non-accurate camera pose is used, which is not an optimal scenario for optical flow techniques (Section 2.3.1.3).

At least 15% of control points that are visible must be updated by the last refinement procedure to validate the correctness of the current tracking. This requirement, together with the affine restriction, identifies bad tracking frames due to fast movements or drift and decides when to execute the tracking by detection method.

3.3.4.2 Tracking by Detection

The tracking by detection method recovers the camera pose without any user intervention in case that the frame-to-frame tracking fails. In order to achieve this, features extracted in the current camera image are matched with those indexed in the database during the offline phase.

Each of the four textures of the marker has been codified with a unique colour, building a different database for each one (Section 3.3.3.3). Thus, for each feature detected in the current image the texture it belongs to is calculated. To do this, for each feature the number of pixels of its vicinity

(10x10 window) that belong to each colour is counted. Furthermore, to determine the colour of each pixel its corresponding values in the HSV colour space (Hue-Saturation-Value) are considered. If the saturation or value component of a pixel is in the extreme of its range, then this pixel is not considered as coloured, but rather as a white or black pixel. Otherwise, the hue channel specifies the colour of a pixel. Moreover,a feature is considered as coloured if half of their surrounding pixels belong to the same colour; i.e., there is a dominant colour. This colour codification step is very useful since it reduces the number of matching combinations by 75% comparing each feature with only one of the four textures. It also rules out too many points that do not belong to the frame of the marker.

The simplified-SIFT descriptor is calculated for those features that belong to the frame of the marker and pass the color test. Considering that each of these features has been matched to one of the four textures of the marker, only the corresponding database is used to get a correspondence. The query to the database returns the two closest descriptors (d_1 and d_2, respectively) according to the input descriptor (d_i). FLANN library gets an efficient approximation of the nearest neighbour, while the euclidean distance is used to determine the proximity between two descriptors. Therefore, d_1 and d_i are positively matched as long as the distance between d_1 and d_i is considerable smaller than the distance between d_2 and d_i ($k * \|d_1 - d_i\|^2 < \|d_2 - d_i\|^2$, where $k \in [0, 1]$ is constant set to 0.6 for this thesis).

As a result of matching current detected features with those stored in the database a set of 2D-3D matches are obtained. Nonetheless, some of these matches may be outliers, so the robust PROSAC method (Chum and Matas, 2005) is used to compute the camera pose. This is a hypothesize-and-verify method similar to RANSAC, with the difference that the most probable matches have higher probability to be selected. Here, the probability of a match is considered inversely proportional to the distance between its corresponding descriptors.

An ad-hoc PROSAC has also been proposed to exploit some properties of the new design of the marker. It is based on a new sample selection criteria that favours the quality of the homography that represents the transformation between the two sets of points (2D-3D). First, two textures of the marker that have, at least, two correspondences are randomly chosen, and then, two matches for each of those two textures are selected according to their matching quality. This hierarchical criteria overcomes the restriction of the homography generation, which states that not all the points can be collinear. In addition, it returns more stable homography transformations due to the optimal spatial distribution of selected points.

Finally, the maximum number of matches has also been limited to 75 with the aim of running the method in real time. This way, the matches with the highest quality will be selected first.

3.3.5 New Interface Possibilities

The new marker design that is proposed provides more information when the marker is occluded, which can be used to develop new interfaces. These new interface possibilities are complementary to those interfaces created by other authors, who have also used occlusions to implement them. For example, (Lee et al., 2004) places multiple markers at specified locations and depending on which markers are occluded different user actions are interpreted. In a similar way, (Canada et al., 2003) presents a simple hand gesture recognition. It is noteworthy that all these interfaces can be easily adapted to Occlusion-patches, while the novel interfaces that are shown here are more difficult to obtain with other designs.

All the interfaces presented here have to determine which features of the marker are visible. This is a simple task when the marker is not completely inside the field of view of the camera because the projection of the occluded points fall outside the image. However, this is not the case for occlusions caused by other objects such as the user's hands. For these cases, only those features that have a point with similar GREY descriptor in its vicinity are considered as visible. Note that this information is already generated by the frame-to-frame tracking (Section 3.3.4.1), so it has a negligible computational overhead. Two novel interfaces are presented below to demonstrate these abilities, but it can be extrapolated to numerous applications.

Occlusion Signal

The placement of textures along the frame of the marker implies the detection of multiple features. Using this assumption the degree of occlusion can be measured, which is proportional to the number of occluded points. Indeed, an occlusion signal can be displayed to inform the user about the degree of occlusion (Figure 3.20). This is similar to the idea of a coverage signal used by mobile devices.

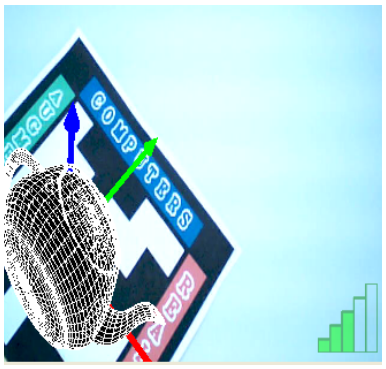


Figure 3.20: An example of Occlusion Signal output.

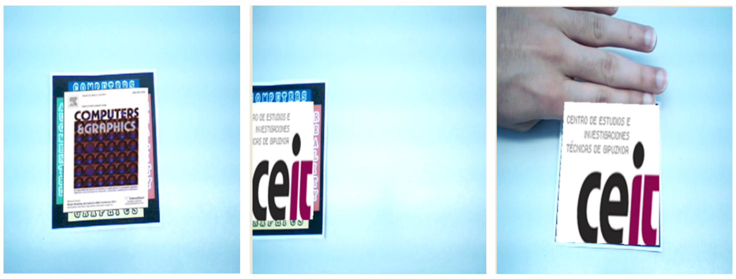
Photo viewer

Due to the presence of textures multiple features are extracted in each border of the marker. Thus, the border with fewer visible features can be identified, which corresponds to the occluded side of the marker. Moreover, the occlusion of each border can be interpreted as a different action. To prove this usability a photo viewer has been implemented (Figure 3.21), for which a different action is executed depending on which border is occluded:

- left-border: go to the previous photo.
- right-border: go to the next photo.
- up-border: apply a positive zoom to the current photo.
- down-border: apply a negative zoom to the current photo.

3.3.6 Experiments and Results

In this Section several characteristics of the new marker design are investigated. The hardware setup consists of an Intel Core 2-Duo at 2.40GHz and 2GB of RAM equipped with a Logitech QuickCam Connect webcam. All the experiments have been executed with a Windows 7 operating system.



(a) Initial state. (b) Previous photo. (c) Positive zoom.

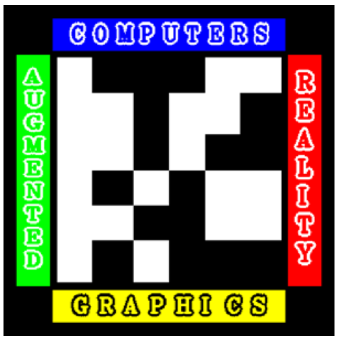
Figure 3.21: Photo Viewer sequence.

3.3.6.1 Tracking by Detection Parameterisation

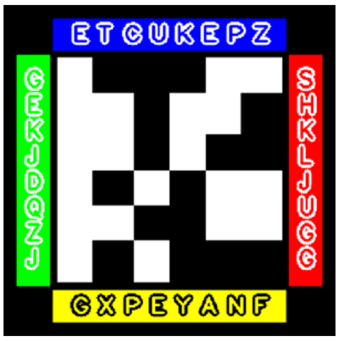
First, the response of the tracking by detection method has been studied. Multiple marker designs have been used (Figure 3.22) to investigate the importance of choosing a good texture content. More precisely, four different marker designs have been developed to simulate multiple conditions:

- **Words** (Figure 3.22(a)) represents a typical design that can be used for marketing purposes by encoding some words of a message along the frame of the marker.
- **Random-characters** (Figure 3.22(b)) is similar to Words, but using randomly selected characters to demonstrate that no specific letters are required.
- **Random-shapes** (Figure 3.22(c)) symbolizes textures that are made using logos.
- **Repetitive-symbols** (Figure 3.22(d)) is referred to those shapes that have a repetitive pattern and do not provide distinctiveness.

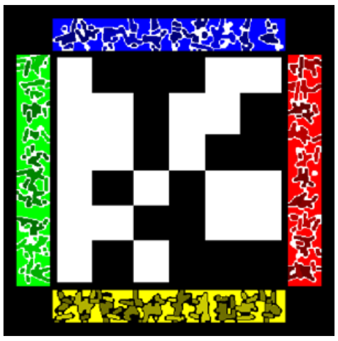
To obtain the results that are presented below a set of images with a known camera pose are required. According to this, a set of snapshots was acquired, in which the marker was completely visible. These images were processed with the normal ARToolkitPlus pipeline, obtaining the corresponding camera pose for each one. Moreover, these poses were used



(a) Words.



(b) Random-characters.


(c) Random-shapes.

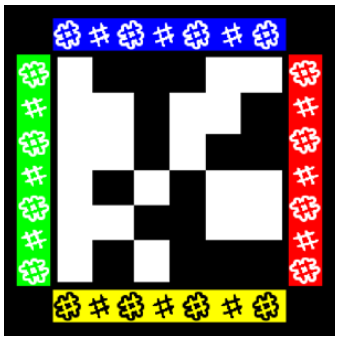

(d) Repetitive-symbols.

Figure 3.22: Marker designs for experiments.

to draw black patches on the surface of the marker, simulating marker occlusions (Figure 3.23).

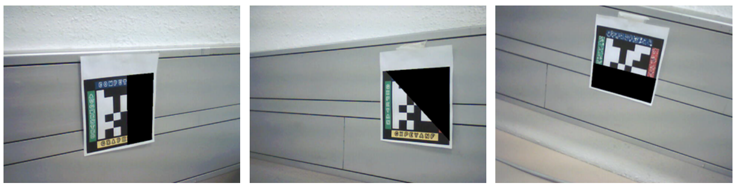


Figure 3.23: Simulation of the partial occlusion of the marker.

The scale is the first parameter that has been examined. Nevertheless, this parameter is not dependent on the marker design, but the size of the textures. It is noteworthy that the size of the textures is very small compared to the size of the whole marker, as the total increase in the size of the marker has been minimized (Section 3.3.1). This, however, increases the difficulty of their detection, since textures appear too small in the image, without sharpness. Figure 3.24 shows the average success ratio of tracking by detection for the four marker designs of Figure 3.22. It is displayed according to the width of the marker and the resolution of the image.

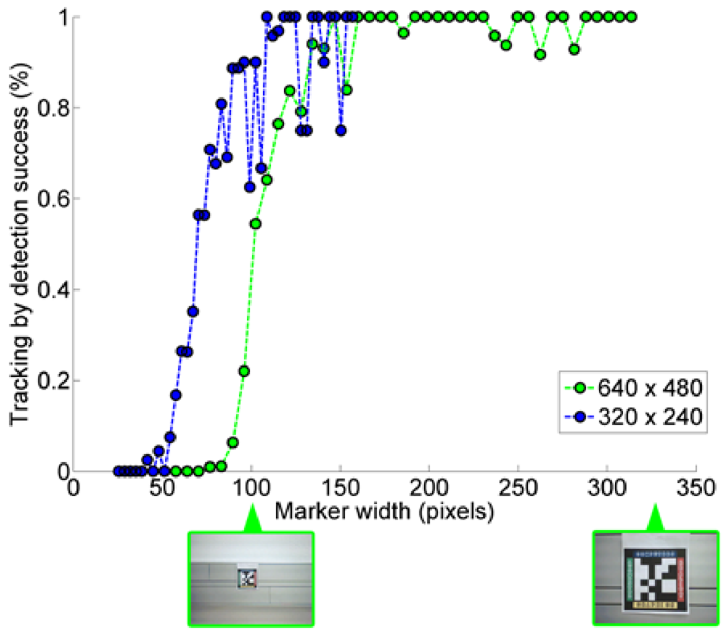


Figure 3.24: Scale study for tracking by detection.

This ratio is increased for larger marker sizes because textures appear clearly and more accurate simplified-SIFT descriptors are calculated. At 320x240 resolution a good detection ratio is obtained when the marker width is larger than 75 pixels (22,44% of the image width), while at 640x480 resolution a similar ratio is obtained for a width of 125 pixels (19,53% of the image width). This difference can be explained by the larger amount of blur that 320x240 images have.

The detection is considered successful for an input image when the reprojection error of the marker bounding box using the real pose and that returned by tracking by detection is below a threshold (5 pixels, for example). In addition, should be point out that tracking by detection has not had false positives for these experiments; i.e. the presence of the marker has not been detected unless it was actually there. This is a desirable property, as this is the starting point for the frame-to-frame tracking method, which is based on temporal coherence.

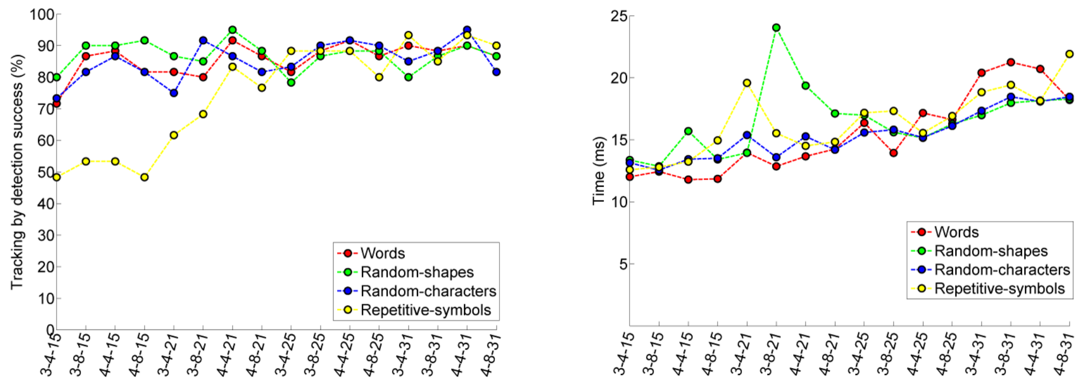
Simplified-SIFT has also been analysed. A set of snapshots have been taken for each marker design of Figure 3.22 using the scale ranges in which the tracking by detection is executed successfully (Figure 3.24). These snapshots have been used to perform several executions of the same image set, but with different simplified-SIFT configurations. The $i - j - k$ parameterisation of simplified-SIFT corresponds to the number of regions (i), the number of histogram bins (j), and the patch size (k) that is used to build the simplified-SIFT descriptor (Appendix A). Figure 3.25 shows the success ratio of tracking by detection for each marker design, according to the simplified-SIFT parameterisation and for different image resolutions.

The ratio of success increases for larger simplified-SIFT patches because more stable descriptors are built. Likewise, the computational cost increases proportionally with the size of the patch, so a compromise between robustness and computational cost must be reached. Furthermore, the image resolution also influences this decision, as low resolutions require smaller patch sizes to get those results achieved by high resolutions.

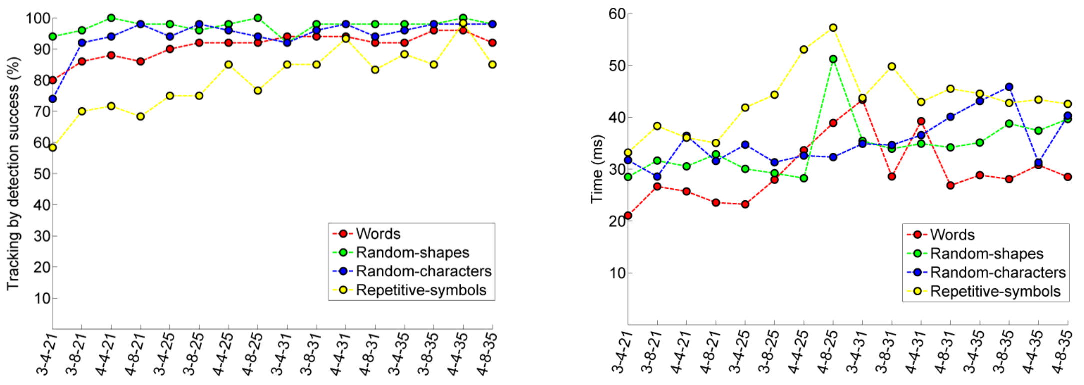
The other two parameters of simplified-SIFT (the number of regions and the number of histogram bins) are related to the size of the descriptor. They offer the possibility of establishing the degree of descriptor-distinctiveness that best suits the properties of the corresponding textures. According to Figure 3.25 the best results are obtained when the patch is divided into $4 * 4$ regions and the number of histogram bins that codify each region is set between 4 and 8.

Although a different set of images has been used for each marker design, which could explain small differences in the success ratio between two different marker designs, the results of Figure 3.25 for 640x480 resolution establish that textures with a repetitive pattern (Repetitive-symbols) harm the descriptor-distinctiveness. This implies that multiple false matches appear during the detection, which increases the computational time that PROSAC takes to discard these outliers (Section 3.3.4.2).

The good response of Random-characters and Random-shapes designs demonstrates that textures with words and shapes are valid solutions, since



(a) 320x240 resolution.

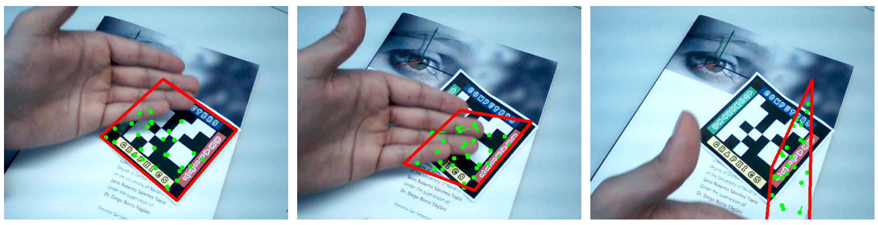


(b) 640x480 resolution.

Figure 3.25: The tracking by detection success (left) and computational time (right) according to the simplified-SIFT parameterisation and image resolution.

they build good quality descriptors as a result of generating a lot of unique patches around each 3D point. The response of Words proves that positive detection results are also obtained using textures with customized words as long as a *repetitive-pattern* does not appear.

Taking everything into consideration, 4-4-25 and 4-4-31 simplified-SIFT parameterisations are the configurations that generally provide the best compromise between ratio of success and computational cost for 320x240 and 640x480 resolutions respectively.


(a) Without refinement step.

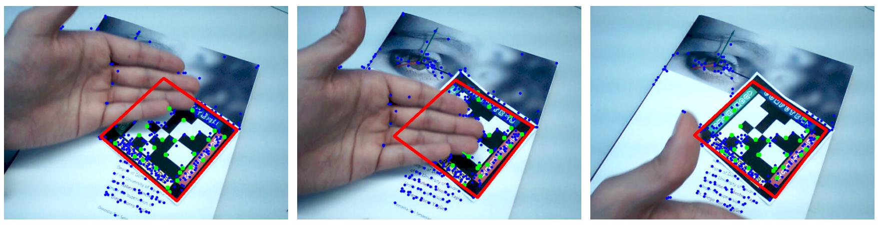

(b) With refinement step.

Figure 3.26: Camera pose (red) using the frame-to-frame tracking.

3.3.6.2 Frame-To-Frame Tracking Robustness

Only the Words design (Figure 3.22(a)) has been used to validate the robustness of the frame-to-frame tracking, as the results are similar for other designs. The same video sequence has been executed twice, one without the refinement step presented in Section 3.3.4.1 and the other one using the refinement process. In the video the marker is occluded by the user's hand movement and lies on a textured surface to increase the difficulty of tracking. In addition, although the video is at 640x480 resolution the frame-to-frame tracking is executed at 320x240 resolution to reduce the computational cost.

As shown in Figure 3.26, this experiment demonstrates the robustness of the refinement step, which returns a good pose despite the hand occlusion because poorly tracked features are corrected using the correlation of GREY descriptors (Section 3.3.4.1). The absence of the refinement step increases the tracking error due to the poor prediction of the optical flow, which is confused by the hand movement. This last option is the technique used by other authors (Malik et al., 2002; Wagner et al., 2008), so the frame-to-frame tracking proposed here offers greater robustness compared to similar solutions.

The computational time for the two executions was 8.1 and 13.05 milliseconds respectively. The time difference is due to the cost of computing and matching the GREY descriptors that are necessary for the refinement step. Should be noted that this is the worst scenario for the frame-to-frame tracking, as there are a lot of visible features that must be tracked (none of the control points of the marker fall outside the image). Additionally, there are a lot of corners around the marker (blue points of Figure 3.26(b)), whose GREY descriptors must be computed and which can produce false matches. Despite all these difficulties the frame-to-frame tracking runs in real time and computes a good camera pose.

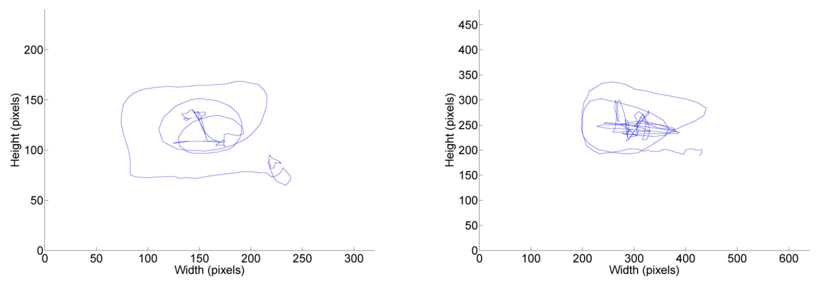
3.3.6.3 Occlusion-patches Robustness

Figure 3.27 shows the response of Occlusion-patches for a video sequence in which both tracking methods are mixed. The marker was visible during the entire sequence in order to successfully execute ARToolkitPlus and provide the correct pose of each frame of the video. Marker occlusions were simulated using the procedure described in Section 3.3.6.1.

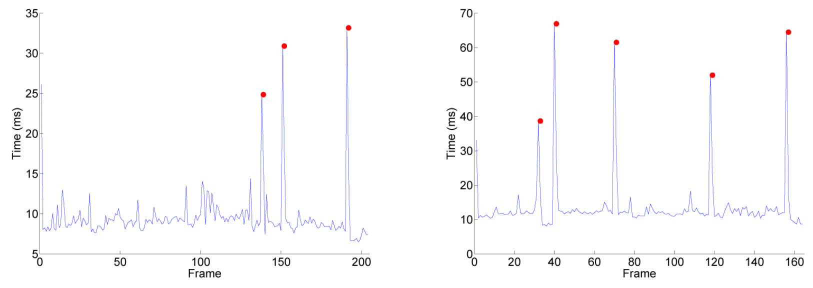
The image position of the center of the marker along the entire sequence is shown in Figure 3.27(a) to indicate that the marker is continuously moving. Even fast movements that impede tracking can also be appreciated. Red dots of Figure 3.27(b) represent the execution of the tracking by detection. The computational time is the sum of both tracking techniques.

The frame-to-frame tracking is executed as long as possible due to its low computational time (Figure 3.27(b)). When this method fails as a consequence of a fast movement or error accumulation the tracking by detection is called. This method computes the correct camera pose and restarts the frame-to-frame tracking. This way, the reprojection error between the correct camera pose and that returned by Occlusion-patches (Figure 3.27(c)) is maintained at reasonable levels most of the time. The few frames that have higher reprojection error are those frames that the frame-to-frame tracking takes to indicate failure. Even so, the time it takes to react is very small and the results are still visually acceptable.

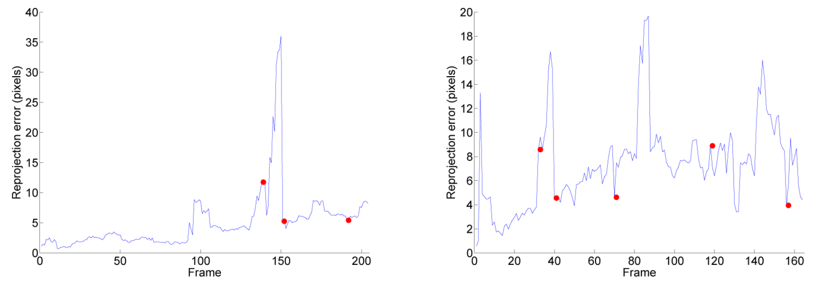
Figure 3.28 displays the output of Occlusion-patches for different types of marker occlusions. A virtual teapot as well as the marker bounding box are rendered to indicate tracking success. Figure 3.28(a) demonstrates that Occlusion-patches supports hand occlusions, including recovery from a



(a) Marker motion for 320x240 (left) and 640x480 (right).



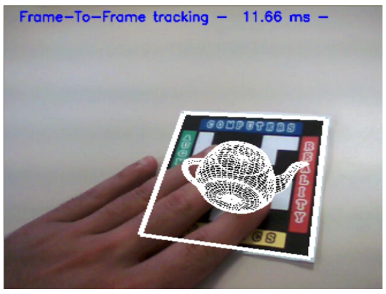
(b) Computational time for 320x240 (left) and 640x480 (right).



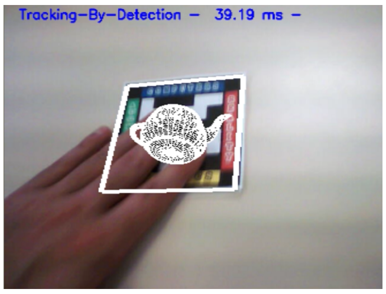
(c) Reprojection error for 320x240 (left) and 640x480 (right).

Figure 3.27: Occlusion-patches response for a video sequence.

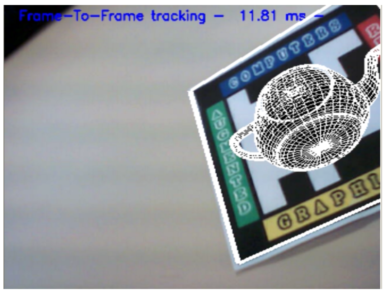
fast camera movement (Figure 3.28(b)). Figure 3.28(c) shows the ability to compute the camera pose when the marker is partially outside the image. In addition, Figure 3.28(d) represents the tracking success when the marker is occluded by multiple objects. Likewise, Figure 3.28(e) and Figure 3.28(f) display the robustness against occlusions made by objects with similar colour.



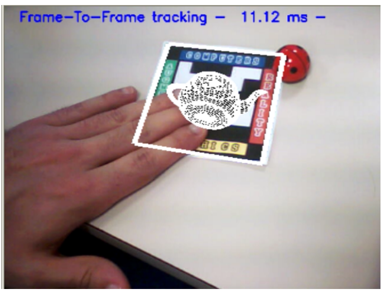
(a) Hand.



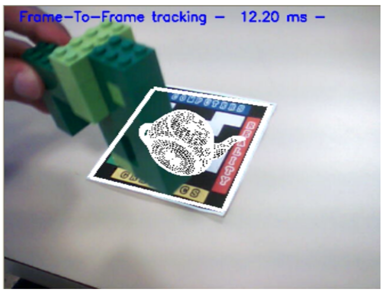
(b) Hand (tracking by detection).



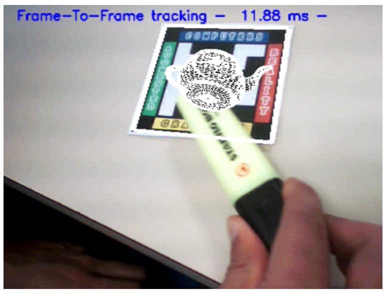
(c) Image border.



(d) Multiple objects.



(e) Coloured object.



(f) Coloured object.

Figure 3.28: Occlusion-patches output for different occlusions.

3.4 Discussion

As presented in Chapter 1, marker tracking systems are a widespread alternative for the development of AR applications, including industrial environments. Moreover, as stated by (Cawood and Fiala, 2008), there are many criteria/metrics that must be considered when evaluating the quality of a marker tracking system, among which appears occlusion immunity. Despite these observations a marker tracking system usually fails when the marker is not completely visible in the image.

Marker occlusion may be caused by a sudden camera movement, which puts the marker partially outside the image. This forces to put the marker completely visible in the image again to recover the camera pose. However, this is not always an easy task, as sometimes people are limited by the small size of their work space. In other cases, this type of occlusions are facilitated by the great freedom of movement offered by handheld devices. This is the case of a worker who uses a mobile device equipped with a camera to receive augmented instructions that help in maintenance operations. There are other type of occlusions caused by the presence of objects between the camera and the marker, which are also a likely scenario in maintenance operations since occlusions with tools and the hands of the worker are common.

Knowing the importance of occlusions in industrial environments, in this chapter two different approaches (Occlusion-OBB and Occlusion-patches) have been proposed to address the problem of marker occlusion. Both use computer vision techniques to update the camera pose in real time when the marker is partially occluded. They discard the option of adding more environment adaptation (multiple markers), as they work with a single marker.

Occlusion-OBB uses simple image processing techniques and temporal coherence assumptions. It only updates 4 DOF of the camera pose, but has a very low computational cost. It was designed having in mind the limited computing capabilities of mobile devices, which were the target market. It has been implemented assuming square markers, so it is limited to those type of patterns. Indeed, it would be difficult to compute the rotation of circular markers using these ideas.

Although Occlusion-patches enhances the robustness and is oriented to standard PCs, it is noteworthy that similar techniques have already been implemented for mobile platforms (Wagner et al., 2010), which suggests that it could be adapted to these devices. Occlusion-patches is based on a new marker design, which places textures in the frame of the marker. These textures are customizable by the users, which is a desirable property to offer high adaptability and for marketing purposes. This approach combines two different tracking methods to obtain more robustness, which has been demonstrated through a set of experiments. The first one is a fast method based on temporal coherence, which works fine while no fast camera movements take place, while the second one is a tracking method based on appearance, which supports rapid camera movements. Additionally, two novel human-machine interfaces have been presented to show the new possibilities that have been arisen as a result of obtaining more information during marker occlusions. In contrast to Occlusion-OBB, although this approach has been tested with square markers, it can be applied to markers with any shape. Textures should be considered as a set of tags that can be attached to any surface.

Besides other advantages, this chapter covers broadly the problem of marker occlusion by offering a solution for different computation capabilities and for different requirements (4 DOF vs 6 DOF).

Chapter 4

3D Object Recognition

You have to have the fighting spirit.
You have to force moves and take chances
BOBBY FISCHER

This section describes a monocular 3D object recognition oriented to untextured 3D models, which are very common in industrial environments. Due to the lack of texture, this recognition method uses geometric features of the model (junctions and edges) to recover the camera pose, without knowledge about previous frames. Indeed, recognition methods are used to initialize markerless tracking systems as well as failure recovery because of their non-recursive nature.

A synthesis of this chapter has been submitted to:

> *Álvarez, H. and Borro, D. "Junction assisted 3d pose retrieval of untextured 3d models in monocular images". Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011.*

4.1 Introduction

The problem of 3D object recognition tries to recover the six camera parameters that define the viewpoint from which the model is shown in the image, which is hampered by the lack of knowledge about previous frames. Although the recognition of a 3D model in monocular images has been investigated in many computer vision areas, its greatest popularity has been achieved in industrial environments. It is used for multiple tasks, such as automatic inspection of model properties (von Bank et al., 2003; Schoenfelder and Schmalstieg, 2008) or pick and place operations. Moreover, most of the models related to this environment share a common property: the lack of texture on their surface.

This chapter addresses the challenge of recognising an untextured 3D model from a monocular image, in the shortest amount of time and minimizing the user interaction. This is a difficult task since robust and fast techniques based on texture (Lowe, 2001; Lowe, 2004; Rothganger et al., 2006; Bay et al., 2008) cannot be used. Therefore, the proposed approach is based on the geometric properties of the model such as edges and junctions (points where several edges meet).

In the following sections previous works on 3D object recognition will be presented, and then, the proposed approach will be detailed. Some experiments and results will be also included, which highlight the good ratio that the proposed approach obtains between performance and robustness.

4.2 Previous Works

3D model recognition in monocular image is a well-known problem that has been studied by several authors, which offer different ways of proceeding according to the model and environment properties.

(Lowe, 2001; Lowe, 2004; Rothganger et al., 2006; Bay et al., 2008) use feature descriptors to determine a set of 2D-3D matches that define the 3D pose of the model. Some 2D views of the model are selected as representative during the training step. These views are processed, and features that lie on the surface of the model are extracted. For each feature its 3D position and descriptor is calculated and used to train a classifier that will be the responsible of the matching during the online phase. Furthermore, descriptor-based techniques combined with geometric constraints improve precision and efficiency (Hinterstoisser et al., 2007). All of them achieve good results and high frame rates, but the robustness is decreased for untextured objects. They are oriented to rich textured objects, since descriptors that can be extracted from their surface are more discriminative, thereby improving the matching results. Generally, these methods also require user intervention to get the training dataset.

Other approaches use the contours of the model to get a positive match

(Selinger and Nelson, 1999; Ulrich et al., 2003; Opelt et al., 2006; Ferrari et al., 2008; Holzer et al., 2009; Hinterstoisser et al., 2010). They are based on the shape of the model, and therefore, they are suitable for untextured models. In the training step 2D views of the model are processed in order to extract the corresponding edges and transform the online problem to a 2D-2D edge pattern matching. Although some of the works cited only address the problem of 2D-2D edge matching (Ulrich et al., 2003; Opelt et al., 2006; Ferrari et al., 2008), they can be extrapolated to 3D recognition using a similar training step to that mentioned above. All of them need the acquisition of real 2D views, so they are configured for specific light conditions and edge responses extracted from the real training images. This is not the case of (Stark et al., 2010; Ulrich et al., 2009), which use a virtual training to build 2D artificial views of the object, reaching more generality.

The third group of alternatives are based on geometric features (Lamdan and Wolfson, 1988; Costa and Shapiro, 2000; Shahrokni et al., 2002; Sehgal, 2003; Kotake et al., 2007). Distinctive 3D geometric features of the model are extracted in the offline phase, and their geometric relationships are indexed in a data base. During the online phase image input features are detected and compared to the database. Consistent matches receive a vote and the solutions with the highest number of votes are selected for further processing. Generally, these features provide poor distinctiveness, so an input feature generates multiple matches, which increases the computational effort. Due to their computational requirements they are more popular for no time-critical applications such as 3D pose recovery in 3D scenes (Drost et al., 2010). They have also been optimised using the correspondences specified by the user (Franken et al., 2005), but this has the disadvantage of requiring user intervention.

4.3 Proposed Method

The proposed approach is related to geometric and contour based solutions. In the offline phase, it builds a global hash table using geometric relationships between the junctions of the model. These relationships are extracted from a set of 2D synthetic model views, which are created automatically, without user intervention. In addition, a sophisticated geometric constraint between two junctions is proposed, which allows to reduce false matches, and consequently, the computational time. This is the main difference with geometric alternatives, which waste too much time processing many unnecessary matches. In the online phase, rather than using a voting scheme, matches with a similar position and scale are clustered. Each cluster is considered as a valid hypothesis and is evaluated using a similarity measure based on contours. This improves robustness, as hypotheses are not thresholded according to the number of votes. In summary, optimised geometric constraints are used to generate pose candidates, which are evaluated using contours.

(Ulrich et al., 2009) is the work that comes closest to the 3D model recognition described in this chapter. Both studies create artificial 2D views of the model in the training step, use a robust contour based similarity measure, and both can recognise untextured 3D models in a reasonable amount of time. Compared to (Ulrich et al., 2009), the proposed method offers less memory requirements and it is not limited to a predefined scale of the model.

The use of junctions for object recognition is not a new idea. They have demonstrated their viability for 2D object recognition (Wang et al., 2010). However, an expensive edge detection algorithm and complex descriptors are required to get successful results, which takes several minutes. The solution presented here, meanwhile, offers an efficient and robust mechanism to handle junctions and recognize 3D objects in a few hundred of milliseconds, which is fast enough for some time-critical applications.

4.3.1 Algorithm Overview

The proposed algorithm overview is shown in Figure 4.1. It is divided into two main blocks. In the first one (*Offline*) geometric features of the model are extracted and used to build a collection of virtual keyframes. The second block (*Online*), meanwhile, processes the camera image to find positive correspondences between the camera image features and the virtual keyframe features, as these positive correspondences will establish the presence of the object in the current view. An in depth explanation of each block is given in the following paragraphs.

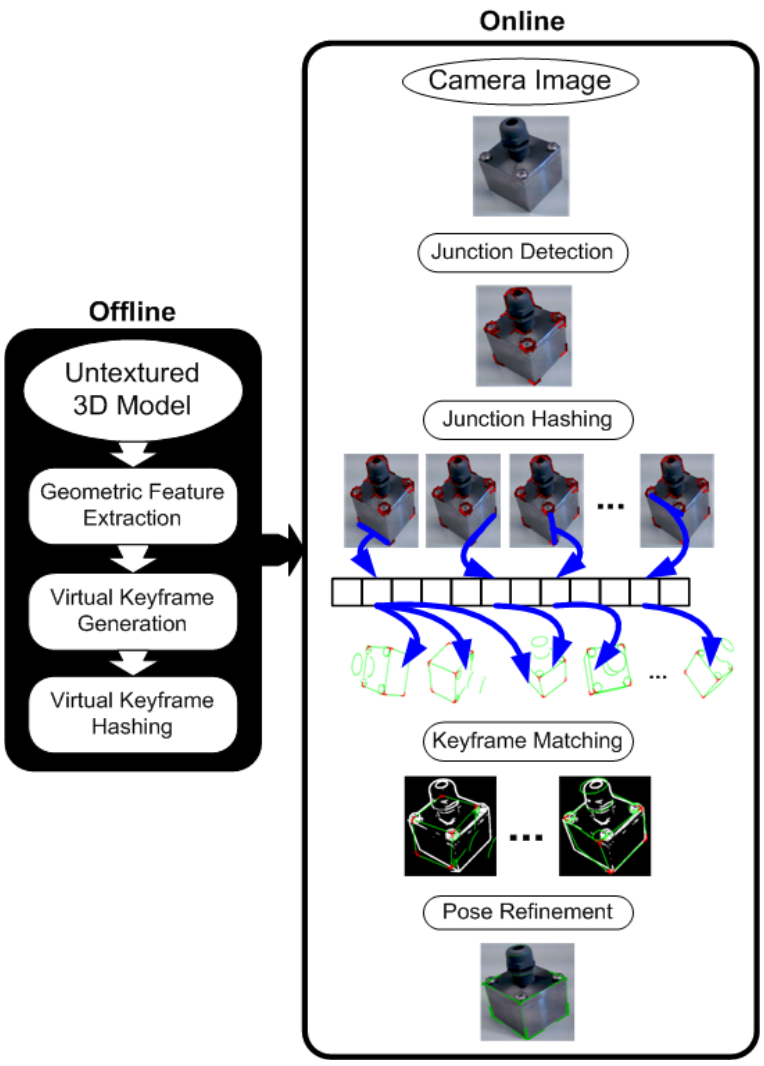


Figure 4.1: 3D object recognition overview.

4.3.2 Offline

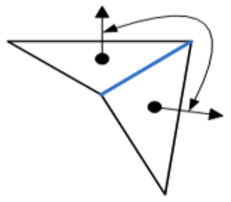
This stage extracts automatically all the information that the model provides and overrides completely the user interaction. This aim is hampered by the lack of texture in the model, which precludes any well known method of the literature (Lowe, 2001; Lowe, 2004; Rothganger et al., 2006; Bay et al., 2008). Thus, although the model only retains information about its geometry (triangle mesh), it will be enough to perform an automatic object detection, as it is shown in this chapter. This stage is executed only once per model.

4.3.2.1 Geometric Feature Extraction

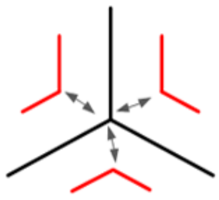
The detection of geometric features (such as edges) from a triangular mesh is not a new problem (Jiao and Heath, 2002; Yamakawa and Shimada, 2005; Platonov and Langer, 2007). Most techniques are based on the detection of sharp changes between the normals of neighbour triangles, besides some local constraints to improve results (Jiao and Heath, 2002; Yamakawa and Shimada, 2005). Other authors, however, render the model from different points of view and backproject detected 2D edges to 3D space by taking into account the texture of the model (Platonov and Langer, 2007).

A simplified version of (Jiao and Heath, 2002; Yamakawa and Shimada, 2005) has been implemented for this thesis. Therefore, a 3D sharp edge is detected if the angle between its neighbour facets is larger than a predefined threshold (30-40 degrees for example), without considering local constraints (Figure 4.2(a)). Additionally, 3D junctions of the model are detected, which is a problem that has not been previously addressed by other authors. If two 3D sharp edges have one coincident vertex and the angle that they form satisfies a predefined minimum and maximum thresholds (good results have been obtained with 80 and 100 degrees respectively), then both of them build a 3D junction. This way, the extracted 3D junctions are L junctions. In fact, the vertex where more than 2 sharp edges meet are split into several L junctions (Figure 4.2(b)). These definitions are shown in Figure 4.2, while real examples are presented in Figure 4.3.

This technique also considers sharp edges and junctions that belong to the inner boundary of a model (Figure 4.4). These types of edges and junctions are always hidden by other parts of the model and do not provide relevant information. Similarly, undesired sharp edges and junctions can be detected due to a bad tessellation. Following this reasoning, a refinement step is applied to discard these outlier edges and junctions. The model is rendered from different points of view using the OpenGL pipeline and following a sphere path. Good results have been obtained with steps of altitude and azimuth of 30 degrees, which gives a total of 144 views. For each



(a) 3D sharp edge.



(b) L junctions.

Figure 4.2: Definition of 3D sharp edges and 3D L junctions.

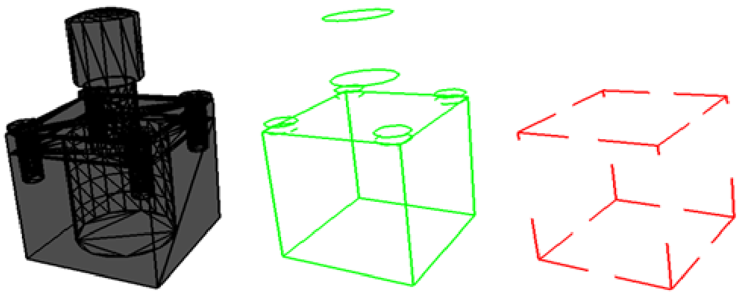
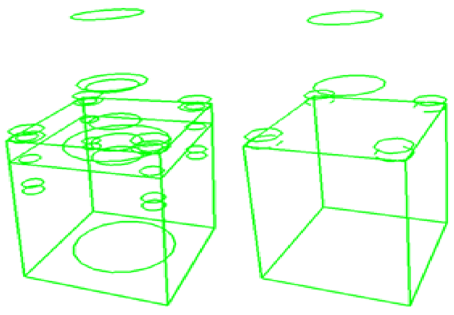


Figure 4.3: Geometric Feature Extraction for a 3D box. Facets (left), 3D sharp edges (middle) and 3D L junctions (right).

view a 2D edge map is generated applying the Sobel operator to the depth buffer image of OpenGL, which is filled rendering the 3D triangle mesh, similar to (Wuest et al., 2007). This 2D edge map stores the projection of strong edges and junctions that are visible from the current point of view. On the other hand, visible 3D edges and junctions are determined for the current view from the collection of 3D sharp edges and junctions using the OpenGL occlusion query. Thus, in case that a 3D sharp edge is visible and its projection appears in the 2D edge map, it receives a vote. Notice that this vote scheme is analogous for junctions and repeated for all views. Finally, only the 3D sharp edges and junctions with a minimum number of votes are retained as inliers, as they are strong 3D edges and junctions that belong to the outer boundary.

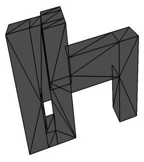
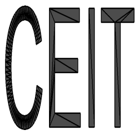
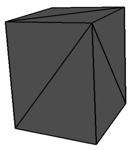
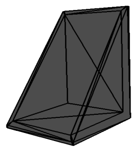
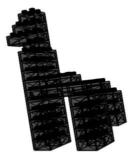
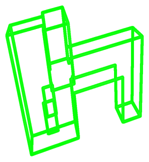
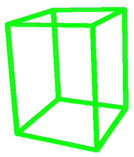
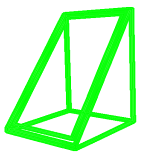
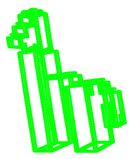
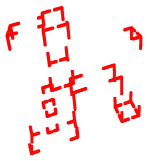
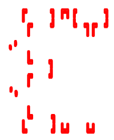
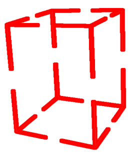
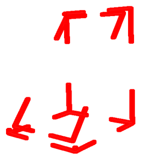
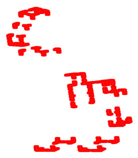
This refinement process improves the quality of the geometric feature extraction as well as guarantees the robustness of the recognition. Table 4.1 shows resulting geometric feature extraction for several 3D models.



(a) All edges. (b) Outer edges.

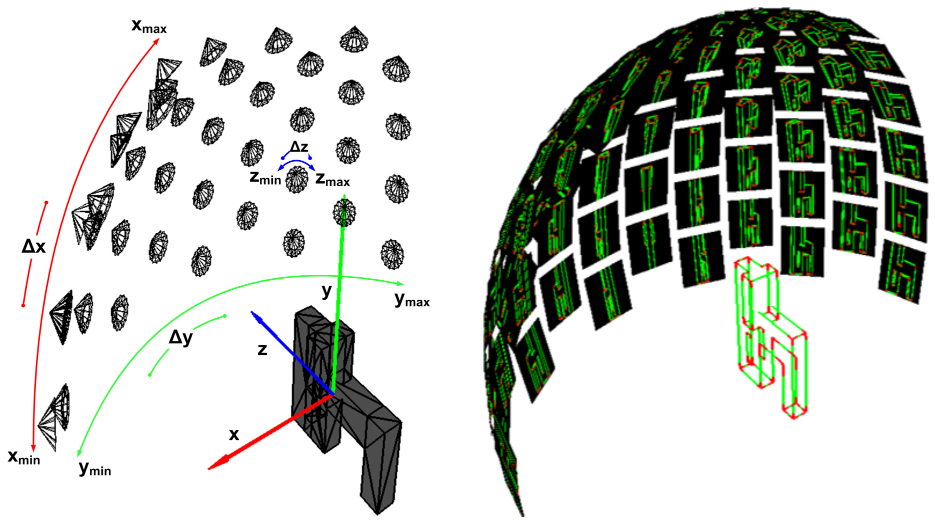
Figure 4.4: 3D sharp edges before and after removing inner edges.

Table 4.1: Geometric Feature Extraction examples.

	Elephant	3D text	Box	Tri. bracket	Giraffe
Facets	104	158	12	38	53040
					
Edges	57	42	12	23	148
					
Junctions	76	28	24	30	124
					

4.3.2.2 Virtual Keyframe Generation

As a result of the previous step 3D sharp edges and 3D junctions are obtained for an input 3D model, but this 3D information cannot be matched directly to the camera 2D image. A training process is required to create several 2D views of these 3D features and simplify the problem to a 2D-2D matching. Because of that, some 2D synthetic views of the model (virtual keyframes) are created moving a virtual camera (that is pointing to the model center) along a sphere. This procedure is illustrated in Figure 4.5.



(a) Virtual cameras. (b) Virtual keyframes.

Figure 4.5: Virtual keyframe generation.

This is an automatic process parameterised by the rotation ranges along each axis ($[x_{min}, x_{max}]$, $[y_{min}, y_{max}]$ and $[z_{min}, z_{max}]$) and their corresponding rotation steps (Δx, Δy and Δz) (see Section 4.4 for an in depth discussion). In contrast to other similar approaches (Ulrich et al., 2009), space partition of the model scale is not required because the proposed method computes the scale during the execution (Section 4.3.3.3), i.e., it is not limited to a set of scales fixed in the offline phase.

For each virtual keyframe visible 3D sharp edges and 3D junctions are determined using an OpenGL occlusion query. They are also projected to the image plane of the virtual camera, and therefore, each virtual keyframe stores a list of 2D edges and a list of 2D junctions as well as their corresponding 3D values. Furthermore, for each 2D junction two angles are stored: (1) the minimum angle between its two branches (α) and (2) the minimum angle in the clockwise direction that one of its two branches form with the x axis (β) (see Figure 4.6).

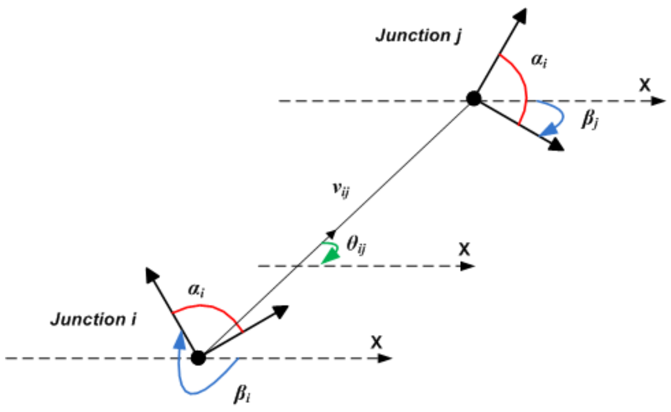


Figure 4.6: Junction basis definition.

Notice that a 2D junction that is considered separately does not provide much distinctiveness for untextured models since it does not lie on a textured surface. It only offers a simple geometric constraint based on the angle of its branches. Due to this reasoning, a *junction basis* descriptor is used, which stores the relative orientation of two 2D junctions (Figure 4.6) and offers a more sophisticated geometric constraint. It is similar to the *point pair feature* described in (Drost et al., 2010), but oriented to junctions. A junction basis of i and j 2D junctions is defined as:

$$junction\_basis(i,j) = (\alpha_i, \beta_i, \alpha_j, \beta_j, \theta_{ij}, d_{ij}), \tag{4.1}$$

where θ_{ij} is the angle between the v_{ij} vector and x axis, $d_{ij} = \|v_{ij}\|_2$, and v_{ij} vector is defined by the junction i and junction j centers. It follows that a junction basis is asymmetric since $\theta_{ij} \neq \theta_{ji}$.

Degenerated results arise when the distance between the junction centers is close to 0 ($d_{ij} \approx 0$) (see Section 4.3.3.3), so they are discarded. In summary, each virtual keyframe is composed of a list of 2D visible edges plus their 3D values and a junction basis set. In fact, for each virtual keyframe with n junctions a maximum of $n!/(n-2)!$ junction bases are stored.

4.3.2.3 Virtual Keyframe Hashing

After processing all virtual keyframes, all junction bases will be grouped in a global hash table, similar to (Lamdan and Wolfson, 1988; Drost et al., 2010). This way, an input junction basis is associated with all virtual keyframes that have a junction basis with similar description, which provides an efficient mechanism to match junction bases.

The key of the hash table is extracted from β_i, β_j and θ_{ij} of each junction basis. Other parameters are discarded for key generation to simplify the procedure and reduce key dimensionality. The key is thereby computed applying a proper quantisation step to these 3 angles ($\Delta\beta_i$, $\Delta\beta_j$ and $\Delta\theta_{ij}$, respectively), and each record of the hash table stores a list of pairs *(keyframe_id, junction_basis_id)*.

This is an efficient scheme for matching an input junction basis in constant time. However, the partition of β_i, β_j and θ_{ij} angles in several intervals introduces errors at their boundaries. Due to the impreciseness of the junction detector, besides distortions caused by the projective transformations when processing the input camera image, some angles may cross the boundary to the next angle interval. This would jeopardise the matching quality by promoting unstable results. That is why the same criteria adopted by (Ulrich et al., 2003) is applied, considering overlapping intervals. Hence, in the global hash table a junction basis will be stored in the corresponding position and in neighbour records. More precisely, a junction basis with β_i, β_j and θ_{ij} angles will be indexed using the ranges

$$\begin{gathered}[\beta_i - \Delta\beta_i * \xi_i,\ \beta_i + \Delta\beta_i * \xi_i],\\ [\beta_j - \Delta\beta_j * \xi_j,\ \beta_j + \Delta\beta_j * \xi_j],\\ [\theta_{ij} - \Delta\theta_{ij} * \xi_{ij},\ \theta_{ij} + \Delta\theta_{ij} * \xi_{ij}],\end{gathered} \tag{4.2}$$

where ξ_i, ξ_j, $\xi_{ij} \in [0, 1]$ are the ratio of overlap for intervals. See Section 4.4 for reasonable values of these parameters.

4.3.3 Online

Once an input 3D model has been processed in the offline-phase, its geometry description is represented by a set of junction bases that are scattered along a set of virtual keyframes. Therefore, given an input image, the problem is to find the optimal junction basis correspondences that recover the 3D pose of the object. The necessary steps to achieve this goal are detailed in the following subsections.

4.3.3.1 Junction Detection

Before proceeding with the junction basis construction the presence of 2D juntions in the image must be determined. Junction detectors can be classified into two main groups: *region based* and *edge based* approaches (Cazorla and Escolano, 2003). The first group processes intensities inside a predefined circular region to segment them into homogeneous intervals. The edge based techniques, meanwhile, try to identify edges that emanate from the junction center so that the response of edge filters and the value of a statistical test determine the presence of junction branches.

Although there are several junction operators (Cazorla and Escolano, 2003; Parida et al., 1998), most of them are computationally demanding, making them unsuitable for real time applications. JUDOCA (JUnction Detection from Circumferential Anchor points) (Laganiere and Elias, 2004) is one of the most efficient junction detectors, which is public available[1]. It belongs to the edge based group, and has demonstrated a good relation between robustness and performance. Due to this property, this operator has been chosen for this dissertation. Notice, however, that all the framework that is described in this chapter is valid regardless of the selected junction detector. For the completeness of the thesis a brief explanation of how JUDOCA works is given here. An interest reader can refer to (Laganiere and Elias, 2004) for an in depth explanation.

JUDOCA considers a circle of radius p centered on each edge pixel of the image (Figure 4.7). Furthermore, each edge point in the circle perimeter is an anchor point that defines a putative radial line with the center pixel. If most of this putative radial line lies on an image edge, then it is considered as a junction branch. Finally, edge pixels with more than one branch

[1] *www.site.uottawa.ca/research/viva/projects/judoca*

emanating from them are defined as junctions. Indeed, continuing with the same criteria as Section 4.3.2.1, junctions with more than 2 branches are split into several L junctions, and α and β angles are extracted for each junction.

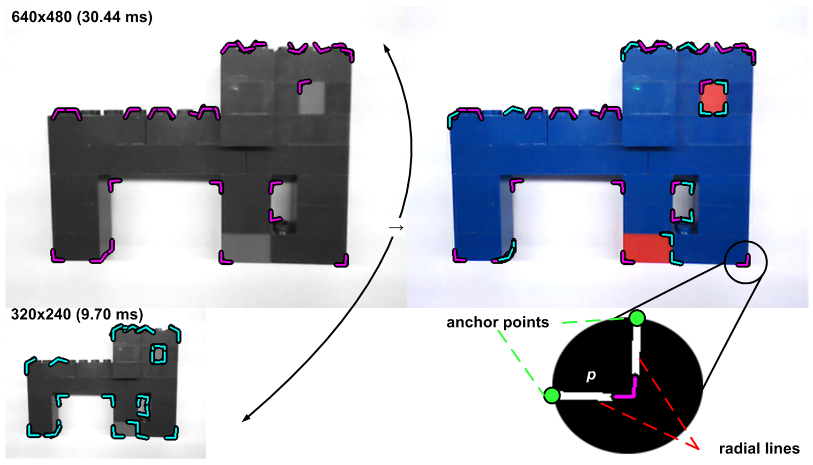


Figure 4.7: Junction extraction using pyramidal JUDOCA.

Some modifications have also been introduced to the original JUDOCA. First, a junction is suppressed only if there is another junction with similar and stronger branches in its vicinity. In contrast to the original method, branch orientations are taken into account for non-maxima suppression. This guarantees a better repeatability ratio at the expense of computational effort. Similarly, JUDOCA algorithm is executed at two different image resolutions to increase the detection ratio. Finally, due to the algorithm profile, operations associated with each circle are independent, so parallel techniques are also applied to decrease the computational time. As shown in Figure 4.7, the parallel execution of JUDOCA for a non-complex image takes less than 31 milliseconds at 640x480 resolution using an Intel Core i7-860 at 2.80GHz and 3GB of RAM.

4.3.3.2 Junction Hashing

After processing the input image, all pairs of extracted junctions that do not have coincident centers are considered as potential junction bases that lie on the model. These junction bases are built as explained in Section 4.3.2.2

and provide a set of votes for each keyframe. For each potential junction basis the hash table computed offline (Section 4.3.2.2) is accessed and the list of *(keyframe,junction basis)* pairs associated with it are obtained. Thus, an image junction basis is matched with multiple virtual junction bases scattered along the virtual keyframes. After repeating this process for all image junction bases, each virtual keyframe have a list of *(image junction basis, virtual junction basis)* matches, which represents the relationship between the virtual keyframe and the input camera image.

Since all pairs of image junctions are considered, the computational complexity increases exponentially with the number of image junctions. This is the case of cluttered scenes, where too many junctions are detected. Nonetheless, all image junction bases do not have the same importance because only a few of them belong to the model. Based on this assumption, it would be interesting to use a heuristic that selects the most probable junction basis in complex scenes while maintaining the computational cost at reasonable levels (see Section 4.4.1). Hence, the weight (w) of each image junction basis is based on:

$$w(junction\_basis_{ij}) = (1.0/d_{ij})^{p_i * p_j * N} \tag{4.3}$$

where d_{ij} is the distance between junction centers, p_i and p_j are the quality of junction i and j respectively, and N is a scale factor. Edges of the image are broken by curvature points, and the quality of each junction is proportional to the size of the edge portion it belongs to. This tries to penalise junctions extracted from short false edges due to noise, i.e., junction bases whose junctions are extracted from long straight edges and which are not too distant from each other are favoured. Since this heuristic is only applied in cluttered scenes, where the model is surrounded by other objects, setting more weight to close junctions sounds reasonable because distant junctions probably belong to two different objects.

Taking everything into consideration, all image junction bases are sorted in descending order using Equation 4.3 and only the best k junction bases are considered for the hashing process described above. Note that the pose of the model is not selected according to the number of matched junction bases, but all hypotheses generated by all matches are explored, so few matches of the optimum pose are required to get the correct one (Section 4.3.3.3). Because of that, the use of the heuristic does not decrease robustness and increases significantly performance.

4.3.3.3 Keyframe Matching

All the steps involved in Keyframe Matching are shown in Figure 4.8 to facilitate comprehension.

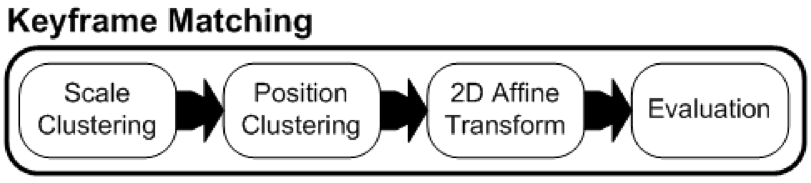


Figure 4.8: Keyframe Matching steps.

As a result of the previous steps, each keyframe has a list of *(image junction basis, virtual junction basis)* matches, i.e., matches have been clustered by keyframes. However, the matches of a single keyframe can represent different image positions at different scales. Thus, each keyframe matches must be clustered by both scale and position.

Scale Clustering

The scale (s) of each *junction basis* match is computed as:

$$s(junction\_basis_i, junction\_basis_j) = d_i/d_j, \tag{4.4}$$

where d_i is the distance between junction centres of *junction basis*$_i$ and d_j is the distance between junction centres of *junction basis*$_j$. Both distances are higher than 0 because it is a precondition to build a junction basis. To accelerate the process the scale space has been divided at predefined intervals. Additionally, overlapping intervals have been taken into account by associating each junction basis match with the two closest intervals.

Position Clustering

Once junction basis matches have been clustered by scale, then they are grouped by the image position they point to, similar to (Opelt et al., 2006). To achieve this, for each keyframe the gravity centre of all its junctions is calculated in the offline phase, and for each virtual junction basis the vector that is pointing to the gravity centre is stored (Figure 4.9). Therefore, in the online phase this vector is applied to the respective matched image junction basis with a proper scale, obtaining a unique image position. Junction basis matches that point to close image positions are grouped together, which is controlled by a threshold that has been set proportionally to the scale.

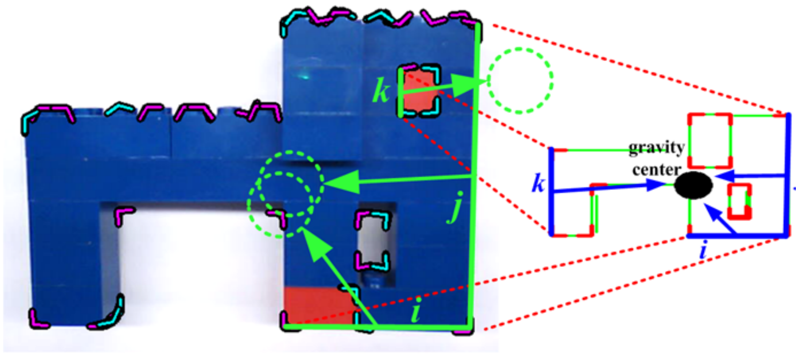


Figure 4.9: Junction basis clustering based on the image position they point to. Junction bases i and j are clustered together, while junction basis k is an outlier.

2D Affine Transform

After all clustering processes, all junction basis matches of a keyframe that share scale and image position are clustered together (c_i). The next step requires the analysis of each c_i cluster through the computation of the 2D affine transform (see Appendix C) that maps the virtual junction basis to the matched image junction basis. Each junction basis match provides two diferent 2D-2D point matches given by the implicit matching of its junction centers. Notice that although two junction basis matches are only required for each c_i test, the best affine transformation (a_i) in the least squares sense is retrieved for each c_i.

Evaluation

Afterwards, each a_i is evaluated, and the a_i with the highest score is selected for each keyframe. Only one a_i is maintained for each keyframe so that multiple instances of the model can be recognised in the same image, but from different points of view (Figure 4.16).

For the evaluation of each a_i the similarity measure explained in (Steger, 2002) and used successfully in (Ulrich et al., 2009) is applied. The set of 2D edges extracted in the offline-phase for each keyframe is transformed according to a_i (Figure 4.10), generating a set of edge samples characterised by their normals (e_i). Then, the similarity measure (m) is computed as the sum of the dot products between e_i and image edge gradients (g_i) at the corresponding transformed locations:

$$m = \frac{1}{n} * \sum_{i=1}^{n} \frac{|\langle e_i, g_i \rangle|}{\|e_i\|_2 * \|g_i\|_2}, \tag{4.5}$$

where n is the number of edge samples. Notice that the absolute value of the dot products is considered to ignore the contrast between the model and scene, which is unknown for keyframes taken from virtual processing. Modifications proposed in (Steger, 2002) are also used to speed up calculations.

Due to imprecision in the calculation of 2D affine matrices and projective distortions, the alignment of the transformed keyframe edges and image edges may be undesired (Figure 4.10), which decreases the value of m. Thus, image edges are searched along each e_i, as it is done for pose refinement (Drummond and Cipolla, 2002). The search is performed in a limited neighbourhood of each edge sample, proportional to the scale, until an image edge is found or the search space is exceeded. Although this increases the computational effort a bit, it is a necessary step to avoid false detections.

Based on the similarity measure of Equation 4.5 a_i candidates that do not fit well to image edges are discarded, i.e., a_i is ruled out in case that its corresponding m value is below a predefined threshold (thr_m). The score of the remaining a_i is computed as

$$m_a = m * v \tag{4.6}$$

where $v = |\{(e_i, g_i) \setminus |dot(e_i, g_i)| > thr_m\}|$. This favours candidates with good edge fitting and high scale (high number of edge samples). If multiple instances of the model are recognised, then the candidate with the highest scale covers the larger area of the image, and it is probably the view that the user wants to select. Due to this reasoning only the a_i with the highest

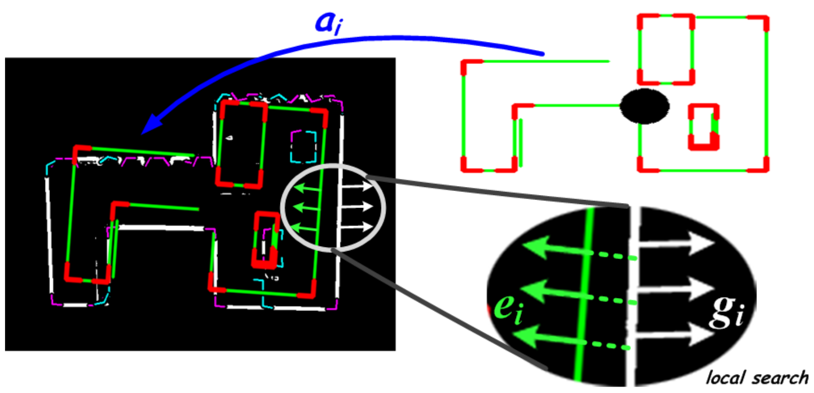


Figure 4.10: Similarity measure for a 2D affine transformation.

m_a score is maintained for each keyframe.

All the operations explained in this section are independent for each keyframe, so parallel techniques are applied easily. This is an essential step because the execution time is reduced considerably.

4.3.3.4 Pose Refinement

The previous keyframe matching process obtains a list of 2D affine matrices, one for each view in which the model is detected. However, in case that only one instance of the model would be required, then the global a_i with the highest m_a will be selected.

Once the subset of the target keyframes is selected, the final step requires the acquisition of the 3D pose for each one. For this aim the 2D data (scale, rotation and position) provided by each a_i can be interpreted as a combination of different 3D transformations, as it is done in (Ulrich et al., 2009). Nevertheless, here the efficient PnP algorithm described in (Lepetit et al., 2009) is applied because of its simplicity and good performance. 2D edge samples are transformed according to a_i and matched to their corresponding 3D values stored in the offline phase, formulating the PnP problem. Additionally, the 3D pose is refined using the procedure explained in (Drummond and Cipolla, 2002). 3D visible edges are projected, and for each projected 2D edge sample a local search is performed along the edge normal. Each 3D sample looks for the presence of edges in the corresponding projected image area. This produces multiple hypotheses for each edge sample, which are handled with a robust estimator. The resultant refined 3D pose is obtained from the non-linear optimisation of the reprojection error.

4.3.3.5 Pseudocode

The pseudocode of the Online phase is presented here (Algorithm 4.1) to improve comprehension and facilitate its reproduction.

4.3.4 First Camera Pose

Applications that use this method to solve the *first pose* problem can achieve greater robustness using a history of votes. In cluttered scenes, where some false positives may appear due to the presence of objects with similar shapes, a minimum number of repetitions may be requested to consider the extracted view as an inlier recognition. If a keyframe is detected in the current frame, it gets a vote. Thus, if in the last q frames a keyframe has been detected p times ($p << q$), then it is considered as an inlier view and it is returned. This scheme is supported by the idea that the user will try to focus on the model, covering the main area of the image. If the user realises that the model is not currently detected he/she will move the camera and will generate a sequence of different views that feeds the voting scheme.

4.4 Experiments and Results

This section evaluates the response of the proposed algorithm. All the experiments have been executed with an Intel Core i7-860 at 2.80GHz and 3GB of RAM. Images were captured with a Logitech QuickCam Connect webcam and a resolution of 640x480 pixels.

4.4.1 Parameter discussion

The most critical parameters to configure are related to the space partition that defines the junction basis data base. The range of target

Algorithm 4.1 Pseudocode of 3D object recognition.

```
 1: Input: camera image (I)
 2: Output: recognised 3D poses (P)
 3: K: set of keyframes
 4: $J \leftarrow JUDOCA(I)$
 5: for all $j_i \in J$ do
 6:     for all $j_k \in J$ do
 7:         if $distanceCenters(j_i, j_k) > 0$ then
 8:             $b_{ik} \leftarrow junction\_basis(j_i, j_k)$
 9:             $L_{matches} \leftarrow hash(b_{ik})$
10:             $updateMatches(K, L_{matches})$
11:         end if
12:     end for
13: end for
14: $F \leftarrow \emptyset$
15: for all $k_i \in K$ do
16:     $A \leftarrow \emptyset$
17:     $S \leftarrow clusterByScale(k_i)$
18:     for all $s_i \in S$ do
19:         $C \leftarrow clusterByPosition(s_i)$
20:         for all $c_i \in C$ do
21:             $a_i \leftarrow buildAffine(c_i)$
22:             $A \leftarrow A + a_i$
23:         end for
24:     end for
25:     $F \leftarrow F + maxScore(A)$
26: end for
27: $P \leftarrow \emptyset$
28: for all $f_i \in F$ do
29:     $p_i \leftarrow build3DPose(f_i)$
30:     $P \leftarrow P + p_i$
31: end for
```

views are specified by $[x_{min}, x_{max}]$, $[y_{min}, y_{max}]$ and $[z_{min}, z_{max}]$ and their corresponding steps Δx, Δy and Δz. The angle steps that characterise the number of bins for the global hash table must be also provided: $\Delta\beta_i$,

$\Delta\beta_j$ and $\Delta\theta_{ij}$ (Section 4.3.2.2). Additionally, the overlapping percentage between adjacent bins has to be set: ξ_i, ξ_j and ξ_{ij}.

To show the influence of the parameterisation, the proposed method has been executed several times with different parameter values and the same input images.

Input images were acquired in a scene similar to the one presented in Figure 4.11. Using a standard marker tracking system (Wagner and Schmalstieg, 2007), the relative pose between the marker and the camera is known. Likewise, a fixed transformation between the marker and the 3D model was set to obtain the relative pose between the 3D model and the camera. Furthermore, as the position of the 3D model in the scene and its corresponding projection could not match precisely (red wireframe of Figure 4.11), the reprojection error was minimized (Drummond and Cipolla, 2002) in order to achieve a precise match and increase the accuracy of the 3D pose (green wireframe of Figure 4.11). Finally, the camera was moved around the model in order to build a set of input views with a known 3D pose of the model.

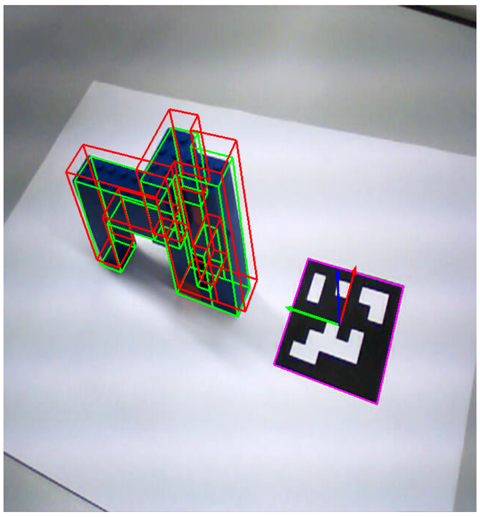


Figure 4.11: Image and camera pose acquisition for experiments.

$[x_{min}, x_{max}]$, $[y_{min}, y_{max}]$ and $[z_{min}, z_{max}]$ have been fixed to $[-10°, 90°]$, $[-90°, 90°]$ and $[-90°, 90°]$ respectively. Although smaller ranges could have been used in order to get lower computational times, the experiments have been configured for the general case by covering a wide range of views. Δx,

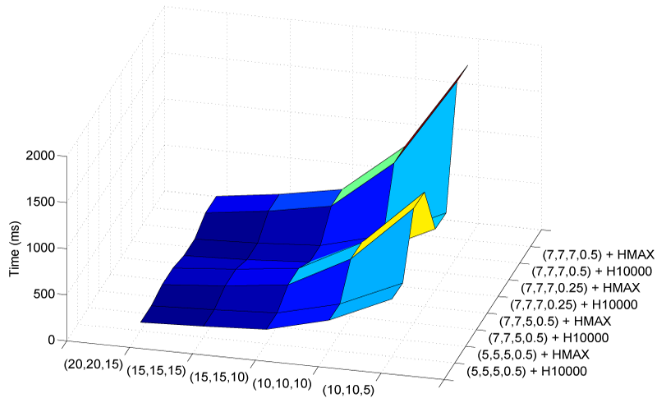
Δy and Δz values have been taken from the $[5°, 20°]$ range, while $\Delta\beta_i$, $\Delta\beta_j$ and $\Delta\theta_{ij}$ from the $[5°, 7°]$ range. ξ_i, ξ_j and ξ_{ij} have been set to 0.25 or 0.5. The experiments have been executed using a heuristic (Section 4.3.3.1) with a maximum of 10000 junction bases ($H10000$) and without any limitations ($HMAX$).

This procedure has been repeated for two different objects: the elephant and the 3D text presented in Table 4.1. The number of input images for each model has varied between 50 and 75. The results for both 3D models are presented in Figure 4.12 and Figure 4.13.

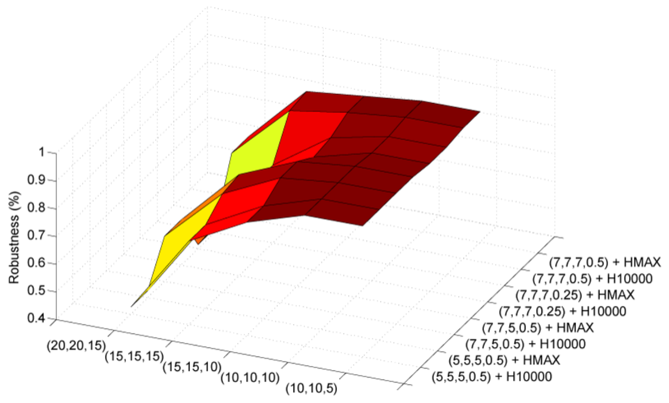
The computational time, accuracy and robustness of the proposed parameterisations have been compared. The computational time of each parameterisation is computed as the mean time for all input images. The accuracy, meanwhile, is calculated as the mean edge reprojection error between the input 3D pose and the 3D pose generated by our technique. Finally, the robustness is considered as the ratio between successfully recognised images and the total number of input images.

Analysing the pattern of Figures 4.12 and 4.13 the following general conclusions are extracted:

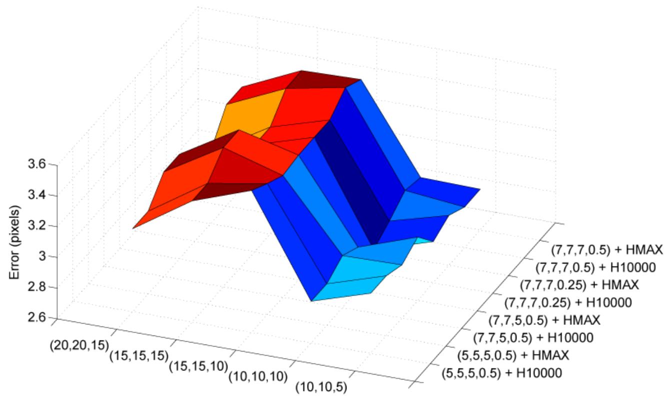
- Small Δx, Δy and Δz values produce more robust and accurate results at the expense of higher computational cost. Fine space partition increases the number of keyframes to match and consecuently the execution time is higher. The probability of getting a keyframe close to the input camera increases, which improves robustness and accuracy.
- $\Delta\beta_i$, $\Delta\beta_j$ and $\Delta\theta_{ij}$ angles with their associated ξ_i, ξ_j and ξ_{ij} values alter the results as well, but have less impact on them. Small intervals produce less *junction basis* matches in the Junction Hashing step (Section 4.3.3.2) which reduces the computational effort and decreases robustness.
- The results of the 3D text model are similar to those of the 3D elephant model, but with smaller computational times and more robustness (Figure 4.13). The 3D text model is simpler, with fewer junctions, so a simple junction basis database is built.
- The influence of the proposed heuristic is hardly appreciated in these results. The threshold of maximum junction basis is high enough



(a) Computational time.

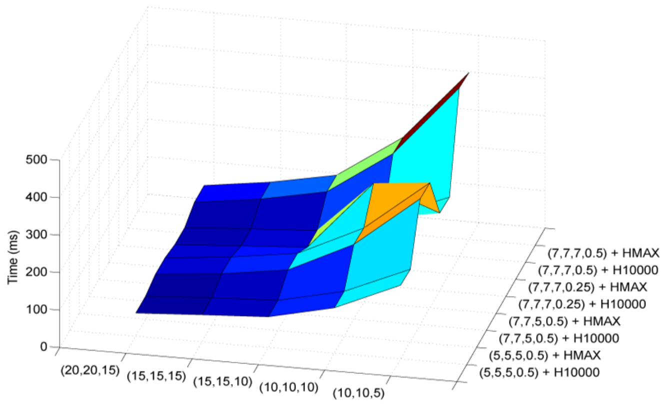


(b) Robustness.

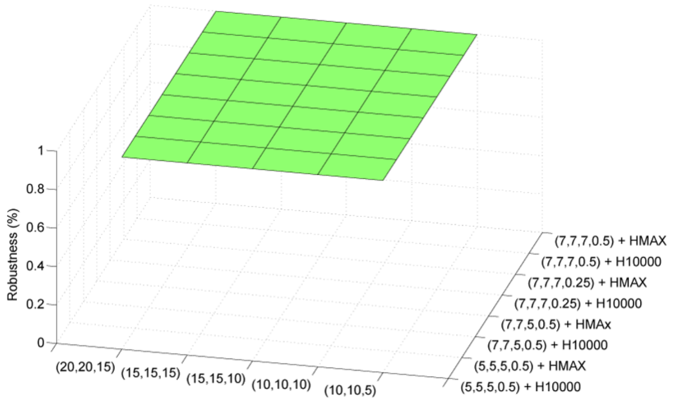


(c) Reprojection error.

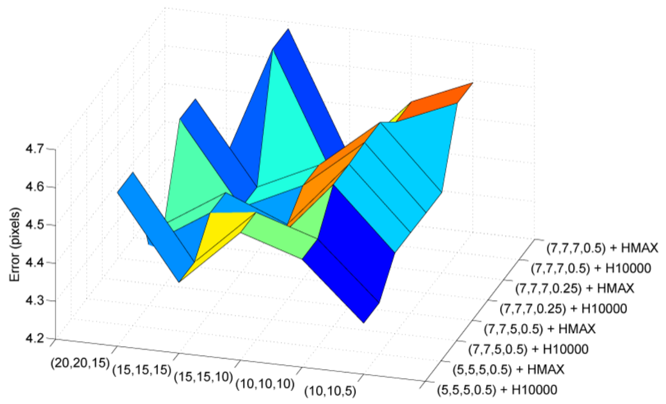
Figure 4.12: Recognition results for 3D elephant model.



(a) Computational time.



(b) Robustness.



(c) Reprojection error.

Figure 4.13: Recognition results for 3D text model.

to not jeopardise the matching. Its importance is only perceived in cluttered scenes (Figure 4.14). Because of that, the use of the heuristic is recommended, as it does not decrease the performance in general cases and it limits the computational time in cluttered scenes.

Taking these experiments into consideration, a good trade between execution time, robustness and accuracy is obtained using the following parameterisation: $\Delta x = 15°$, $\Delta y = 15°$, $\Delta z = 10°$, $\Delta\beta_i = 7°$, $\Delta\beta_j = 7°$, $\Delta\theta_{ij} = 7°$, $\xi_i = 0.5$, $\xi_j = 0.5$, $\xi_{ij} = 0.5$ and a heuristic with a maximum of 10000 junction bases. In addition, this parameterisation can be used with any model because the pattern of figures is similar for every single model.

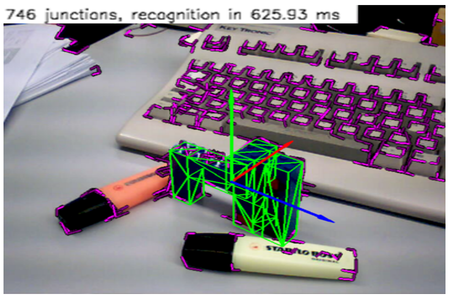
4.4.2 Examples

The response of the proposed 3D recognition method against different models and scenes is shown in Figures 4.14 and 4.15. The parameterisation suggested by the previous section is used to configure the 3D recognition, while the properties of each 3D model are detailed in Table 4.1. All of them are untextured models, with different shapes and materials.

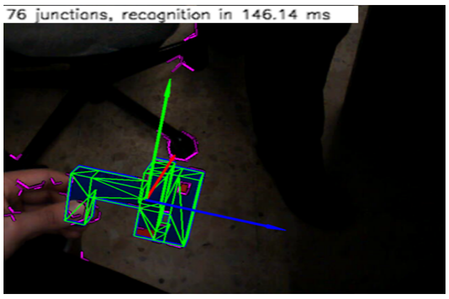
Figure 4.14 demonstrates that the proposed method works fine under difficult conditions. It is able to find the true 3D pose in very cluttered scenes (a)(c)(h). It can also retrieve the 3D pose in scenes with poor lighting (b) or against partial occlusions (b)(d)(e). Reflections of the mechanical pieces are supported satisfactorily as well (f)(g)(h). Moreover, the same junction basis database can be used for models with the same geometry, but different texture (e)(f).

Figure 4.15 also highlights that the proposed 3D recognition can handle complex models (53040 facets) positively.

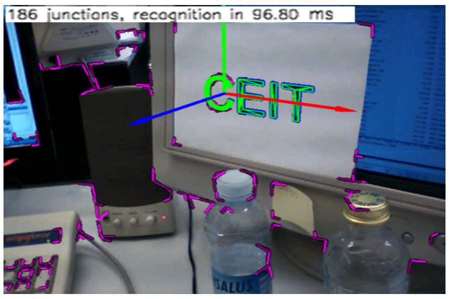
It is noteworthy that the execution time is maintained at reasonable values in unfavourable contexts, while it is quite low in scenes with few junctions (less than 51 milliseconds in (f)). In fact, in non-complex scenes this method can be used for tracking due to the high frame rate. In scenes with high number of junctions the hashing and matching steps consume most of the computational time, since many a_i candidates are processed. In non-complex scenes, however, JUDOCA is the critical step. In these examples, although JUDOCA reduces the number of junctions to build and decreases the computational time, hashing and matching steps are



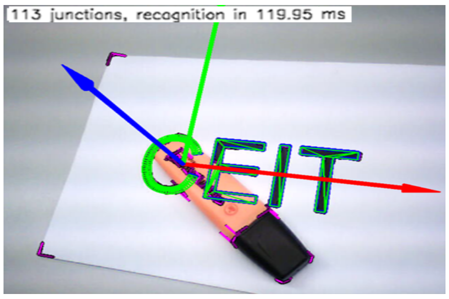
(a) Elephant in a cluttered scene.



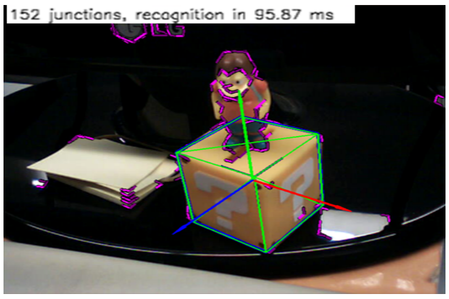
(b) Elephant against poor lighting.



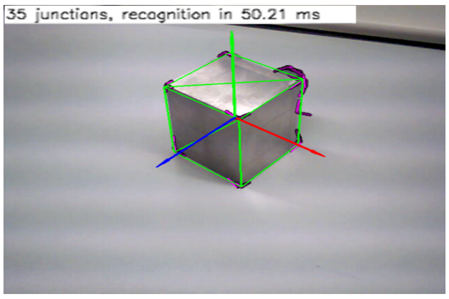
(c) 3D text in a cluttered scene.



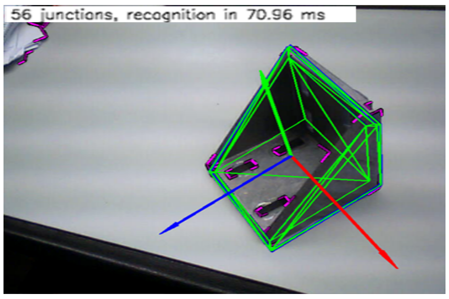
(d) 3D text with partial occlusion.



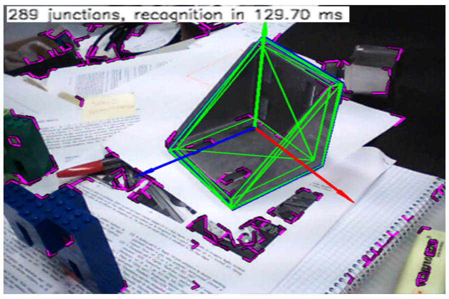
(e) Box with partial occlusion.



(f) Box with light reflections.

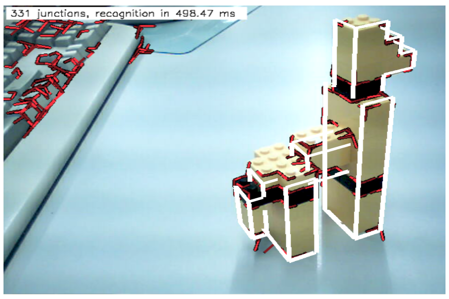


(g) Triangular bracket with light reflections.

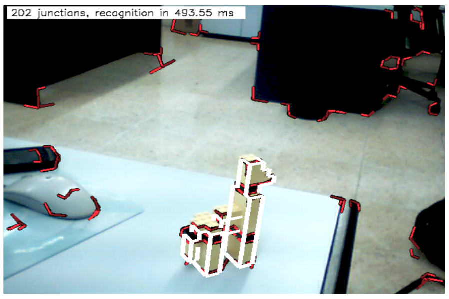


(h) Triangular bracket in a cluttered scene.

Figure 4.14: Recognition results for different scenes and models.



(a) Complex giraffe close to the camera.



(b) Complex giraffe far from the camera.

Figure 4.15: Recognition results for a complex model.

most benefited steps due to the low number of a_i candidates (see Table 4.2 for an example of computational times in these two scenes).

Table 4.2: Exec. time (milliseconds) for Figures 4.14 (a) and (f).

Scene	JUDOCA	Hashing + Matching	Refinement
Figure 4.14(a)	76.96	545.29	3.68
Figure 4.14(f)	33	14.65	2.56

Finally, Figure 4.16 illustrates that the proposed method detects multiple instances of a model in a single image. Here, the computational time does not suffer a noticeable peak, only the time associated to the pose refinement is increased, as it returns multiple pose instances.

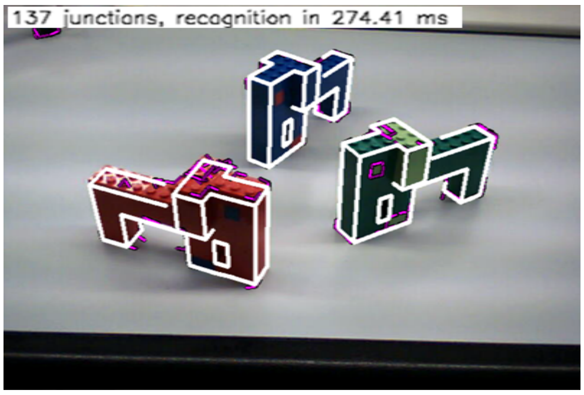


Figure 4.16: Recognition of multiple instances of a 3D model.

4.5 Discussion

In this chapter a solution for the 3D recognition of untextured 3D models in a single image has been detailed. Despite the lack of texture, it obtains good results because is based on geometric features of the model, which are extracted automatically, without user intervention. An efficient geometric constraint based on a pair of junctions is presented, which offers a sophisticated mechanism for matching and discarding the high amount of unnecessary matches that are generally processed by geometry based approaches. The computational effort is thereby decreased, obtaining reasonable execution times. Generally, the method presented here takes less than 100 ms to recognise a 3D object in a non-complex scene.

Parameterisation of the method has also been studied by providing a balanced configuration between robustness and performance. Indeed, although there are several parameters to configure, there is a single parameterisation that works fine for most models. This way, there is no need to modify parameters when different target models are used, which is a desirable property.

Some limitations should also be noted. As the recognition is based on matching junctions and edges over the entire image, complex scenarios with cluttered backgrounds may cause the detection of some false positives. This is noticeable when the user points to a keyboard, which has numerous edges, and therefore the system gives false positives detecting an object where none exist. If this happens, a simple camera shake will be enough to restart recognition. Another option would be to impose a restriction on colour similarity to validate the detection, which should be set during the offline phase. On the other hand, the proposed recognition relies on the existence of edges and junctions in the model, so its application domain is reduced to polyhedral objects. Nonetheless, it does not require texture information, it can handle textureless models. Thus, the proposed method offers an alternative to the problem of recognition, since it can work fine with untextured models that the existing 3D recognition methods based on texture cannot. Note that in case of trying to recognize an object without texture or without well-defined geometric properties, then the recognition would be almost impossible. Additionally, this thesis is oriented to industrial environments, where the presence of untextured polyhedral objects is quite common.

Chapter 5

Markerless Tracking

When you see a good move, wait,
look for a better one
Emanuel Lasker

This chapter presents a 3D markerless tracking algorithm adapted to the characteristics of industrial environments. It offers a robust tracking by combining multiple visual cues, so it is able to handle untextured 3D models.

A part of this chapter has been published in:

Barandiarán, J., Álvarez, H., and Borro, D. "Edge-based markerless 3d tracking of rigid objects". In International Conference on Artificial Reality and Telexistence (ICAT'07), pp. 282–283. Esbjerg, Denmark. November, 2007.

Álvarez, H., Aguinaga, I., and Borro, D. "Providing guidance for maintenance operations using automatic markerless augmented reality system". In Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR'11), pp. 181–190. Basel, Switzerland. October, 2011.

5.1 Introduction

Markerless tracking methods update the camera pose by using visual cues that are already in the scene, without having the need to modify the environment with artificial landmarks. As stated in Chapter 2, markerless tracking can be classified according to the knowledge about the geometry of the scene, but this chapter only focuses on *model-based markerless tracking that uses frame-to-frame techniques*. In other words, this chapter addresses the problem of finding the new camera pose giving the information of the previous frame and the geometry of the 3D model, since the target 3D models lack texture.

The proposed markerless tracking method assumes that the information about previous frame is available. Thus the 3D recognition method presented in Chapter 4 is used to obtain the first camera pose, i.e., to initialize the tracking. Then, the camera pose calculated at time $t - 1$ is used to feed the tracking at time t.

Additionally, some geometric features are extracted from the input 3D triangle mesh during an offline phase, similar to that explained in Section 4.3.2.1. This way, 3D sharp edges and 3D L junctions are automatically extracted, without user intervention. From here it is assumed that this information is available, but the explanation of this procedure is not repeated here to avoid duplications.

5.2 Proposed Method

Given the intrinsic camera parameters and the data generated in the previous frame, the frame-to-frame tracking developed in this chapter computes the extrinsic camera parameters by using temporal coherence and combining multiple visual cues. More precisely, self-supplied and robust markerless tracking that combines an edge tracker, a point based tracker and a 3D particle filter has been designed to continuously update the camera pose. Each of these tracking techniques are detailed below, including how they are combined to exploit their best properties. An in depth explanation about tracking methods that use different visual cues can be found in (Lepetit and Fua, 2005).

5.2.1 Edge Tracker

Edge based trackers are computationally efficient and stable to lighting changes, even for specular materials. Therefore, these solutions satisfy real time requirement besides tracking untextured 3D models. These methods can be grouped into two categories:

- **Without edge extraction:** strong gradients are looked for in the current image around the estimation of the previous object pose, without explicitly extracting the contours (Harris, 1992; Drummond and Cipolla, 2002; Vacchetti et al., 2004; Barandiarán et al., 2007).
- **With explicit edge extraction:** image contours are processed, and some primitives such as straight line segments are extracted and matched to model primitives (Lowe, 1992; Koller et al., 1993).

The first group is a fast and general approach, so it is used in this thesis. Indeed, RAPID (Harris, 1992), which belongs to the first group, was one of the first 3D trackers that worked successfully in real time. It considers a set of 3D points that lie on the 3D model edges, called control points, and the 3D motion of the model between two consecutive frames is recovered from the 2D displacements of these control points (Figure 5.1).

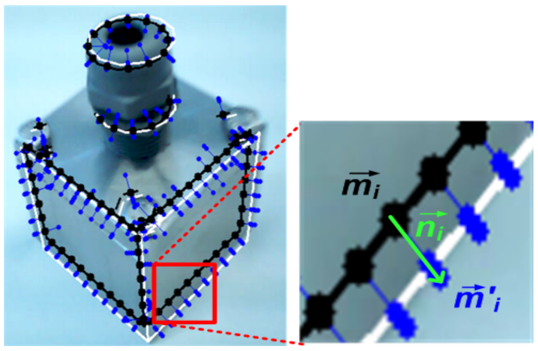


Figure 5.1: 2D displacements of control points. They are used to compute the motion of the model between the previous (black wireframe) and current frames (white wireframe).

Moreover, there are several ways to calculate the 2D displacements of control points. Some solutions consider those control points that lie on the same edge as a single primitive (stronger movement constraints)(Armstrong and Zisserman, 1995), while other alternatives consider multiple displacement hypotheses for each control point (Figure 5.2)(Vacchetti et al., 2004). This second option has been implemented for this dissertation due to its robustness against cluttered scenes.

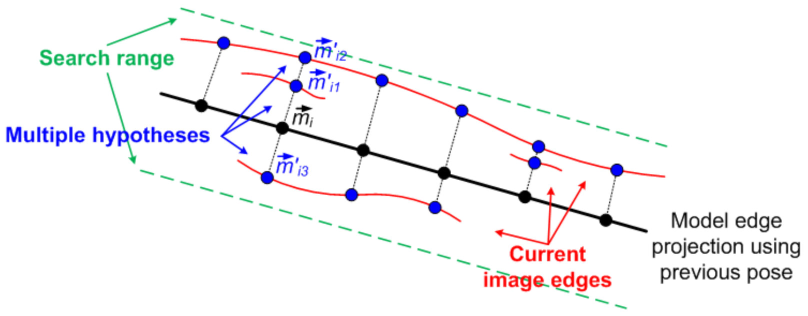


Figure 5.2: Multiple hypotheses for each control point.

Assuming that 3D edges are extracted in an offline phase, the 3D edge tracker performs a simple loop. Visible 3D edges are detected and projected into the current camera image using the OpenGL occlusion query and the previous camera pose respectively. Then, for each projected 3D edge a set of 2D edge samples ($\vec{m}_i$) are selected (control points, whose 3D coordinates $\vec{M}_i$ are known), and a local search along the edge normal ($\vec{n}_i$) is done to establish the presence of these samples in the current camera image ($\vec{m}'_i$) (Figure 5.1). If an image edge pixel is detected in the vicinity of a control point, then a positive correspondence is found. Note that this procedure can provide multiple match hypotheses for each control point (Figure 5.2). Finally, the camera pose of the current frame (R_t) results from the minimization of the distances between selected image edge pixels and projected control points (minimization of the reprojection error):

$$R_t = \underset{R_t}{\operatorname{argmin}} \sum_i P_{Tuk}\left(\min_j \left\|(\vec{m}'_{ij} - KR_t\vec{M}_i) * \vec{n}_i\right\|\right), \tag{5.1}$$

where $\vec{m}'_{ij}$ represents the hypothesis j for the control point i, and P_{Tuk} is the Tukey function that reduces the influence of outliers using the M-estimator approach:

$$P_{Tuk}(x) = \begin{cases} \frac{c^2}{6}\left[1 - \left(1 - (\frac{x}{c})^2\right)^3\right] & \text{if } |x| \leq c \\ \frac{c^2}{6} & \text{otherwise} \end{cases}, \tag{5.2}$$

where c is a bandwidth parameter that usually is set proportional to the standard deviation of the estimation error. This way, when $|x|$ is larger than

c Tukey estimator becomes flat and large residual errors have no influence at all.

5.2.2 Feature Tracker

The recursive edge tracking presented above is fast and precise, but it does not support fast movements. Indeed, several authors (Rosten and Drummond, 2005; Vacchetti et al., 2004) have integrated an accurate edge tracker with a robust feature tracker in order to cope with abrupt camera motions. It is well known that point based trackers can deal with rapid motions, since features are relatively easy to localize and match between frames. Following this reasoning, a point based tracker that is compatible with the previous edge tracker has been designed.

The proposed feature tracker combines the best properties of the point based trackers presented in (Bleser et al., 2005; Platonov et al., 2006). Hence, optical flow (Section 2.3.1.3) estimates the new feature locations, while simplified-SIFT descriptor (Appendix A) is used to get a more accurate position. It consist of a simple loop of two steps: *3D point generation* and *3D point tracking.* The first one back-projects features that belong to the surface of the model, while the second one performs the tracking of these features between consecutive frames.

- **3D point generation:** 2D features are extracted from the previous frame using the FAST operator (Section 2.3.1.1). Then, assuming that the pose of the previous frame is known, features that lie on the surface of the model are back-projected. This can be done efficiently using the OpenGL pipeline and the previous camera pose. Hence, each model facet is rendered with a unique colour so that each 2D feature location points to a unique colour, i.e., it can be used to index the corresponding facet. Once the 3D facet (with vertex $\vec{r}_1$,$\vec{r}_2$ and $\vec{r}_3$) in which the 2D feature lies is known, the 3D coordinates ($\vec{r}$) of the feature are computed using barycentric coordinates (λ_1, λ_2 and λ_3): $r = \lambda_1\vec{r}_1 + \lambda_2\vec{r}_2 + \lambda_3\vec{r}_3$ subject to $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Barycentric coordinates of a triangle represent the proportion of each triangle vertex to make a specified point the center of mass of the triangle. They are given by:

$$\lambda_1 = \frac{(y_2-y_3)(x-x_3)+(x_3-x_2)(y-y_3)}{(y_2-y_3)(x_1-x_3)+(x_3-x_2)(y_1-y_3)}$$

$$\lambda_2 = \frac{(y_3-y_1)(x-x_3)+(x_1-x_3)(y-y_3)}{(y_2-y_3)(x_1-x_3)+(x_3-x_2)(y_1-y_3)} \tag{5.3}$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2$$

where (x,y) and (x_i,y_i) are the image coordinates of point $\vec{r}$ and vertex $\vec{r}_i$ respectively.

Although many features can appear on the surface of the model, only a small subset of them are retained due to real-time requirements (30 features for this thesis). As a result, features with strongest response are selected first. Additionally, simplified-SIFT descriptor is stored (with a fixed patch size of 25 pixels) for each selected feature.

This step is executed every frame to continuously refresh the 3D point cloud and deal with appearing and disappearing points, which are usually caused by rotations.

- **3D point tracking:** features selected by the *3D point generation* step ($\vec{M}_i$) are tracked with pyramidal Lukas-Kanade optical flow (Section 2.3.1.3). This provides an estimation of the current 2D position of these 3D points. To get the correct 2D location ($\vec{m}'_i$), the current image is processed to find a feature with a similar simplified-SIFT descriptor (similar to the refinement step of Section 3.3.4.1). This search is limited to the vicinity of each feature (a window search of 75 pixels for this dissertation) due to the temporal coherence assumptions between consecutive frames. The matching of two simplified-SIFT descriptors follows the simple rule mentioned in Section 3.3.4.2: *two descriptors are matched if the euclidean distance of the second nearest descriptor is significantly greater than the distance to the nearest descriptor*. If no match is found, this 2D-3D correspondence is discarded, otherwise it assists in the calculation of the new camera pose. A non-linear optimisation (Levenberg-Marquardt) of the reprojection error of the selected correspondences is performed by using the previous camera pose as an initial guess. More precisely, the pseudo-Huber robust cost function (Platonov et al., 2006) is minimized to reduce the influence of outlier correspondences, similar to the Tukey function of the edge tracker[1]:

$$R_t = \underset{R_t}{\operatorname{argmin}} \sum_i P_{Hub}\left(\left\|\vec{m}_i' - K R_t \vec{M}_i\right\|\right), \tag{5.4}$$

where $P_{Hub}(x) = 2b^2(\sqrt{1+(x/b)^2}-1)$ and b is a bandwidth parameter that usually is set proportional to the standard deviation of the estimation error.

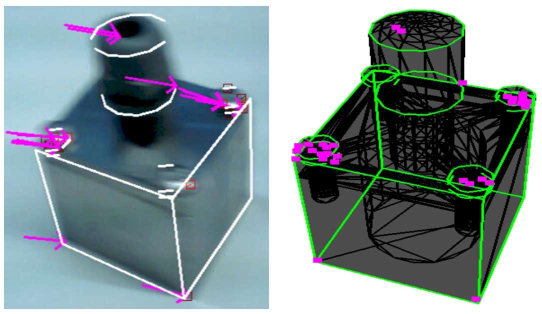
Figure 5.3 shows an example of the proposed feature tracker. Fuchsia points display back-projected features, fuchsia arrows represent 2D displacements of the surface points, and the white wireframe is the estimated camera pose. Notice that although the 3D point generation step can make an erroneous back-projection of features due to occlusions (points that lie on the marker pen, Figure 5.3(b)), the feature tracker remains stable as long as the occlusion is not severe.

The new camera pose that is calculated by using the 2D displacements of the features that lie on the surface of the model is used as a starting point for the edge tracker presented above.

5.2.3 Particle Filter

The combination of the edge and feature trackers presented above offers good tracking results. However, the feature tracker only works when a minimum set of features are detected on the surface of the model. This requirement is generally satisfied, since industrial models are composed of several parts, which favours the presence of corners. An example of this property can be observed in Figure 5.3, where assembled screws generate multiple salient points on the surface of the box. Note that this is not the same for all models (Box model of Table 4.1), where not enough features are detected due to the homogeneity of the outer surface. Following this reasoning, a 3D *particle filter* that uses multiple visual cues has been incorporated to the proposed markerless tracking, making it able to handle objects with different surface materials.

[1]Huber estimator makes the convergence to a global minimum more reliable, while Tukey estimator is preferred to remove the influence of outliers when a starting point close to the actual minimum is provided.


(a) Fast camera movement.

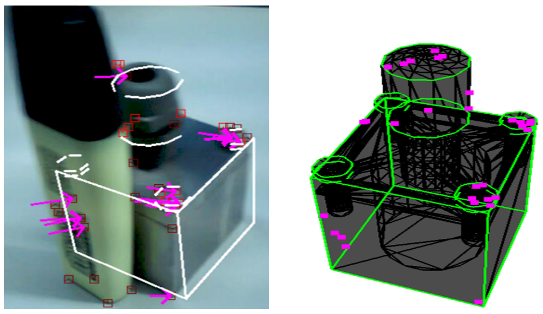

(b) Fast camera movement plus an occlusion.

Figure 5.3: 3D point tracking (left) and generation (right).

Particle filters belong to the family of *Sequential Monte Carlo* (SMC) methods, and are robust against non-static scenes in which multi-modality is likely (Pupilli and Calway, 2005). The key idea is to represent the required posterior distribution of the camera motion by a set of random samples with associated weights and to compute estimates based on these samples and weights (Figure 5.4). They are divided into two stages: *particle generation* and *particle evaluation*. In the particle generation step, the previous camera pose is perturbed to generate multiple camera pose candidates of the current frame (particles). However, it is not possible to generate samples directly from the target distribution, so they are drawn from an approximated distribution called *importance density*. The particle evaluation step, on the other hand, is responsible of assigning a weight to each particle and selecting the correct one. Due to its recursive nature, particles of time i are propagated using the particles of time $i-1$, so their associated weights also depend on their previous values (*Sequential*

Importance Sampling, SIS). This recursive propagation involves the problem of *degeneracy*, where after a few iterations all but one particle will have negligible weight. In order to avoid this, only particles with higher weights are augmented and those with low weights are discarded, which is a technique called *Sequential Importance Resampling* (SIR). Both robustness and computational cost are proportional to the number of particles, so this parameter is critical to reach good results within reasonable computational times. An in depth discussion about particle filters can be found in (Arulampalam et al., 2002; Salih and Malik, 2011).

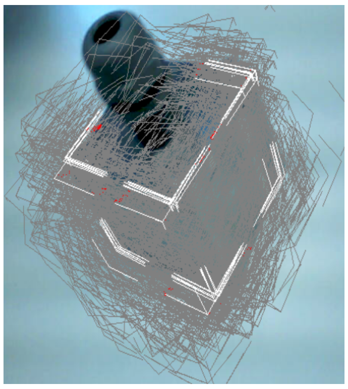


Figure 5.4: Tracking using a particle filter. Samples are drawn in gray. White indicates particles with higher weight.

One of the greatest difficulties to be faced when working with particle filters consist of determining the functions that generate and evaluate particles. For this thesis, a random walk motion model is used to generate particles, i.e., particles are distributed uniformly along the interest search space. This decision has been taken because there is no information or constraints about the type of movement that the camera is going to perform. The particle evaluation step, meanwhile, uses the 3D junctions and edges extracted in the offline phase as well as the 3D points back-projected by the feature tracker. More precisely, the likelihood (w) of each particle ($\mathbf{P_i}$) is inversely proportional to the reprojection error of visible 3D junction centers (X_i) and back-projected 3D features (Y_i), taking into account how many pixels of the projection of visible 3D edges coincide with an image edge:

$$w(\mathbf{P_i}) = \frac{|V_i| \, / \, |Z_i|}{\sum_{j=1}^{n} P_{Hub}(\delta(\mathbf{P_i}, X_j)) + \sum_{j=1}^{m} P_{Hub}(\delta(\mathbf{P_i}, Y_j))} \tag{5.5}$$

where $|Z_i|$ is the number of pixels that results from the projection of 3D visible edges with the camera matrix $\mathbf{P_i}$ provided by the particle i, $|V_i|$ is the number of Z_i pixels whose distance to the closest image edge point is below a predefined threshold, δ defines the reprojection error, P_{Hub} represents the pseudo-Huber function (Section 5.2.2), n is the number of visible 3D junction centres and m is the number of back-projected 3D features.

The current 2D locations of the corresponding points must be known in order to calculate the reprojection error. Thus, those 3D junction centres that were visible in the previous frame are projected and tracked using a pyramidal Lukas-Kanade optical flow (Section 2.3.1.3). Notice that the current 2D positions of back-projected 3D features are known because they have already been processed by the feature tracker, so they are not calculated again by the 3D particle filter. All these 2D translation estimations are also used to fix the optimum number of particles in accordance with demand (between 200 and 1000 for this dissertation). If strong 2D translations are detected the number of particles is increased, while in the absence of movement few particles are needed.

Efficient *distance transforms* are used to obtain the ratio of inlier edge pixels ($|V_i|/|Z_i|$), similar to (Klein and Murray, 2006), but executing all the calculations in the CPU. A distance transform of an image stores for each pixel the distance to the closest edge of the image (Figure 5.5). Based on this definition, visible 3D edges are projected and for each 2D edge sample the distance to the closest edge pixel is obtained with a simple look-up in the image distance transform. Thus, Z_i is the set of all 2D edge samples, while V_i is the set of 2D edge samples whose distance to the closest edge pixel is below a predefined threshold. Furthermore, orientation of edges is considered to discard false matches, as it is done in (Heisele and Rocha, 2008). In fact, the current camera edge image is split into four orientation specific edge images and for each edge image its corresponding distance transform is computed. This way, an input 2D edge sample is only compared to the image distance transform indexed by its orientation.

In addition, visible 3D edges and junctions are pre-computed for a set of points of views along a 3D sphere (analogous to Virtual Keyframe Generation of Section 4.3.2.2) in order to reduce the computational cost.

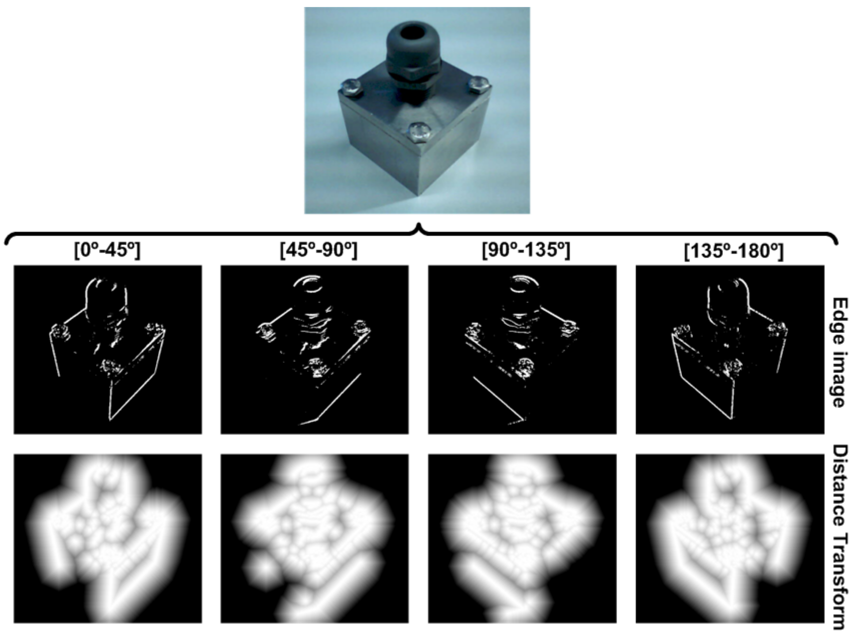


Figure 5.5: Oriented distance transform (white ≡ edge proximity).

This increases memory requirements, but decreases the execution time because given an input camera pose its visible 3D edges and junctions are obtained with a simple look-up.

A particle annealing is also applied to get correct results with a limited number of particles (Deutscher et al., 2000; Gall et al., 2007). Particle annealing is a process in which samples are iteratively focused onto potential modes. More precisely, particles are propagated with an uniform distribution with smaller width and re-weighted with a more stringent function (Equation 5.5) at each annealing step, which results in greater concentration of the samples around the most probable particle clusters. This technique allows moving towards the global maximum without being distracted by local minima (Figure 5.6).

Analogous to the feature tracker, the new camera pose estimated by the particle filter is used as an initial guess for the edge tracker presented above.

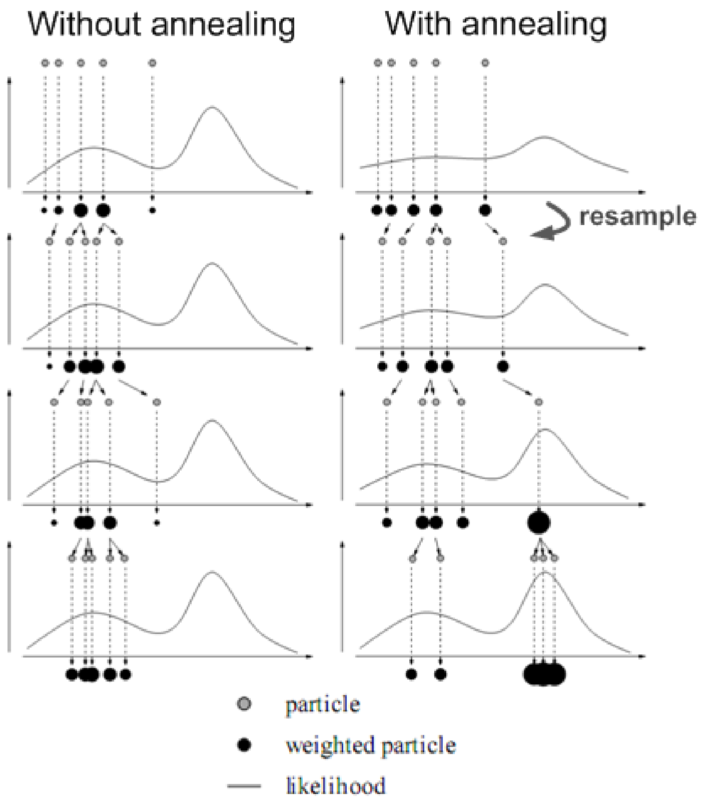


Figure 5.6: Particle annealing effect to avoid local minimums. Image courtesy of (Gall et al., 2007).

5.2.4 Integration of Multiple Trackers

The overall markerless tracking scheme is shown in Figure 5.7. The feature tracker is executed first to handle strong movements. If a minimum number of 3D points are back-projected (7 for example), then the 3D point tracking step is executed and the camera pose is updated. Otherwise the 3D particle filter is called to approximate the current camera pose. Afterwards, the edge tracker is executed to refine the camera pose, avoiding the possible error caused by a non-accurate reconstruction of the 3D point cloud or non-accurate particle likelihood calculation. In fact, the edge tracker is executed several times recursively to obtain more accurate results, since after each iteration the estimated camera pose is closer to the real one (a good estimation is obtained after 2-3 iterations). The feature tracker and the 3D particle filter offer a good initial guess of the correct camera pose, while the edge tracker calculates a precise one.

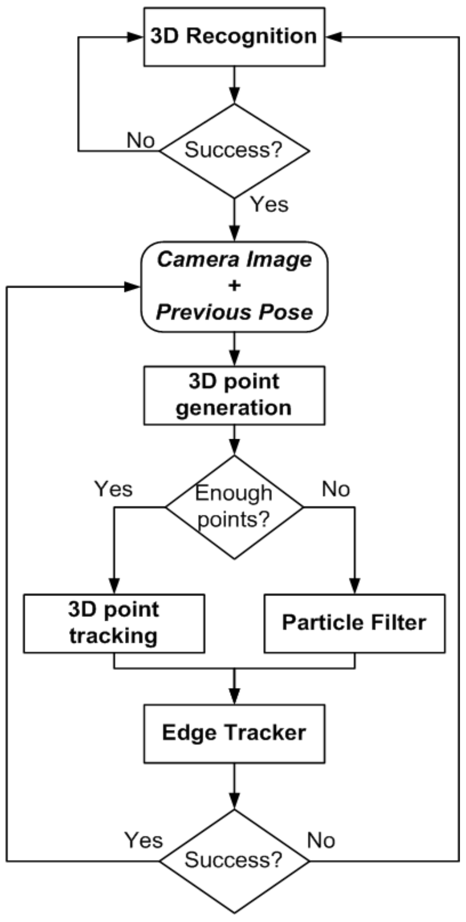


Figure 5.7: Markerless tracking algorithm.

It should be noted that both the feature tracker and the particle filter are executed in down-sample resolution images to reduce the computation time. The edge tracker, meanwhile, is executed at the original scale to get more accuracy (Section 5.3).

This procedure is executed every frame until the edge tracker fails. In order to determine the success of the edge tracker the similarity measure explained in Section 4.3.3.3 is used. Visible 3D edges are projected using the camera pose refined by the edge tracker, and the score is the sum of the dot products between the gradients of these projected samples and the image edge gradients at the corresponding locations. If this score is below a predefined threshold (~ 0.7), then the edge tracker fails and the system starts the recovery mode and returns to the 3D recognition stage (Chapter 4).

5.3 Experiments and Results

This section presents the results of the proposed markerless tracking algorithm. The hardware setup consists of an Intel Core i7-860 at 2.80GHz and 3GB of RAM equipped with a Logitech QuickCam Connect webcam.

To prove the benefits of the proposed tracking method the same video sequence (640x480 resolution) has been executed with three different tracking configurations. The first one combines a feature tracker, a particle filter and an edge tracker (*Point+PF+Edges*). The second one disables the particle filter (*Point+Edges*), and the last one disables the feature tracker (*PF+Edges*). The main goal of this comparison is to show the advantages of using different types of tracking techniques together, obtaining a robust tracking and satisfying real time requirements.

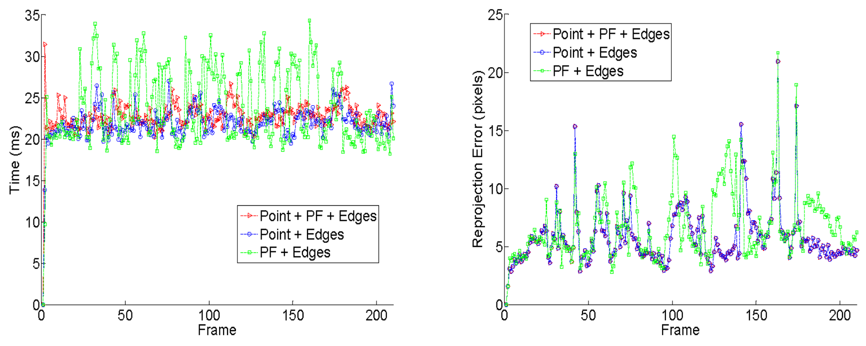
All the process has been repeated with three different 3D models (Box-1, Matryoshka and Gear-box of Figure 5.9, whose geometric properties are shown in Table 6.2) to simulate multiple conditions. These three models offer the majority of possible scenarios. The Box-1 model represents an industrial model with some corners on its surface, the Matryoshka model represents a model with an homogeneous outer surface (without feature presence) and the Gear-box represents an industrial model with complex geometry.

The feature tracker and the particle filter perform their calculations in down-sampled resolution (320x240) to reduce the computational time, while the edge tracker is executed in the original scale (640x480) to achieve high accuracy. Additionally, the particle filter is limited to a maximum set of 1000 particles to control the total runtime.

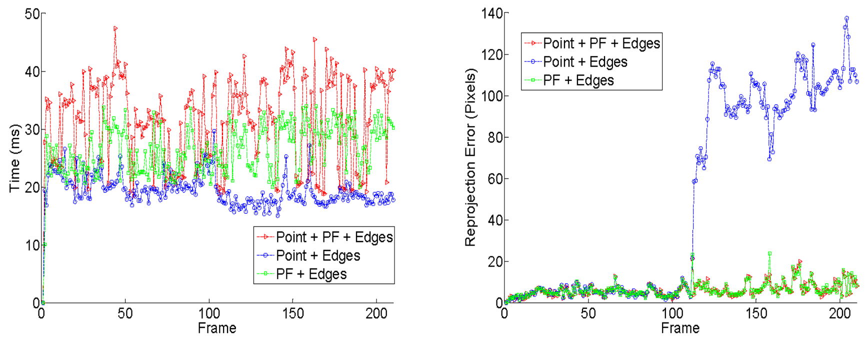
Similarly to Section 4.4, a marker has been placed in the scene, and the real camera pose of each video frame has been obtained using the ARtoolkitPlus marker tracking system. This allows to calculate the error between the estimated and real camera poses. However, it is noteworthy that there is an increase in measurement error because the video sequence consists of rapid camera movements that try to lose the tracking. The non-perfect calibration between the marker and model reference systems also distort error measurements.

The execution time and the reprojection error of different tracking configurations and 3D models is shown in Figure 5.8 for the input video sequence. Box-1 and Gear-box are tracked successfully with the three tracking configurations, while the (Point+Edges) option fails for the Matryoshka example after frame 110 (Figure 5.9(b)). The reason for this failure is that the Matryoshka model has an homogeneous outer surface where not many features are detected. Therefore, in the absence of features the point based tracker fails, and the particle filter is the more robust tracker. Nevertheless, in the Box-1 example, where many features lie on the surface of the box due to the presence of screws, the feature tracker runs faster and has lower reprojection error than the particle filter (Figure 5.9(a)). For the Gear-box example, although the feature tracker is slower than the particle filter, its reprojection error is lower (Figure 5.9(c)). In addition, it has been observed during experiments that the feature tracker can track fast camera movements that the particle filter option cannot. Thus, the feature tracker is the best option for most examples, as it is very common to detect a minimum set of features on the surface of the model. A tracker that combines both of them gets the best properties of each option, running fast and accurately for models with features on their surface and offering high robustness in absence of features.

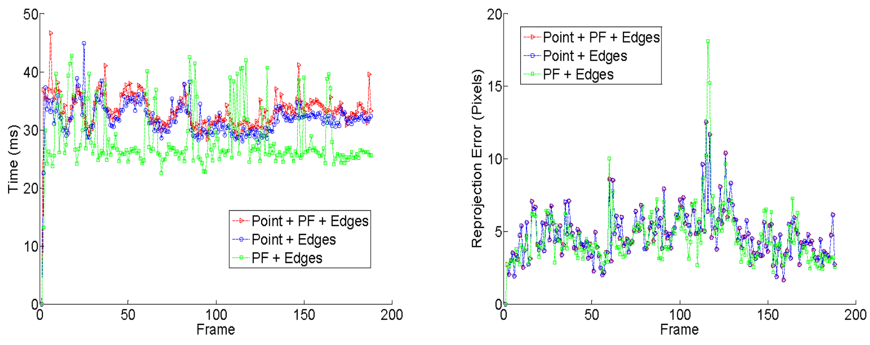
The execution time of Box-1, Matryoshka and Gear-box models for the video sequence using (Point+PF+Edges) configuration is detailed in Table 5.1. It displays the execution time required by each tracking step, including the *Image Processing* step, which handles a large camera image (640x480). Although the feature tracker includes the computation of the simplified-SIFT descriptors parametrized with $4 * 4$ regions and 8 orientations ($4 * 4 * 8 = 128$ bins), its execution time remains low. Note that the total execution time is kept below real time limits.



(a) Box-1.

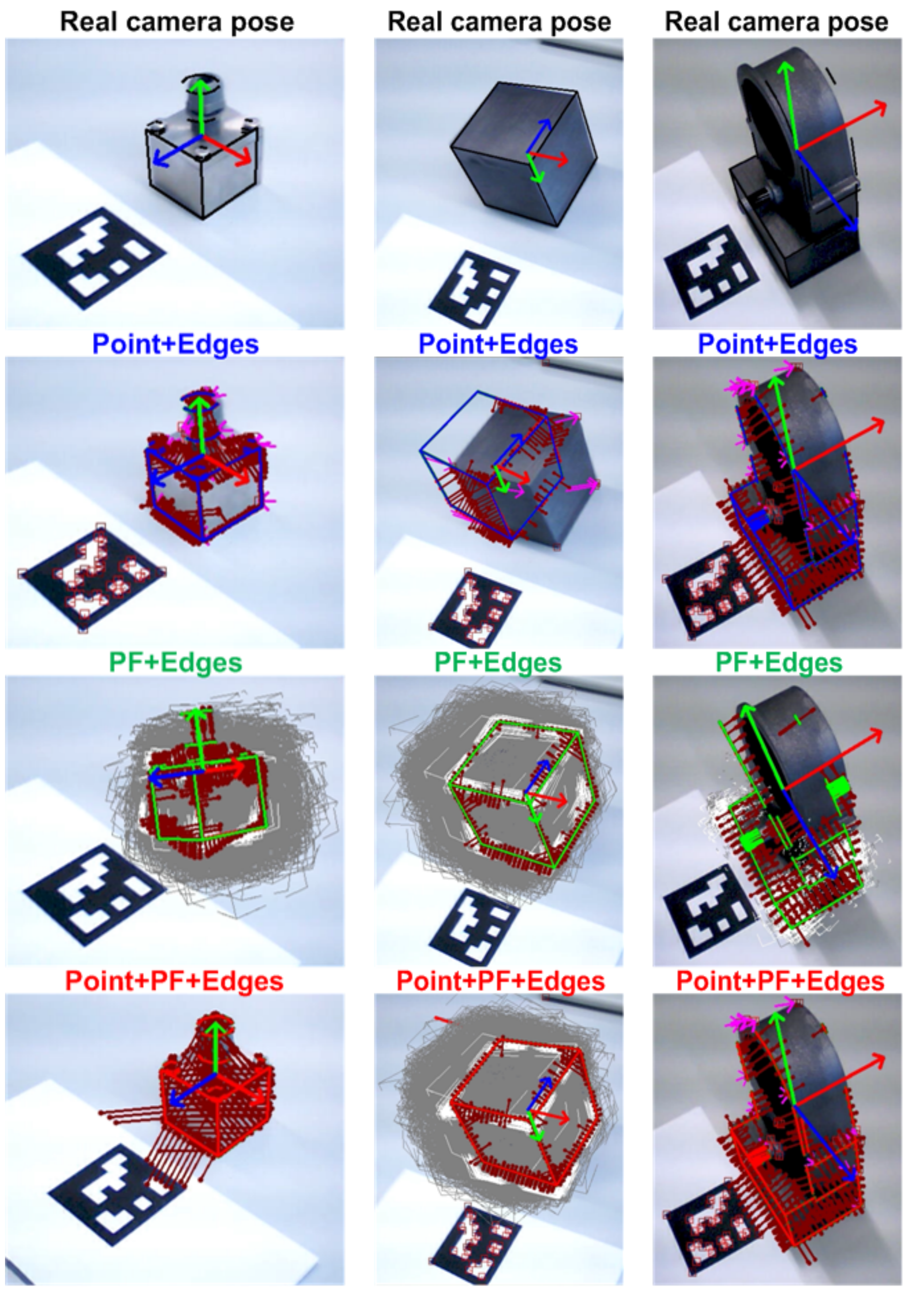


(b) Matryoshka.



(c) Gear-box.

Figure 5.8: Exec-time and error for different tracking setups.



(a) Box-1, frame 130. (b) Matryoshka, frame 111. (c) Gear-box, frame 114.

Figure 5.9: Response of different tracking configurations.

Table 5.1: Exec-time of (Point+PF+Edges) for the input video.

	Time (ms)					
	Box-1		Matryoshka		Gear-box	
	Mean	Std.	Mean	Std.	Mean	Std.
Image Processing	7.29	0.7	7.29	0.57	7.41	0.62
Feature Tracker	12.14	1.49	8.8	2.4	19.72	2.85
Particle Filter	0.78	0.1	11.51	7.73	1.47	0.41
Edge Tracker	2.47	0.76	3.65	0.8	4.58	0.55
Total	22.73	2.15	31.3	7.66	33.23	3.54

5.4 Discussion

There are several markerless tracking techniques in the literature, each one with its own advantages and disadvantages. Some of them works fine when some distinguishable points are detected in the scene (tracking based on features), supporting even fast camera movements. Other approaches, meanwhile, handle multiple motion hypotheses (particle filters), whose validity can be verified using different image measurements, not just features. These solutions are robust, but require significant computational time and their accuracy is limited by the number of hypotheses (particles). Finally, other alternatives such as those based on edges are computationally efficient methods that offer high accuracy, being stable to lighting changes.

Following this reasoning, in this chapter a robust markerless tracking method that uses multiple tracking techniques has been presented. It combines an edge tracker, a feature tracker and a 3D particle filter to offer a robust 3D tracking against multiple and undesirable conditions such as homogeneous surfaces that can be found in many industrial objects. The feature tracker offers robustness against rapid camera movements, the particle filter is used as a robust alternative in the absence of features (untextured scenes) and the accurate edge tracker refines the final camera pose. More precisely, the particle filter is called in case that the feature tracker fails. In this situation the information processed by the feature tracker is reused to determine the optimal number of particles, which is set according to the type of motion observed and real time requirements. Furthermore, the camera pose calculated by the feature tracker or by the particle filter is used as an initial guess by the edge tracking to obtain accurate results. This way, different tracking techniques have not been used independently, but they have been integrated efficiently to build a single, robust and real time markerless tracking method.

Besides combining multiple tracking methods, these tracking techniques have also been modified to suit the initial preconditions of the thesis, i.e., real time, automatic nature and industrial environments. The feature tracker uses simplified-SIFT descriptors to fulfil real time limits. It also generates and refreshes automatically the 3D point cloud. Likewise, the particle filter is adapted dynamically by setting the number of particles proportionally to the motion, which reaches a balance between robustness and computational time. Indeed, this last technique has been configured to use multiple image measurements (edges, junctions and features) and handle objects with homogeneous surfaces, typical of industrial models.

Additionally, this markerless tracking has been combined with the 3D recognition method proposed in Chapter 4. The 3D recognition method initializes and restarts the tracking in case of failure, so a complete solution is offered.

As argued throughout all previous paragraphs, this chapter describes a complete framework for a self-supplied, robust and real time markerless tracking that can be used in multiple conditions, including untextured 3D models of industrial environments.

Chapter 6

AR Disassembler

The winner of the game is the player who makes
the next-to-last mistake
SAVIELLY TARTAKOWER

This chapter describes a new real-time augmented reality tool to help in disassembly during maintenance operations. This tool provides workers with augmented instructions to perform maintenance tasks more efficiently.

In order to compute the camera pose and render augmented instructions, the 3D recognition and tracking methods presented in Chapters 4 and 5 are used, respectively. Therefore, this chapter shows a real world applicability of the methods proposed throughout this thesis.

A disassembly planing module is also integrated to offer a complete system, obtaining automatically the assembly/disassembly sequence of an input 3D model.

A synthesis of this chapter has been published in:

Álvarez, H., Aguinaga, I., and Borro, D. "Providing guidance for maintenance operations using automatic markerless augmented reality system". In Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR'11), pp. 181–190. Basel, Switzerland. October, 2011.

6.1 Introduction

Augmented Reality is a set of technologies that enrich the way in which users experience the real world by embedding virtual objects in it that coexist and interact with real objects. This way, the user can be exposed to different environments and sensations in a safe and more realistic way. Because of that, this chapter describes a real time automatic markerless AR disassembler for maintenance and repair operations.

The system generates instructions indicating which the next step is and the way to proceed. The instructions are displayed graphically by superimposing a virtual representation of the next step on top of the real view of the system being repaired. Thus, the next part to disassemble is virtually coloured in red and translated along the extraction path. Indeed, a virtual arrow emphasize the extraction direction. This online assistance reduces the time spent by technicians searching for information in paper based documentation, so it enables faster and more reliable operation.

The main challenge addressed by this AR system is the automatic generation of the suitable information to provide user feedback. The system receives as an input a single untextured 3D triangle mesh of each component. The assembly/disassembly sequence is computed automatically from this input, finding collision-free extraction trajectories and the precedence relationship of the disassembly of the different parts composing the target system (Section 6.3.1). Additionally, some geometric features, such as edges and junctions, are also automatically extracted to identify and track the different components during their manipulation (Section 6.3.2).

6.2 Previous Works

Several works (Raghavan et al., 1999; Tang et al., 2003) demonstrate the benefits of AR based approaches applied to the assistance in maintenance and repair operations in terms of operation efficiency. Compared to Virtual Reality (VR), AR offers a more realistic experience because the user interacts with real objects, while VR manipulates virtual objects and is limited by the lack of suitable sensors feedback. Further, VR is oriented to training rather than guidance, improving the skills of users through multiple simulations, while AR can be used for both training and guidance.

Motivated by this reasoning, many authors have been addressed the problem of building an AR tool that provides guidance for maintenance operations. The main characteristics of some of these solutions are detailed below.

(Yuan et al., 2008) describes an AR application for assembly guidance using a virtual interactive tool. It offers a visual interface with an intuitive interactive mechanism. The user controls different instruction messages with a pen and the assembly steps are displayed in the image. However, instructions are limited to 2D photographic images on the camera image border. The system does not perform 3D model tracking to offer 3D augmentation data. Similarly, (Baird and Barfield, 1999) describes an AR maintenance prototype based on an untracked system.

Other approaches (Weidenhausen et al., 2003; Zauner et al., 2003; Liverani et al., 2004; Billinghurst et al., 2008; Schoenfelder and Schmalstieg, 2008; Farkhatdinov and Ryu, 2009; Henderson and Feiner, 2009) rely on marker-based tracking systems to compute the camera position and orientation. Using these systems, instructions are registered in the real world with a high degree of realism. They achieve high accuracy with low computational cost. Nonetheless, they require environment adaptation, which is not always possible, and marker and 3D model coordinate systems must be properly calibrated by the user. In addition, marker tracking systems can fail when the markers are partially occluded (Section 3.1.2). This is a likely scenario in maintenance operations, since occlusions with the hands of the worker and tools are common.

Other alternatives use natural features to retrieve the 6 DOF of the camera. During an offline process, some views of the scenario are taken. These keyframes are used as input for a training process, where 2D-3D correspondences are established by back-projecting detected 2D image features into the known 3D model. During the online phase, 2D features are extracted from the camera image and matched to those that were indexed in the offline phase, obtaining the camera pose. Using this procedure, (De Crescenzio et al., 2011) describes an application for airplane maintenance, while (Platonov et al., 2006) details a prototype for car repair guidance. These solutions do not modify the environment, but they need some user intervention to generate the training datasets. They are also optimised for textured scenes, which do not always exist in industrial environments.

Some interesting solutions do not need the 3D model, but require high interactivity with users (Reitmayr et al., 2007). Some virtual annotations (disassembly instructions) are added manually to the scene by clicking on the image position that should be. Then, the system estimates the 3D location of the notes automatically, which are rendered correctly registered to the environment using SLAM techniques.

Compared to all these works, the proposed approach is a more robust framework. It is able to compute automatically the assembly/disasssembly sequence, which has not been addressed by any of the papers cited above, since this information is usually considered an input parameter given by the user. Additionally, the proposed markerless tracking handles untextured 3D models. All the tracking data is extracted automatically from the model geometry, without user intervention. Based on these statements some problems outlined earlier are overcome: environment adaptation and textureless scenes.

6.3 Proposed AR Disassembler

Figure 6.1 shows all the steps of the proposed AR framework. The automatic offline phase receives as input a 3D model of the system to maintain. This model is composed of several parts, each one with its own 3D triangle mesh and located in its assembly position (without texture). With this minimal information, the disassembly-planning module is able to extract the precedence order in which the parts should be placed or removed (Section 6.3.1). Furthermore, some geometric features, such as edges and junctions, are extracted to address the tracking problem (Section 6.3.2). These geometric features let us perform the 3D model recognition and tracking during the online phase. Within this phase, once the camera pose has been computed, the system offers an augmented instruction, indicating how to assemble/disassemble the next part. Then, the users perform the corresponding operation and notify the system that they want to continue with the next step. Note that this notification has been implemented as a simple keystroke, but it could have been implemented with a more sophisticated voice command.

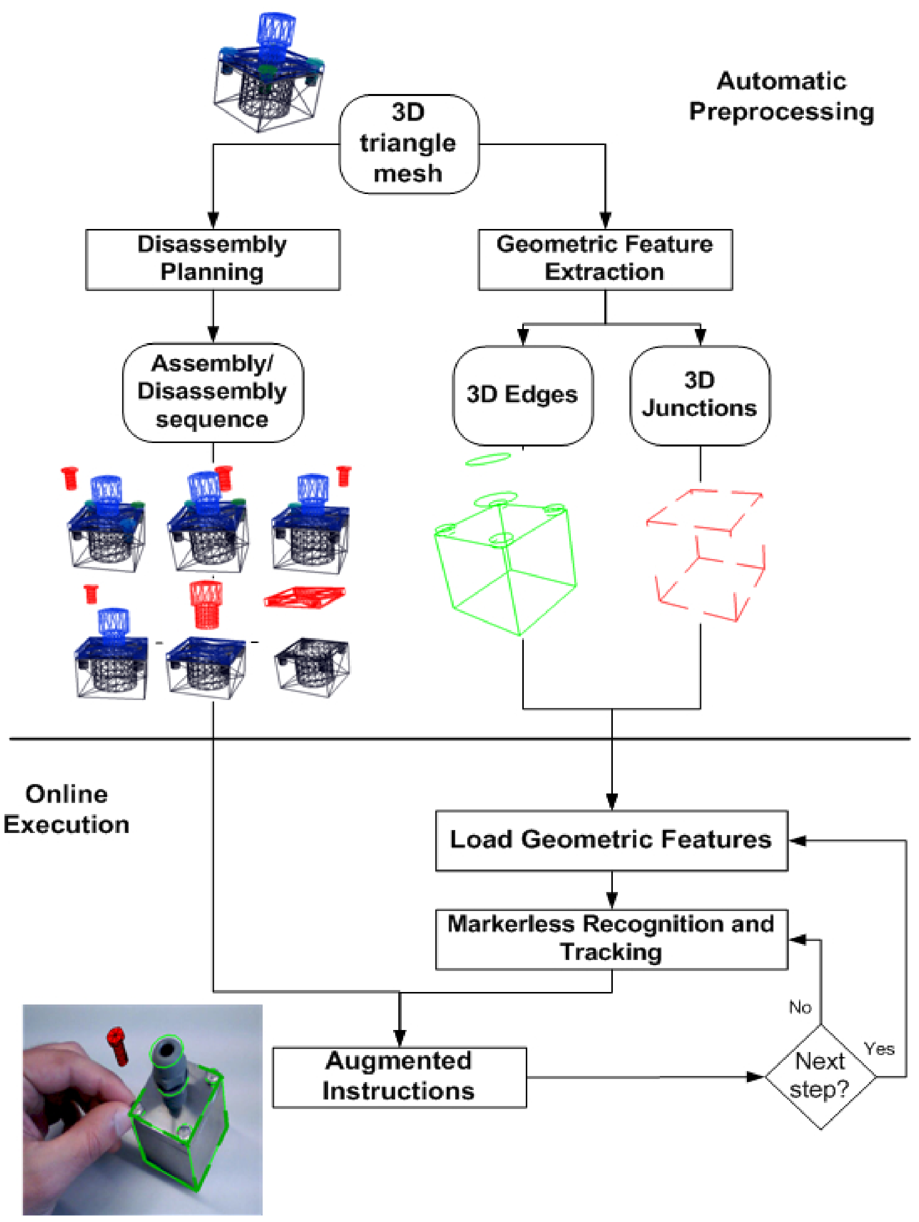
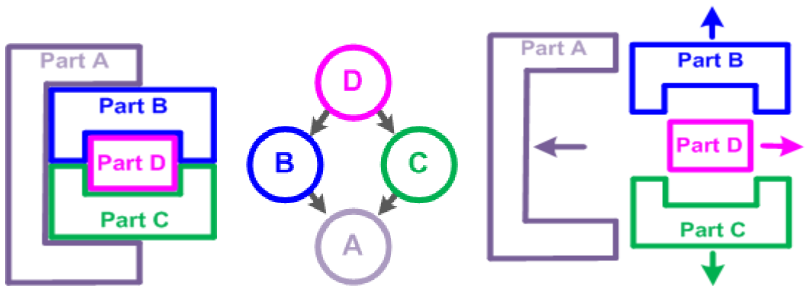


Figure 6.1: Automatic AR Disassembler overview.

6.3.1 Disassembly Planning

The assembly or disassembly procedure for a product can be described by two main components: a precedence graph and the disassembly paths for each component (Latombe, 1999). The precedence graph describes the order of the operations as a graph (Figure 6.2(b)). In this graph each node represents a component and it is connected to a set of other elements whose removal must precede the disassembly of the element represented by it. The second component, the disassembly path (Figure 6.2(c)), represents the motion that is required to extract the component from the assembly.



(a) Model. (b) Precedence graph. (c) Disassembly paths.

Figure 6.2: Disassembly planning procedure.

The initial works on the generation of assembly and disassembly sequences started at the end of the 80's and early 90's (Homem de Mello and Sanderson, 1991). The works on disassembly planning can be classified in two main groups (component based and product based planners) according to the input information used. In component based planning, this information is based on the description of the components of the product by means of its geometry, behaviour, type, etc. The algorithm is usually divided into two sub-problems: the localization of a direction which allows the local translation of the components, and a second step that validates this extraction direction by checking if the generated path is collision free. Some of the first works were based on the recursive partitioning of the assembly into subassemblies (Wilson, 1992).

The second class are product based planners. These use more abstract input describing the product, as the precedence relationships of the disassembly process. Their objective is the generation of optimal sequences.

For this purpose these approaches used optimisation algorithms such as Genetic Algorithms (Marian et al., 2003).

Due to the automatic nature of the methods proposed in this thesis, a component based planner is used here (Aguinaga et al., 2007), whose outcome is the precedence graph and disassembly path for a system. Once this information is computed, the assembly/disassembly sequences can be obtained by traversing the graph from a target component, and removing each other node connected to it. An interest reader can refer to (Aguinaga, 2007) for a more detailed explanation about disassembly planning.

6.3.1.1 Model Format

Input 3D models are composed of a set of parts that need to disassemble. The 3D model of Figure 6.3, for example, is composed by 7 parts: a box, a lid, four small screws, and a cord connector similar to a big screw. No texture information is required as input because it is limited to the geometry of the 3D model. Each part of the model is defined as a simple 3D triangle mesh, and all of them must be provided with respect to the same origin and located in their initial configuration, i.e., as if the model was assembled.

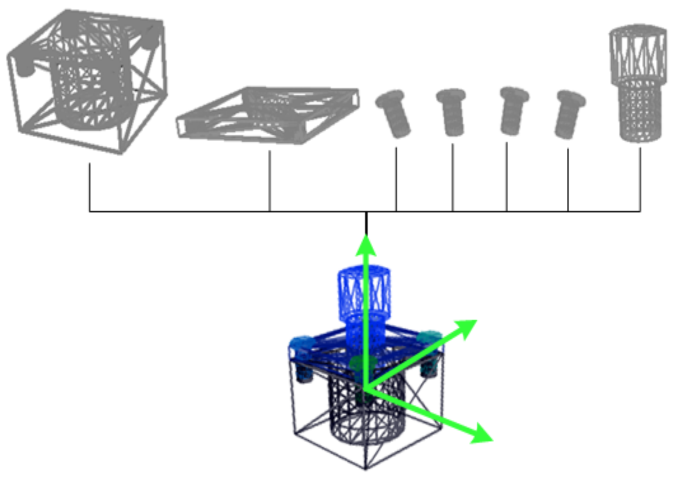


Figure 6.3: Component parts of a model.

It is noteworthy that although CAD software usually allows defining explicitly some relationships (contacts, tolerances, etc.) between the geometric position of the different components of a system, they are not required for the proposed framework.

6.3.1.2 Precedence Graph and Disassembly Path

To obtain the precedence graph and disassembly paths, the algorithm proceeds heuristically to select different components, and then, it tries to generate a disassembly path for them. For most components in industrial settings the extraction path of any component can be represented by a single straight line. The contact information of the different components is used in order to find the set of free local extraction directions. Afterwards, the different directions are tested for collisions in the extraction. Unfortunately, this procedure fails when the extraction path is more complex. In this case, a more robust path planning algorithm based on Rapid Growing Random Trees (RRT) (Aguinaga et al., 2008) is used (Figure 6.4). This approach is more flexible but also more expensive computationally (see Section 6.4).

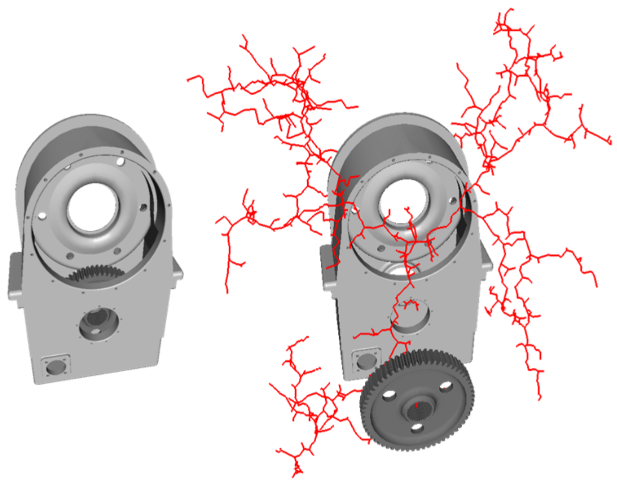


Figure 6.4: Extraction directions using RRT. Image courtesy of (Aguinaga, 2007).

If an extraction path is found for a component, the algorithm finds which components need to be previously removed by testing the removal path with a set of all the components in their initial configuration. Any component that collides with positions in the extraction path needs to be removed prior to it. This way, the complete precedence graph is built in a trial and error process.

Notice that the assembly/disassembly sequence graph is computed only once for each input 3D model. In fact, the entire process is executed in a few seconds during the offline stage. This sequence is used by the Recognition and Tracking module to figure out which is the following scenario to recognize and track.

6.3.2 Recognition and Tracking

The proposed AR disassembler combines the recognition and tracking methods presented in Chapter 4 and 5, respectively. The markerless tracking algorithm is divided into two stages: automatic offline processing and online tracking. In the offline stage, some geometric features, such as edges and junctions, are extracted automatically from the input 3D triangle mesh. This procedure is explained in Section 4.3.2.1, but in this case the models of Table 6.2 are processed. During the online stage, the recognition method (Chapter 4) uses these geometric features to match the input camera image to one of the synthetic views generated during the offline stage. It is used for both initialization and tracking failure recovery. The tracking method (Chapter 5), meanwhile, assumes that the information from the previous frame is available, and combines multiple visual cues to obtain a robust real time tracking under different conditions. The overall markerless tracking scheme is shown in Figure 5.7.

6.3.2.1 Observations

An input 3D model is composed of several components. Therefore, the presence or removal of a part modifies the geometric composition of the model (Figure 6.5), i.e., each assembly/disassembly step detected by the automatic disassembly planning module generates the appearance or disappearance of different edges and junctions. Because of this, each geometric configuration that results from a part removal is treated as a separate 3D model, for which geometric features are extracted. Notice that although multiple disassembly sequences can be considered, only the sequential order set by the disassembly-planning module is used to get the collection of possible geometric configurations. In the same way, during the online execution the next step is known, and it is clear which geometric configuration should be loaded, since the sequence has a predefined order. This method increases the memory requirements, but it offers robust recognition and tracking results.

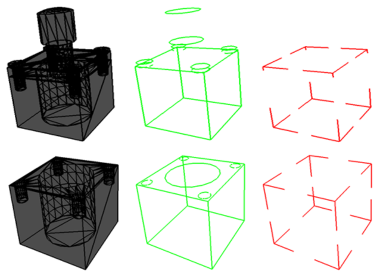


Figure 6.5: Geometric features for different disassembly steps.

During the automatic offline learning stage, the 3D recognition method generates several 2D synthetic views of the 3D geometric features moving a virtual camera along a sphere. Thus, rotation ranges and steps for each axis are parameters that must be set. They can be set by the user or inferred from the disassembly planning module. Since the next part of assembly/disassembly is known, only the views that put this component in front of the camera are trained, ruling out other views. This is not a strict requirement because the user will usually try to focus the camera on this part as an instinctive behaviour. Besides, this action improves the memory requirements and the computational cost, as there will be fewer keyframes to process. Assuming that the next part to process is put in front of the camera (with all the model completely visible and covering the entire image), good results have been obtained processing only those views that are in the $[-45°, 45°]$ range for all axes. Additionally, angle steps that vary from 5 to 8 degrees have been used, which gives a total of 2000-5000 keyframes, depending on the complexity of the model and the maximum storage capacity.

6.4 Experiments and Results

This section presents the results obtained with the proposed AR disassembler. The hardware setup consists of an Intel Core i7-860 at 2.80GHz and 3GB of RAM equipped with a Logitech QuickCam Connect webcam.

In Table 6.1 the execution time of the disassembly planning module is presented for the set of 3D models shown in Figures 6.6, 6.7, 6.8, 6.9 and 6.10 (this time includes the generation of a disassembly graph as well as the disassembly paths and sequences). It is capable of obtaining the disassembly sequence of parts with different geometric properties in a few seconds. The computational cost depends on the complexity and structure of the model. It is noteworthy that it can also find complex disassembly paths using the RRT algorithm (Section 6.3.1.2), as is required for the Gear-box example.

Table 6.1: Execution time for disassembly planning.

Model	Disassembly steps	Time (sec)
Box-1	6	1.5
Box-2	3	0.69
Matryoshka	5	1.6
Gear-box	2	101.660 (RRT)
Elephant	20	6.09

Table 6.2 displays extracted geometric features for the 3D models shown in Figures 6.6, 6.7, 6.8, 6.9 and 6.10. Only one geometric configuration is presented for each 3D model. The number of facets measures the complexity of the extraction procedure.

The response of the 3D recognition module against the geometric configurations processed in Table 6.2 is detailed in Table 6.3. A set of keyframes were taken for each model using a marker tracking system (Wagner and Schmalstieg, 2007). Hence, the real camera pose was known for these keyframes, which was used to obtain the accuracy of the recognition. The automatic training of the 3D recognition module used a range of [-45,45] degrees applied for each axis, starting from the point of view that put the next part to disassemble in front of the camera (Section 6.3.2.1). All test were executed at 640x480 resolution, finding the first camera pose with high accuracy in a few hundreds milliseconds.

In Figures 6.6, 6.7, 6.8, 6.9 and 6.10, some snapshots of augmented instructions that the proposed system provides are displayed for a set of 3D models. Instructions are in red, while the 3D model that is being tracked is highlighted in green. This set of models covers multiple geometric configurations, different types of surfaces, as well as disassembly steps of varying complexity. Figure 6.6 shows the good response of the system

Table 6.2: Execution time for geometric feature extraction.

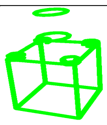
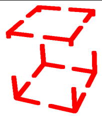
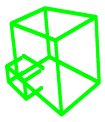
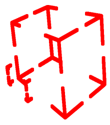
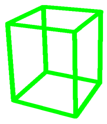
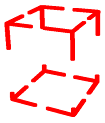
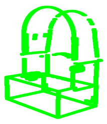
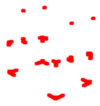
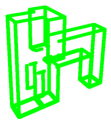
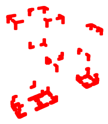
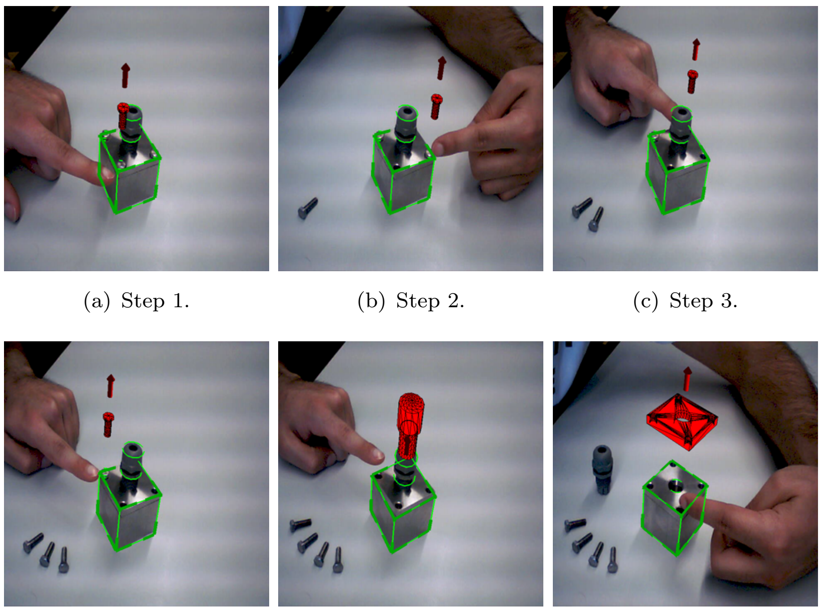
	Facets	Edges	L Junctions	Time (sec)
Box-1	3102	 118	 24	1.43
Box-2 (step 3)	40	 21	 32	1.33
Matryoshka	208	 12	 24	1.47
Gear-box	59658	 172	 59	26.84
Elephant	9084	 72	 80	3.65

Table 6.3: Accuracy and execution time for recognition.

Model	Mean Reprojection Error (pixels)	Time (ms)
Box-1	4.26	99.66
Box-2 (step 3)	4.47	63.16
Matryoshka	4.55	107.52
Gear-box	6.57	155.03
Elephant	4.72	356.05

against models with specular surfaces. In addition, Figure 6.7(c) displays a satisfactory output when small parts are tracked. Figure 6.8 demonstrates the robustness of the system for scenes where multiple parts with similar geometry are put in front of the camera. In fact, this also proves that the system will not be jeopardized by the tools that the worker places in the task-space, provided that objects and tools do not have the same shape.

Likewise, although the recognition and tracking of Box-2 and Matryoshka models is hampered by their textureless and homogeneous outer surface, the results are satisfactory. Figure 6.9 indicates that the proposed tracking can handle models with high complexity, combining both rectilinear and curvilinear edges. Indeed, the extraction path of the gear is composed of several movements in multiple directions, although only the direction of last extraction step is shown in Figure 6.9. Finally, Figure 6.10 proves the ability of the system to disassemble models composed of many parts, as well as, tracking small geometric configurations.

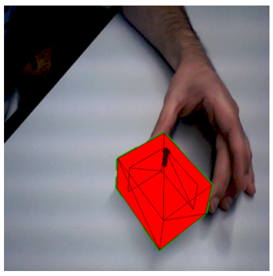

(a) Step 1. (b) Step 2. (c) Step 3.

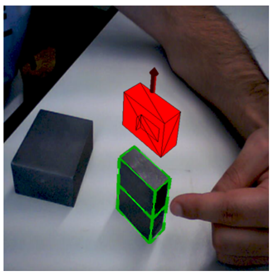
(d) Step 4. (e) Step 5. (f) Step 6.

Figure 6.6: Disassembly of the Box-1 model.

6.5 Usability Analysis

AR has demonstrated its validity for educational purposes (Juan et al., 2010). This way, this section evaluates the effectiveness of the proposed AR system for disassembly tasks. It is based on the idea that AR is beneficial for users, since they can see simultaneously both instructions and the real


(a) Step 1.


(b) Step 2.

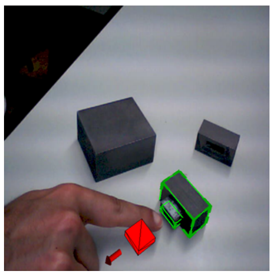
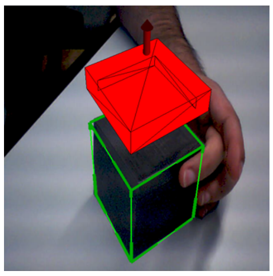
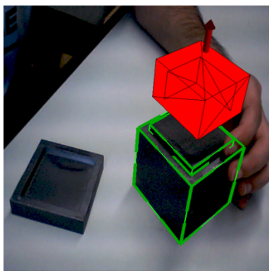
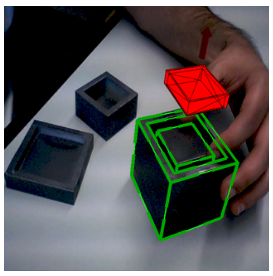
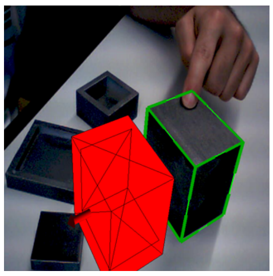

(c) Step 3.

Figure 6.7: Disassembly of the Box-2 model.


(a) Step 1.


(b) Step 2.


(c) Step 3.


(d) Step 4.

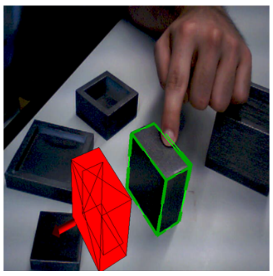
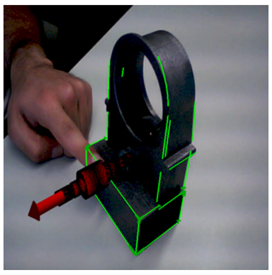

(e) Step 5.

Figure 6.8: Disassembly of the Matryoshka model.

world, i.e., they do not have to constantly change their attention from the instructional media to the task (Baird and Barfield, 1999). The following paragraphs describe the experimental procedure that has been designed to compare AR instructions with the traditional method (documentation based on paper).


(a) Step 1.

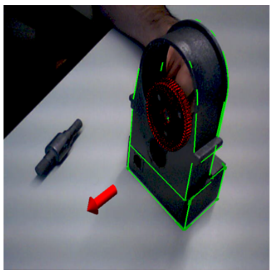
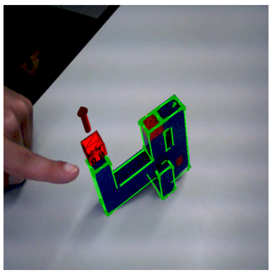
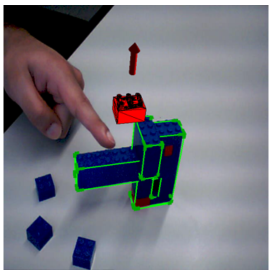
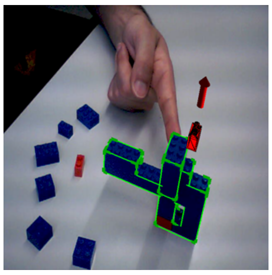
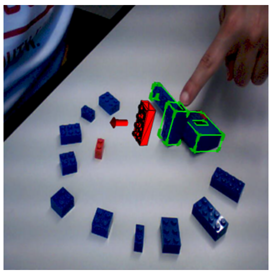

(b) Step 2.

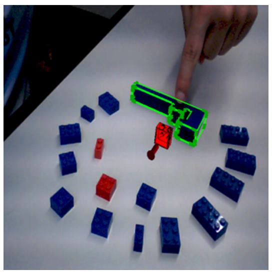
Figure 6.9: Disassembly of the Gear-box model.


(a) Step 1.


(b) Step 4.


(c) Step 8.


(d) Step 12.


(e) Step 16.

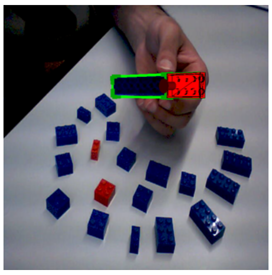

(f) Step 20.

Figure 6.10: Disassembly of the Elephant model.

6.5.1 Subjects

20 participants were asked to perform the experiment, which lasted approximately 10-15 minutes. 14 subjects were male and 6 were female, aged from 23 to 36 years (mean=27.55, stdev=3.55) (top of Figure 6.11).

They did not received any payment for the participation, and subjects were required to respond a questionnaire prior to the experiment to determine their profile. This questionnaire contained the following statements, which had to be answered on a scale from 1 (none/beginner) to 5 (assiduous use/expert):

QP1 Rate your familiarity with computer-assisted disassembly systems

QP2 Rate your familiarity with AR systems

QP3 Rate your skill level in performing assembly/disassembly tasks

Analysing the bottom of Figure 6.11, most of the participants did not have too much experience with disassembly tasks (with and without computer assistance, mean=2.15, stdev=1.39 and mean=2.55, stdev=1.36 respectively) or AR technology (mean=2.65, stdev=1.23). More precisely, 13 subjects (65% of total) reported low familiarity with computer-assisted disassembly tasks (QP1<=2), 16 subjects (80% of total) reported a medium-low familiarity with AR (QP2<=3), and 14 subjects (70% of total) reported a medium-low skill level with disassembly tasks (QP3<=3).

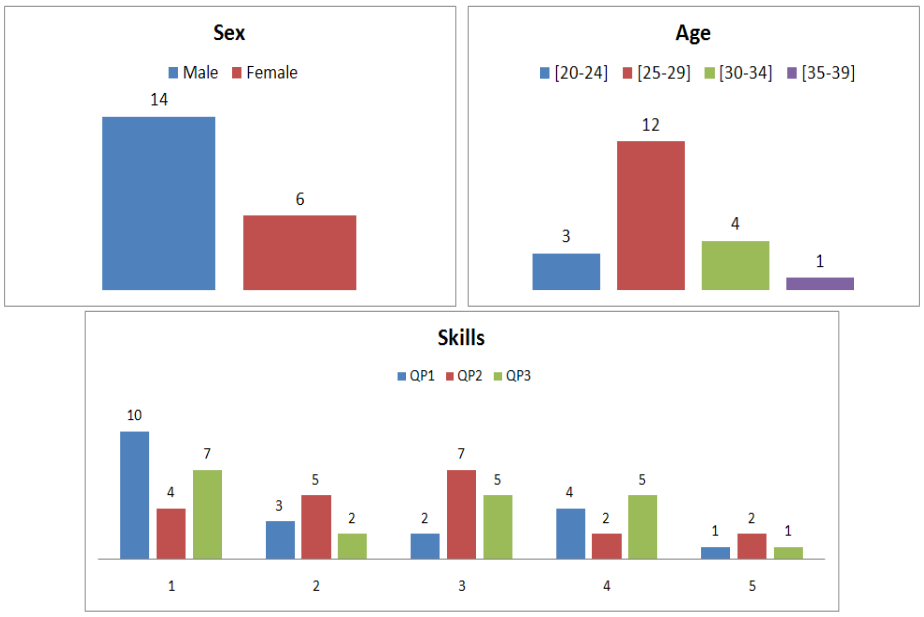


Figure 6.11: Participants profile.

6.5.2 Task

The experimental task in this study consists of disassembling some parts of the Lego Elephant presented in Figure 6.12. There were 7 Lego bricks that users had to disassemble with their hands. Since this was a simple task, users were not informed in advance about what parts had to be disassembled, they discovered the next step during the execution (*just in time*). This complication was intended to make more difficult the disassembly task, requiring the use of instructions. It was considered enough for the purposes of this experiment.

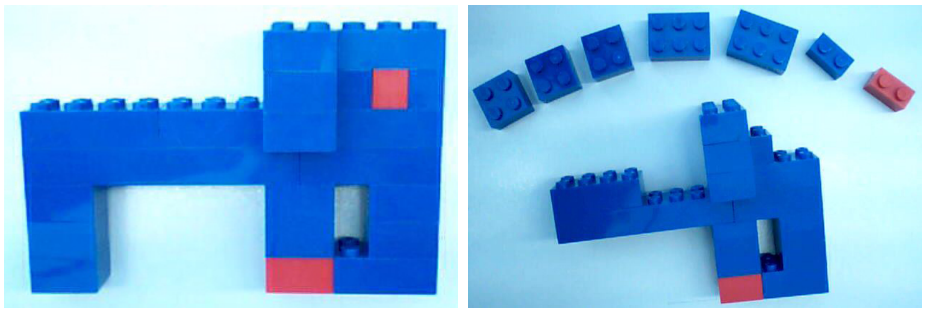


Figure 6.12: Lego Elephant before and after disassembly.

Two different types of instructional media were used to perform the task: paper instructions and computer-aided instructions (AR instructions).

6.5.3 Experimental Design

Two types of designs are used to distribute the subjects of an experiment: *between-subjects* and *intra-subjects*. Between-subjects distributes participants in several groups and each group only executes a part of the experiment, while intra-subject assigns all participants to the same group, who execute the whole experiment. Compared to between-subjects, intra-subject design avoids the appearance of elite groups, i.e. the risk of imbalance distribution of the participants along the groups according to their ability (Otero and Dolado, 2000). However, intra-subjects design suffers from the *learning effect* due to the recurrence of the same participants in all parts of the experiment.

Intra-subjects design was used in this experiment. Furthermore, in order to avoid distortions in the results of this experiment, participants executed all parts of the experiment on a rotating basis: half of the participants executed the experiment in the order *paper-AR* and the other half in the order *AR-paper*, where *paper* indicates that paper instructions were used and *AR* indicates that computer-aided instructions were used.

Additionally, two different disassembly possibilities were prepared (*T1* and *T2*). Both obtained the same result, but the parts were disassembled in a different order. This way, T1 was used for the first execution, while T2 was chosen for the second one, regardless of the instructional media. This resulted in two main options: *paper(T1)-AR(T2)* or *AR(T1)-paper(T2)*. This also helped to minimize the learning effect.

The independent variable for the experiment was the type of media used to display the disassembly instructions (paper or computer-aided). The dependent variables were the time to complete the disassembly task, the number and type of errors during the disassembly task, and responses to a usability questionnaire.

6.5.4 Procedure and Equipment

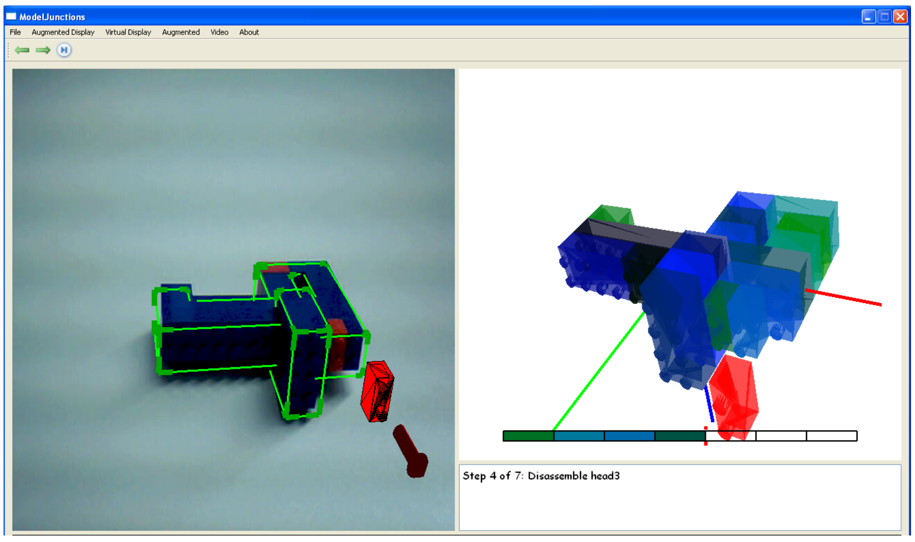
As mentioned above, the subjects had to disassemble the 7 parts of the Lego Elephant presented in Figure 6.12, using two different instructional media: paper and computer-aided instructions. Moreover, the execution order was controlled to minimize the learning effect (*paper(T1)-AR(T2)* or *AR(T1)-paper(T2)*). The subjects did not receive a training session on how to disassemble parts of a Lego object.

For the paper instructions, subjects were given a manual 5 pages long (A4 size), which contained a detailed explanation and a graphic representation of each disassembly step (see Appendix D). The experimenter only indicated that the document had to be read carefully, as it contained all the necessary information.

For the computer-aided instructions, subjects used the AR system that has been described along this chapter, which offers virtual instructions that explain how to perform the disassembly operation (Figure 6.13). The experimenter gave a brief explanation (2-3 minutes) on how to use this tool, indicating that the system recognizes the object that is in front of camera and renders its virtual information in the corresponding image position, i.e., it achieves a perfect alignment between real and virtual objects. The subjects were told that the system overlays a virtual representation of the disassembly operation in the image (left panel of Figure 6.13), indicating the next part to disassemble (coloured in red) and which is the extraction path to follow (emphasized with a red arrow). Additionally, the experimenter explained that there was a virtual panel with the 3D model with which to interact (using the mouse) and have multiple points of view of the disassembly operation (right panel of Figure 6.13). Indeed, the experimenter pointed that they could use keys '$\leftarrow$' and '$\rightarrow$' to navigate between different disassembly steps:

Next step ($\rightarrow$): the system shows the next part to disassemble. If there are not more steps to perform, it displays a message indicating that the task has been finished.

Previous step ($\leftarrow$): the system returns to the previous part of the disassembly operation.



Step 4 of 7: Disassemble head3

Figure 6.13: Snapshot of the AR system.

After finishing the explanation, subjects were invited to simulate a disassembly step using the AR system (familiarization phase). The experimenter did not use the Elephant model to teach how to handle the AR system, but the model of Figure 6.6. An Intel Core i7-860 at 2.80GHz and 3GB of RAM equipped with a Logitech QuickCam Connect webcam was used in collaboration with a simple monitor (22 inches) to offer AR instructions.

As well as timing subjects, the experimenter registered the errors made by the subjects during the experiment. However, due to the clarity of the instructions and the difficulty level of the task, there were no errors.

After finishing the experiment, subjects completed a questionnaire to determine their subjective opinion about the instruction media and the usability of that media. This questionnaire contained the following questions, which had to be answered on a scale from 1 (not very easy) to 5 (very easy):

AR instructions

- **Q1** How easy was it to use?
- **Q2** How helpful was it to solve the disassembly task?
- **Q3** How easy was it to interact with?
- **Q4** How enjoyable was it to use?

Paper instructions

- **Q1** How easy was it to use?
- **Q2** How helpful was it to solve the disassembly task?
- **Q3** How easy was it to interact with?
- **Q4** How enjoyable was it to use?

Q5 What type of instructions do you prefer? (from 1: AR; to 5: Paper) Why?

6.5.5 Results

Performance Times

The mean execution time of paper instructions was 108.4 seconds (stdev=38.43), while the mean of AR instructions was 135 seconds (stdev=47.98). Noting this difference, a paired t-test with a significance level of 0.05 was applied to the execution times, which confirmed that there was a statistical difference (t(19)=2.36, p=0.03). The main reason of this difference is that the subjects did not read the instructions. The disassembly task was intuitive, so that they just watched the graphical representations.

Figure 6.14 shows the execution time (in seconds) for each subject and instructional media. The first 10 measures correspond to the *AR(T1)-paper(T2)* order, while the last 10 measures correspond to the *paper(T1)-AR(T2)* order. It can be observed that the paper execution time of the first 10 measures (mean=85.6, stdev=22.23) was low compared to the last 10 measures (mean=131.2, stdev=38.32). Indeed, an Analysis of Variance (ANOVA) procedure was applied to investigate the potential effect of execution order on the paper execution time. The results of the ANOVA indicated that there was statistically significant effect of execution order on the paper execution time (F=10.6, p=0.004). Based on this, one could argue that AR instructions provide a better understanding of the task, allowing users to extrapolate the acquired skills more efficiently to other settings (paper instructions).

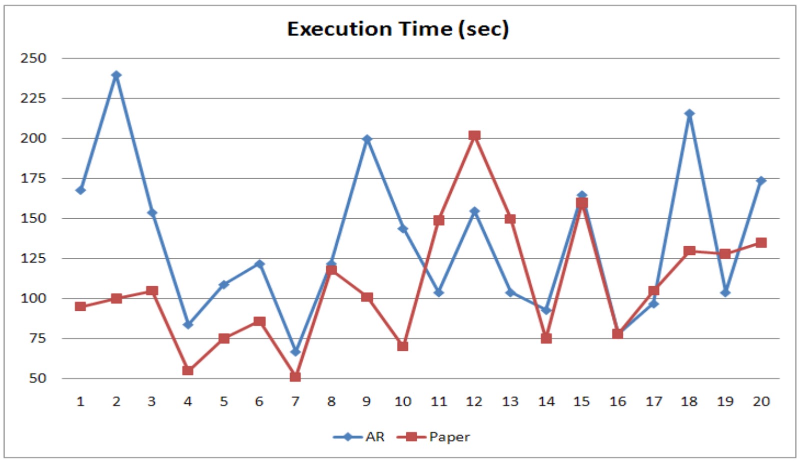


Figure 6.14: Exec. time for each subject and instructional media.

Usability

Figure 6.15 shows the number of subjects for each response of the usability questionnaire, grouped by question number (Q1-Q4) and instructional media. Moreover, Table 6.4 displays descriptive statistics of the responses according to the question number and instructional media.

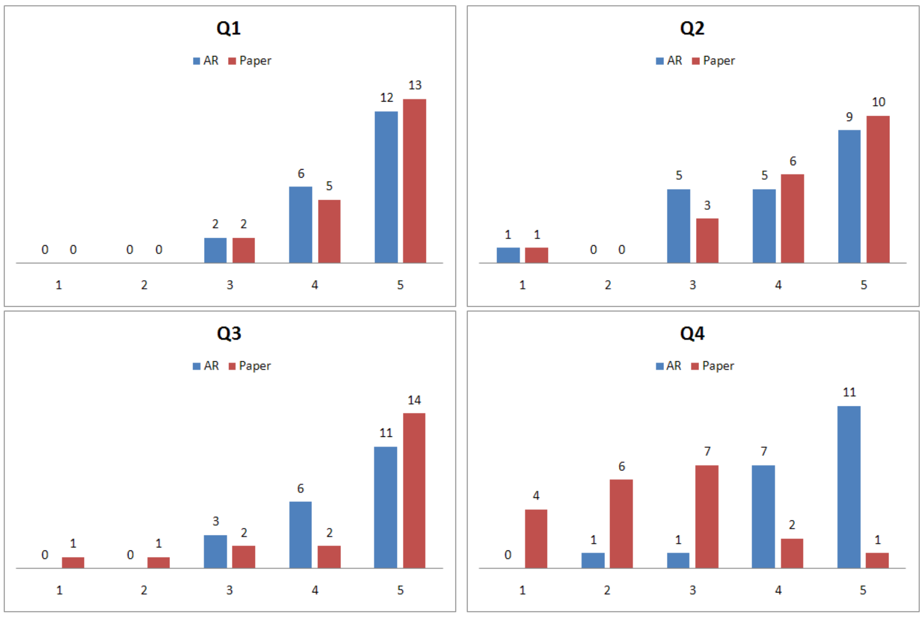


Figure 6.15: Responses for the usability questionnaire.

Table 6.4: Statistics (mean(stdev)) for the usability questionnaire.

	Q1	Q2	Q3	Q4
AR	4.5 (0.69)	4.05 (1.1)	4.4 (0.75)	4.4 (0.82)
Paper	4.55 (0.69)	4.2 (1.06)	4.35 (1.18)	2.5 (1.1)

It should be noted the high scores that most of the users gave to the usability of both instructional media. They found easy to use (Q1), very helpful (Q2), and easy to interact with (Q3). The main difference was the sense of enjoyment (Q4) experimented by the subjects. For most subjects AR instructions were more motivating.

The questionnaire also included a question on which subjects had to specify and argue the preferred instructional media (Q5). Figure 6.16 displays the number of subjects for each preferred option. It is clear that most of the subjects chose AR instructions to perform disassembly tasks. Indeed, subjects who did not choose AR instructions argued that it was too simple task to take advantage of AR technology. They indicated that with more complex tasks they would also have chosen AR instructions.

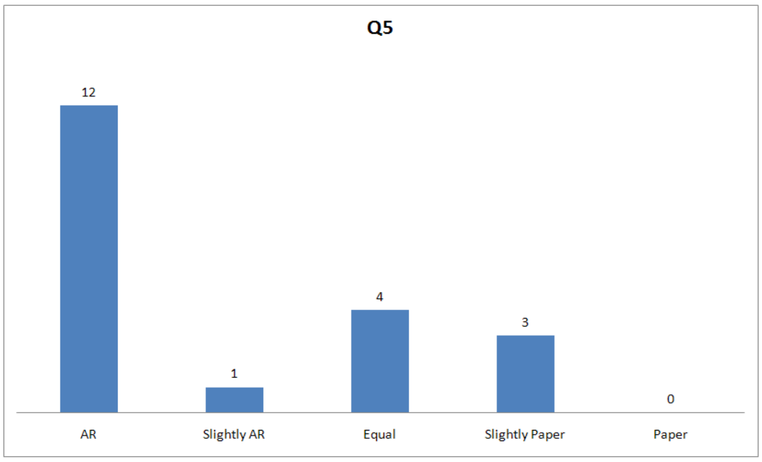


Figure 6.16: Preferences of subjects regarding instructional media.

6.6 Discussion

In this chapter an automatic AR disassembler oriented to maintenance and repair operations has been described. The novelty of the proposed system relies on its ability of being a complete framework that is self supplied. All the data is extracted automatically from a collection of 3D parts that are provided as untextured 3D triangle meshes.

An existing VR disassembly planning module has been adapted to the proposed AR framework to compute the assembly/disassembly sequence, which is obtained finding collision-free trajectories. Moreover, the 3D recognition method presented in Chapter 4 has been used to retrieve the first camera pose of the input untextured 3D model and for recovery in case of tracking failure. It is based on the geometric properties of the model, so the proposed system can deal with scenes that lack texture. Similarly, the markerless tracking method detailed in Chapter 5 has been used to perform the tracking. It offers a robust and real time 3D tracking against multiple and undesirable conditions, so the proposed AR disassembler is a complete framework that runs in real-time and tries to facilitate the work of workers, replacing the paper-based documentation.

The information computed by the disassembly planning is used to guide the training process of the recognition and tracking methods. More precisely, since the next part to assembly/disassembly is known, then the recognition and tracking methods are trained to identify that piece as if it was in front of the camera. This way, the proposed framework is not a set of independent modules, but an AR system that combines multiple methods in an intelligent way.

An experiment that validates the usability of the proposed AR system has been also presented. Although the limited sample size (20 subjects) makes difficult to perform rigorous statistical analysis, the results indicate that the proposed AR system help users understanding the disassembly task. In fact, it was found that the use of AR stimulates users to perform the task. However, due to the simplicity of the disassembly task, most of the subjects performed the task faster with paper documentation because they did not read instructions.

Finally several aspects of this chapter requires a deeper study in the future, such as a visual inspection that ensures the correctness of the assembly/disassembly task or a more complete experiments (with more subjects and with a more complex disassembly task) that validate the usability of the system.

Part III

Conclusions

Chapter 7

Conclusions and future work

Inevitably the machines must win,
but there is still a long way to go
before a human on his or her best day
is unable to defeat the best computer
Garry Kasparov

7.1 Conclusions

This thesis studies several augmented reality tracking techniques for industrial environments. More precisely, real time monocular optical tracking methods are investigated, since these solutions increase the effectiveness of AR (real time) using a single camera (monocular) and a limited budget (optical sensors).

The main contributions of this work are focused on the design and development of recognition and tracking methods that offer a robust response when dealing with the untextured scenes of industrial environments, as well as building an AR system for guidance in disassembly tasks to validate the quality and capacity of the proposed techniques.

Optical tracking methods that are based on markers fail even if the marker is only slightly occluded. Furthermore, due to the fact that occlusions caused by the hands of the worker and tools are common in industrial environments, two new methods have been developed to address this shortcoming: Occlusion-OBB and Occlusion-Patches. Both use computer vision techniques and avoid using multiple markers to obtain the camera pose, i.e., environment adaptation has not been jeopardized in favour of robustness against occlusions. Different computation capabilities and requirements (6 DOF vs 4 DOF) have also been considered when designing these solutions. The main contributions and conclusions obtained for each of these proposals are:

1. **Occlusion-OBB:** This is a method that uses temporal coherence assumptions and simple image processing to track the bounding box of the marker and update the camera pose. Due to the popularity of mobile devices, it has been oriented towards mobile platforms. Indeed, as these devices have limited processing and memory capabilities, Occlusion-OBB assumes some simplifications so that only 4 DOF of the camera are updated (translation in 3 axes and rotation in the axis that is perpendicular to the image plane). It is a valid solution for some tasks, such as advertising applications. Additionally, it can be used in scenes that are already prepared for parallelogram markers (ARToolkitPlus, for example) without having to install anything since it does not need to prepare an environment with more artificial landmarks.

2. **Occlusion-Patches:** This is a new marker design that places customizable texture patches along the frame of the marker to have more visible features (with known 3D coordinates) during marker occlusion, which facilitates the computation of the camera pose (6 DOF). This way, it can be adapted to any marker tracking system that uses its central area to codify digital identification (ARToolkitPlus, for example). These textures are customizable, which lets users make their own designs or address marketing purposes. This approach combines two different tracking techniques (frame to frame tracking and tracking by detection), whose computational cost is considerable compared to Occlusion-OBB. The first one measures 2D displacements of features using Lukas and Kanade optical flow and a simplified version of the SIFT descriptor to calculate the camera pose. The second approach, meanwhile, is based on appearance, so that the image's 2D features are matched with their corresponding 3D features using a simplified version of SIFT descriptors. In addition, two novel human-machine interfaces have been introduced to show the new possibilities that have arisen as a result of obtaining more information during marker occlusions.

Sometimes is not possible to modify the environment, so marker-based alternatives cannot be used. To solve this problem markerless tracking methods take advantage of the visual cues that are naturally in the scene. Some of these solutions store knowledge about the scene in a 3D model, which is available before the camera tracking begins. The 3D model can be represented by its geometry and texture, but most objects that are in industrial environments lack texture, which jeopardizes the tracking task. In order to track untextured 3D models, 3D object recognition and markerless tracking methods based on geometric properties have been developed. The first method initializes the tracking, while the second one updates the camera pose using frame to frame assumptions:

1. **3D object recognition:** This method processes the untextured 3D model to automatically extract its geometric features, such as 3D sharp edges and 3D L junctions. Several 2D synthetic views of these 3D geometric features are taken during the automatic training phase, which are matched to those features with similar geometric description detected in the current camera image. An efficient geometric constraint based on a pair of junctions is proposed to get camera pose candidates, while edges are used to obtain the similarity measure. A parameterisation study has also been presented, providing a configuration that works fine in most scenarios.

2. **Markerless tracking:** Self-supplied and robust markerless tracking that combines an edge tracker, a feature tracker and a particle filter has been developed to continuously update the camera pose. All these techniques have not been used independently, but rather they have been integrated to exploit the advantages of each one. The feature tracker and the particle filter offer a good initial guess of the correct camera pose, even if rapid camera movements are applied, while the edge tracker calculates the precise pose. The particle filter is used as an alternative to the feature tracker in the absence of features. It combines edges, junctions and points to obtain stable results.

Although each proposed method has been validated with its own set of experiments, an AR disassembler tool has been developed to show the real world applicability of the methods proposed throughout this thesis. This tool substitutes paper documentation, and provides workers with augmented instructions in order to perform maintenance tasks. The novelty of the proposed system relies on its ability to be a real time and complete framework that is self-supplied. All the data is extracted automatically from a collection of 3D parts that are provided as untextured 3D triangle meshes. Additionally, an existing VR disassembly planning module has been adapted to this AR system, which computes assembly/disassembly sequences by finding collision-free trajectories. The 3D recognition and markerless tracking methods mentioned earlier are responsible for retrieving the camera pose. Similar to the integration procedure explained above, all these methods have been combined in an intelligent way, taking advantage of the disassembly order to guide the training of the recognition and tracking methods. The usability of this AR system has been also demonstrated through a set of user experiments.

In summary, several real time tracking methods have been proposed to offer robust tracking in industrial environments. Both marker and markerless tracking methods have been presented to fulfil multiple requirements. An AR disassembly tool has been developed to demonstrate its real applicability. Furthermore, the proposed solutions do not need expensive hardware, as they use a single low cost camera to capture images and a standard PC to process them, i.e., they have a wide application domain.

7.2 Future research lines

Several research lines have been identified in this thesis, which can guide future studies. These ideas are presented below according to the part they belong to:

- **Occlusion-Patches:** The accuracy and robustness of the camera pose could be improved if edges are combined with features. The tracking method presented here only uses features, which are robust against rapid camera movements. However, edges give more accurate results, and are stable even with lighting changes.

- **3D object recognition:** This method recognizes an object using its geometric properties. It is optimised for untextured 3D models, so its response can be jeopardized if the model has a textured surface. Thus, it would be interesting to incorporate appearance data, such as texture or colour patterns, to the recognition process. This would decrease false positives and would extend the application domain.

Additionally, other edge similarities should be tested to validate the success of recognition. Currently edge similarity is used (Steger, 2002), but (Hinterstoisser et al., 2010) has recently proposed an efficient technique for recognizing objects using dominant gradient orientations. This method requires real images of the target object to perform the training (manual training), but it is capable of building online templates once the object has been detected. Hence, a system that incorporates this technique will be capable of dynamically learning the contours of the object.

- **Markerless tracking:** The proposed markerless tracking combines multiple tracking techniques (edge tracker, feature tracker and particle filter), which are configured to obtain good results satisfying real time constraints. Moreover, all of them have been implemented using only the CPU, wasting the computing capabilities of the GPU. This way, a GPU implementation of some of these techniques, similar to the GPU particle filter detailed in (Klein and Murray, 2006) or the GPU feature tracking presented in (Sinha et al., 2006), would balance the computational effort between the CPU and GPU. This would save computational time, and more sophisticated techniques could be applied, such as a complete version of the SIFT descriptor (Lowe, 2004) or a real time SLAM technique (Sánchez, 2010). In fact, SLAM is a technique that makes a 3D reconstruction of the scene as well as performs the camera tracking, i.e., it is capable of learning new environment conditions during the online execution so that occlusion or disappearance of the target model is supported.

- **AR Disassembler:** The AR tool that has been presented guides workers in disassembly operations, but it does not verify the correctness of the task. Therefore, a feedback module should be incorporated to determine whether the user performs the disassembly task correctly, one that explain how to proceed in the case of an error. This improves efficiency and is helpful for users (Ryoo et al., 2010).

  The user is responsible for notifying the system that the task is completed, for which a simple key is pressed. This is uncomfortable if his hands are busy, so it should also be controlled by gesture recognition or a voice command. It is noteworthy that the proposed

AR system uses the solution of a simple key press because it is a prototype that serves as *proof of concept.*

As explained in (Hakkarainen et al., 2008), a client-server architecture should be implemented to extend the application domain. Using this architecture, complex calculations are executed on a PC (a server), while the results are displayed on the screen of a remote device (the client). This eliminates the need to move bulky hardware components since a thin client such as a mobile device can access the server results.

An experiment that validates the usability of the proposed AR system has been described (Section 6.5). However, more complete experiments (with more subjects and with a more complex disassembly task) should be done, applying robust statistical analysis. For example, several experiments have demonstrated the effectiveness of AR for assistance in maintenance and repair operations in terms of operation efficiency (Raghavan et al., 1999; Tang et al., 2003), while the experiment that has been presented here did not obtain the same conclusion due to the low difficulty of the proposed disassembly task. The results of more complete experiments would improve the system and elicit suggestions about perceived tracking deficiencies as well as the best way to display augmented instructions.

The SLAM techniques have received much attention in recent years. They do not need environment adaptation and they automatically compute the 3D model of the scene, without providing it as an input parameter. They are interesting solutions, and their computational time decreases every year due to advances in hardware capabilities. Because of that, the study of real time markerless tracking methods oriented toward industrial environments should be extended to SLAM techniques as well.

Part IV

Appendices

Appendix A

Real time SIFT

The following paragraphs present the simplifications that have been applied to the original SIFT method (Lowe, 2004), which have been inspired in (Wagner et al., 2008; Wagner et al., 2010). The resulting approach has been called *simplified-SIFT* and is executed in real time.

A.1 Simplified SIFT

SIFT (Scale Invariant Feature Transform) is a method that locates and describes the features that are in an image. The Difference of Gaussians (DOG) operator is the responsible of extracting features from an image. It provides high repeatability and scale invariance, but has high computational cost. Because of that, it has been replaced by the FAST operator (Rosten and Drummond, 2006), which offers a good balance between repeatability and computational cost. As a counterpart, FAST is not invariant to scale, so the robustness is jeopardized.

The original SIFT descriptor is parametrized by n and b, where $n * n$ is the number of subregions in which the local image area of a feature (denoted by p) is divided. b is the number of bins that are used to build a histogram that describes the image gradients of each subregion. The size of p is estimated by DOG, which approximates the scale of each feature. Simplified-SIFT, however, uses FAST operator, which is not able to estimate the size of p. Thus, simplified-SIFT descriptor is parametrized by n, b and s, where s is the size of the local image patch used to build the feature descriptor. This value is not adapted dynamically, as it is for DOG, but is set by the user. Reasonable values of s belong to the interval

[15..31]. Moreover, s is the same for all features and for all executions (fixed patch size), so some calculus are precomputed to save time during the online execution. More precisely, as SIFT descriptor rotates p according to its dominant orientation (invariance to rotation), the cost of several mathematical operations is saved by precomputing all possible rotations of p.

Knowing that the descriptor of each feature is calculated independently, simplified-SIFT uses parallel techniques. It is based on the SIMD (Single Instruction, Multiple Data) model, where multiple processing units execute the same instruction (feature descriptor computation) sent by the control unit, but with different sets of data (different image patches). The instruction is executed in parallel, but synchronously.

Furthermore, although SIFT replicates the descriptors of those features that do not have strong dominant orientation (multiple orientations with similar weight), simplified-SIFT only retains the highest dominant orientation, which reduces the matching quality but facilitates the parallel computation.

The main differences between SIFT and simplified-SIFT are summarized in Table A.1.

	SIFT	simplified-SIFT
Feature Detector	DOG	FAST
Scale Invariance	Yes	No
Patch Size	Dynamic	Fixed
Dominant Orientation	Multiple	Single
Execution	Sequential	Parallel
Real time	No	Yes

Table A.1: Main differences between SIFT and simplified-SIFT.

A.1.1 Scale invariance

Simplified-SIFT replaces DOG by FAST so that the invariance against scale is lost. Therefore, the content of a feature descriptor may change for two images taken from two different scales, which decreases the matching quality.

This is not a serious problem for some real time tracking approaches. In some cases simplified-SIFT is used to estimated the new location of those features detected in the previous frame. Moreover, in these examples each feature descriptor is updated every frame, so the temporal coherence and real time properties ensure that the scale change between two consecutive frames will not suffer too much change. In these situations the content of a feature descriptor will be similar for two consecutive frames and the matching quality will remain at good levels.

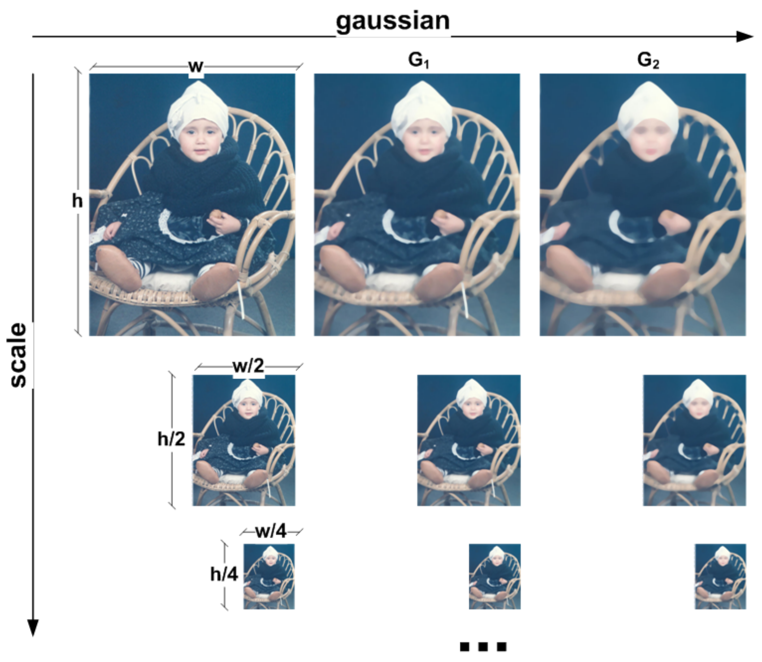


Figure A.1: Image training of simplified-SIFT to avoid scale ambiguity.

SIFT is also used for object recognition tasks. Features of objects are extracted from a set of reference images and stored in a database (*reference features*). Thus, given a new image, those features extracted in the new image are matched to the reference features. The similarity between descriptors is used to get correspondences and an object is recognized if enough positive matches are found. This way, the scale invariance is an important property for this type of approaches. Nevertheless, simplified-SIFT can still be used in such applications by doing a more extensive preprocessing of the set of reference images. Following this reasoning, the reference images are trained along multiple scales and Gaussian filters (Figure A.1), which results in a scale-space technique similar to (Lowe, 2004). This procedure increases storage requirements, as one descriptor is stored for each feature and scale, but it covers the expected ranges at runtime so that the scale ambiguity problem is solved.

A.1.2 Computational Time

Simplified-SIFT has been executed with different parameterisations and hardware configurations (Figure A.2). Two different PCs have been used: an Intel Core 2-Duo at 2.40GHz and 2GB of RAM (Figure A.2(a)), and an Intel Core i7-860 at 2.80GHz and 3GB of RAM (Figure A.2(b)). s has been set to [15,21,25,31] values, $n = 4$, and $b = 8$. The number of features has been increased in ascending order, from 100 to 1000 with a step of 100. The execution time includes both feature extraction (FAST) and the computation of feature descriptors. Moreover, Table A.2 shows the mean execution time of simplified-SIFT for different number of features. All the experiments have been executed at 640x480 resolution.

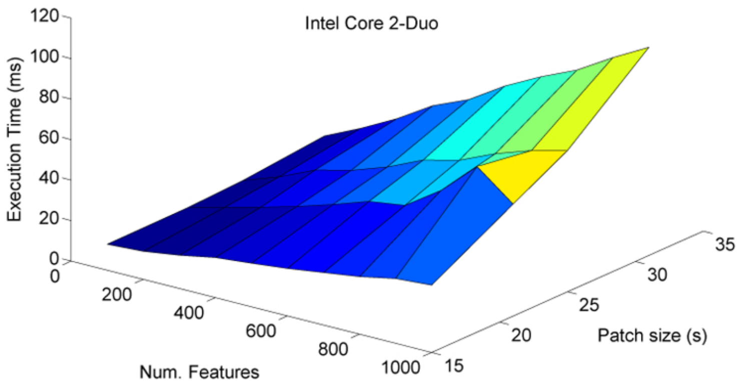
	Num. Features				
	100	300	500	700	1000
Core 2-Duo	14.7	25.75	36.88	48.29	65.29
Core i7-860	10.99	15.36	19.88	24.82	32.31

Table A.2: Mean execution time (ms) of simplified-SIFT.

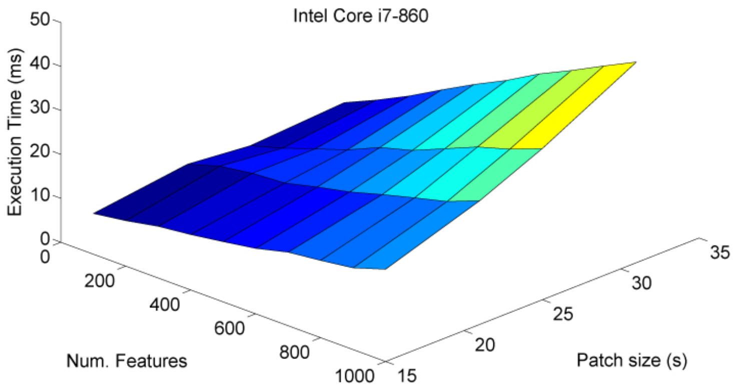
Analysing these values, Intel Core i7-860 can be used with any configuration, as it fulfils the real time requirement ($\sim$ 33 ms). Intel Core 2-Duo, meanwhile, should be limited to 500 features to maintain real time limits, which is already a high value.

The original SIFT takes around 500 ms to process a 640x480 image with 500 features, even using an Intel Core i7-860 at 2.80GHz and 3GB of RAM. Although a non-optimised implementation has been used[1], the difference is remarkable. Notice, however, that the execution of SIFT includes scale invariance. Additionally, a GPU implementation of SIFT is also available[2],

[1]An open-source SIFT library written in C: *http://blogs.oregonstate.edu/hess/code/sift/*.
[2]SiftGPU: *http://cs.unc.edu/ ccwu/siftgpu/*.



(a) Core 2-Duo.



(b) Core i7-860.

Figure A.2: Execution time (ms) of simplified-SIFT.

which computes $\sim$ 4500 points in less than 100 ms for an image at 640x480 resolution. This is a great amount of points, but the computational time exceeds real time requirements.

Appendix B

Pose from a 3D Plane

The following sections describe how to obtain the camera pose from a planar 3D structure when the camera intrinsic parameters are known. A more detailed explanation about this procedure can be found in (Hartley and Zisserman, 2004). In fact, the pose ambiguities[1] derived from the procedure of extracting the camera pose for planar targets are detailed in (Schweighofer and Pinz, 2006), but they are not presented here.

B.1 The Direct Linear Transformation (DLT)

The basic Direct Linear Transformation (DLT) algorithm is a simple linear algorithm that determines H given a set of 2D to 2D point correspondences ($\vec{m}_i \leftrightarrow \vec{m}'_i$), where H is a 3x3 matrix that represents a 2D projection transformation (see Appendix C) such that $\vec{m}'_i = H\vec{m}_i$. H defines a mapping from one plane to another plane (Figure B.1) and has 8 degree of freedom (9 entries defined up to a scale). This way, four point correspondences are only necessary to calculate H, as each point correspondence provides two constraints. Additionally, these correspondences should not form triplets of collinear points in order to avoid a degenerate transformation and obtain a unique solution of H.

That equation can be formulated as $\vec{m}'_i \times H\vec{m}_i = 0$. Moreover, using homogeneous coordinates ($\vec{m}'_i = (x'_i, y'_i, z'_i)^T$), and some changes in notation ($\vec{h}^{jT}\vec{m}_i = \vec{m}_i^T\vec{h}^j$, where $\vec{h}^{jT}$ is the j-th row of the matrix H), the original equation can be written in the form

[1]Existence of two local minima for the pose estimation error function.

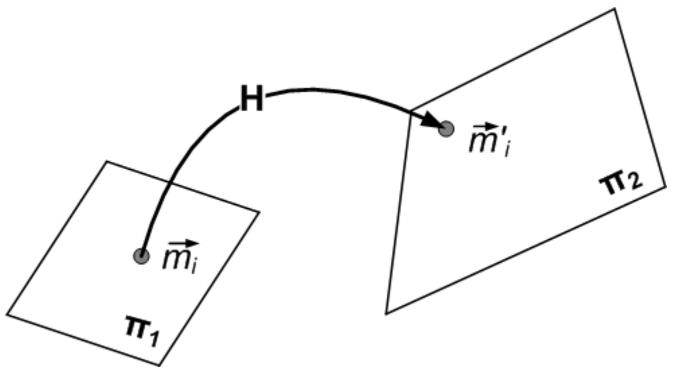


Figure B.1: The mapping of points from π_1 plane to π_2 plane.

$$\begin{bmatrix} 0^T & -w'_i\vec{m}_i^T & y'_i\vec{m}_i^T \\ w'_i\vec{m}_i^T & 0^T & -x'_i\vec{m}_i^T \\ -y'_i\vec{m}_i^T & x'_i\vec{m}_i^T & 0^T \end{bmatrix} \begin{bmatrix} \vec{h}^1 \\ \vec{h}^2 \\ \vec{h}^3 \end{bmatrix} = 0. \tag{B.1}$$

However, only two equations of Equation B.1 are linearly independent, so the set of equations becomes:

$$\begin{bmatrix} 0^T & -w'_i\vec{m}_i^T & y'_i\vec{m}_i^T \\ w'_i\vec{m}_i^T & 0^T & -x'_i\vec{m}_i^T \end{bmatrix} \begin{bmatrix} \vec{h}^1 \\ \vec{h}^2 \\ \vec{h}^3 \end{bmatrix} = 0. \tag{B.2}$$

This can be expressed as a linear equation in the unknown $\vec{h}$: $A_i\vec{h} = 0$, where A_i is a 2x9 matrix whose entries are given by the coordinates of the $\vec{m}_i \leftrightarrow \vec{m}'_i$ points, and $\vec{h}$ is a 9-vector made of the coefficients H_{ij}.

Given a set of four $\vec{m}_i \leftrightarrow \vec{m}'_i$ correspondences, each with its 2x9 A_i matrix, all of them are assembled into a single 8x9 matrix A. Thus, $A\vec{h} = 0$ can be solved using Singular Value Decomposition (SVD) (Flaquer et al., 2004). More precisely, $A_{8x9} = U_{8x9}D_{9x9}V_{9x9}^T$, where $D = diag(\sigma_1, \sigma_2, ..., \sigma_9)$ is the diagonal matrix of singular values arranged in descending order down the diagonal and the matrices U and V are orthonormal. The columns of V are the eigenvectors of A^TA and the required solution ($\vec{h}$) is the column of V corresponding the smallest singular value (σ_9, last column).

DLT minimizes the algebraic error (minimizes the norm $\|Ah\|$), which is not geometrically or statistically meaningful. However it has a low computational cost and offers a linear (and consequently unique) solution, so it is used as a starting point for a non-linear minimization of a geometric or statistical cost function.

B.2 Pose estimation from a 3D plane

Given the known camera intrinsic parameters (K) and the image projection of a 3D planar structure, the camera extrinsic parameters (R_t) can be determined. The relation between the coordinates of points that lie on a 3D plane ($\vec{M_i} = (X, Y, 0)^T$, assuming $Z = 0$ plane) and its image projection ($\vec{m}_i = KR_t\vec{M_i}$) can be represented using homogeneous coordinates as:

$$\vec{m}_i = K \begin{bmatrix} \vec{r}^{\,1} & \vec{r}^{\,2} & \vec{r}^{\,3} & \vec{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K \begin{bmatrix} \vec{r}^{\,1} & \vec{r}^{\,2} & \vec{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \quad \text{(B.3)}$$

where $\vec{r}^{\,1}$, $\vec{r}^{\,2}$ and $\vec{r}^{\,3}$ are the first, second and third column of the rotation matrix R respectively, t is the translation vector and H is a 3x3 homogeneous matrix, called a homography matrix.

The matrix H can be estimated from four correspondences $\vec{m}_i \leftrightarrow \vec{M}_i$ using the DLT algorithm (Section B.1). Furthermore, since K is known, the camera pose can be recovered from the product $K^{-1}H$, where the last column $\vec{r}^{\,3}$ is computed as the cross product of $\vec{r}^{\,1}$ and $\vec{r}^{\,2}$ ($\vec{r}^{\,3} = \vec{r}^{\,1} \times \vec{r}^{\,2}$) to satisfy the orthonormality constraint of the rotation matrix R. Indeed, the orthonormality conditions are never perfectly met, so a renormalization step is applied (Simon and Berger, 2002): $\vec{r}^{\,2} = \vec{r}^{\,2}/\|\vec{r}^{\,2}\|$, $\vec{r}^{\,3} = \vec{r}^{\,1} \times \vec{r}^{\,2}/\|\vec{r}^{\,1} \times \vec{r}^{\,2}\|$, $\vec{r}^{\,1} = \vec{r}^{\,2} \times \vec{r}^{\,3}$.

Appendix C

Hierarchy of 2D transformations

This appendix describes the specializations of a 2D projective transformation and their geometric properties. Assuming that the 2D projective transformation is a group (*projective linear group*), then these specializations (affine, similarity and isometry) are subgroups of this group. All these transformations are described by those terms that are *invariant*, i.e., those elements or quantities of a geometric configuration that are preserved. See (Hartley and Zisserman, 2004) for more details.

C.1 Isometries

2D isometries are transformations that preserve Euclidean distance (iso=same, metric=measure). They are represented as

$$\begin{pmatrix} x\prime \\ y\prime \\ 1 \end{pmatrix} = \begin{bmatrix} \epsilon\, cos\, \theta & -sin\, \theta & t_x \\ \epsilon\, sin\, \theta & cos\, \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{C.1}$$

where $\epsilon = 1$ means that the isometry is orientation-preserving and is a Euclidean transformation (a composition of a translation and rotation). $\epsilon = -1$, meanwhile, means that the isometry reverses orientation.

A 2D isometry transformation has 3 degrees of freedom, one for rotation and two for the translation. Thus, the transformation can be computed from two point correspondences.

The invariants of a 2D isometry are length (the distance between two points), angle (the angle between two lines), and area.

C.2 Similarity transformations

2D similarity transformations are isometries composed with an isotropic scaling. They are represented as

$$\begin{pmatrix} x\prime \\ y\prime \\ 1 \end{pmatrix} = \begin{bmatrix} s\ cos\ \theta & -s\ sin\ \theta & t_x \\ s\ sin\ \theta & s\ cos\ \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{C.2}$$

where the scalar s represents the isotropic scaling.

A 2D similarity transformation has 4 degrees of freedom, the scale factor and those associated with a Euclidean transformation. Thus, the transformation can be computed from two point correspondences.

The invariants of a 2D similarity are angle (parallel lines are mapped to parallel lines), ratio of lengths and ratio of areas.

C.3 Affine transformations

2D affine transformations are non-singular linear transformations followed by a translation. They are represented as

$$\begin{pmatrix} x\prime \\ y\prime \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{C.3}$$

A 2D affine transformation has 6 degrees of freedom corresponding to the six matrix elements. Thus, the transformation can be computed from three point correspondences.

The invariants of a 2D affine are parallel lines (intersection of parallel lines remains at infinity), ratio of lengths of parallel line segments, and ratio of areas.

C.4 Projective transformations

A 2D projective transformation, also called homography, is a linear transformation on homogeneous 3-vectors represented by a non-singular 3x3 matrix:

$$\begin{pmatrix} x\prime \\ y\prime \\ 1 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{C.4}$$

A 2D projective transformation has 8 degrees of freedom (9 entries defined up to a scale). Thus, a 2D projective transformation between two planes can be computed from four point correspondences, with no three collinear on either plane.

A 2D projective transformation is invariant to a ratio of ratios or *cross ratio* of lengths on a line (Figure C.1). Given 4 points x_i the cross ratio is defined as

$$Cross(x_1, x_2, x_3, x_4) = \frac{|x_1x_2|\,|x_3x_4|}{|x_1x_3|\,|x_2x_4|}$$

where

$$|x_ix_j| = det \begin{bmatrix} x_{i1} & x_{j1} \\ x_{i2} & x_{j2} \end{bmatrix}$$

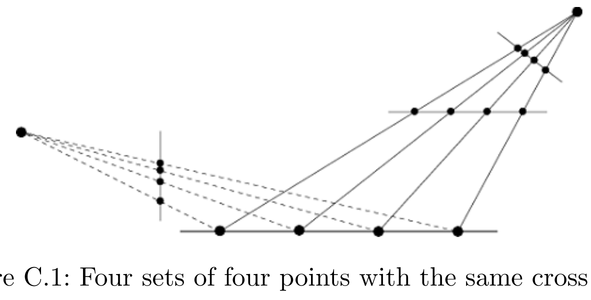


Figure C.1: Four sets of four points with the same cross ratio.

C.5 Summary

Figure C.2 summarizes the 2D transformation groups and their invariant properties. Transformations lower in the image are specializations of those above. A transformation lower in the image inherits the invariants of those above.

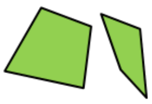
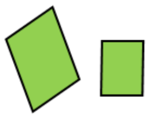
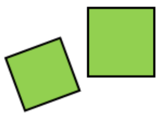
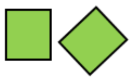
Group	**Matrix**	**Distortion**	**Invariant properties**
Projective 8DoF	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, order of contact (intersection, tangency, inflection, etc.), cross ratio.
Affine 6DoF	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallellism, ratio of areas, ratio of lengths on parallel lines (e.g midpoints), linear combinations of vectors (centroids).
Similarity 4DoF	$\begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratios of lengths, angles.
Euclidean 3DoF	$\begin{bmatrix} \varepsilon\cos\theta & -\sin\theta & t_x \\ \varepsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Lengths, areas.

Figure C.2: 2D transformation groups.

Appendix D

Usability Experiments

This appendix completes the description of the experiment that has been done to validate the usability of the AR system (Section 6.5).

D.1 Paper Documentation

Here the paper documentation that the experimenter gave to the subjects is presented. This correspond to the T1 disassembly order. The document of the T2 disassembly order is analogous, changing the sequence order of some steps: 3, 4, 1, 2, 5, 7, 6. This way it is not copied here to avoid duplication.

Disassembly of a Lego Elephant

This manual outlines the steps that must be followed to disassemble some parts of the Lego Elephant presented in Figure 1.

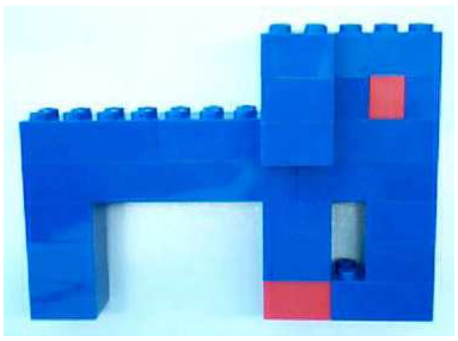
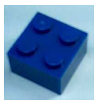
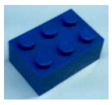


Figure 1 - Lego Elephant.

Note that you do not need any special tools to disassemble this object, just use your hands.

7 parts must be disassembled in total:

- 3 2x2 Lego bricks 
- 2 2x3 Lego bricks 
- 2 1x2 Lego bricks

IMPORTANT: These parts must be disassembled in a correct order, i.e., there is a predefined disassembly order that must be satisfied.

Please read the following instructions carefully to avoid mistakes.

Step 1:

Disassemble the 2x2 Lego brick that is at the bottom of the hind leg of the Elephant: take the Elephant with one hand and pull out the brick with the other hand. Do it gently, trying not to destroy the overall structure of the object.

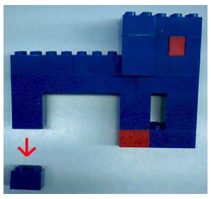


Figure 2 – Step 1.

Step 2:

Disassemble the 2x2 Lego brick that is at the bottom of the hind leg of the Elephant after Step 1: take the Elephant with one hand and pull out the brick with the other hand. Do it gently, trying not to destroy the overall structure of the object.

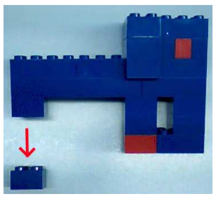


Figure 3 – Step 2.

Step 3:

Disassemble the 2x3 Lego brick that is at the head of the Elephant after Step 2: take the Elephant with one hand and pull out the brick with the other hand. Do it gently, trying not to destroy the overall structure of the object.

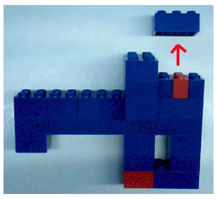


Figure 4 – Step 3.

Step 4:

Disassemble the 1x2 Lego brick that is close to the eye of the Elephant after Step 3: take the Elephant with one hand and pull out the brick with the other hand. Do it gently, trying not to destroy the overall structure of the object.

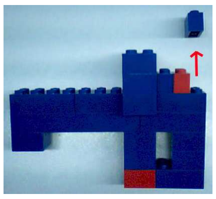


Figure 5 – Step 4.

Step 5:

Disassemble the 2x3 Lego brick that is at the top of the body of the Elephant after Step 4: take the Elephant with one hand and pull out the brick with the other hand. Do it gently, trying not to destroy the overall structure of the object.

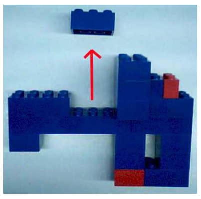


Figure 6 – Step 5.

Step 6:

Disassemble the 1x2 Lego brick that corresponds to the eye of the Elephant: take the Elephant with one hand and pull out the brick with the other hand. Do it gently, trying not to destroy the overall structure of the object.

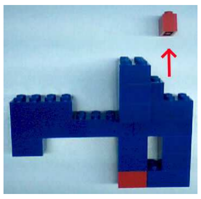


Figure 7 – Step 6.

Step 7:

Disassemble the 2x2 Lego brick that is at the bottom of the hind leg of the Elephant after Step 6: take the Elephant with one hand and pull out the brick with the other hand. Do it gently, trying not to destroy the overall structure of the object.

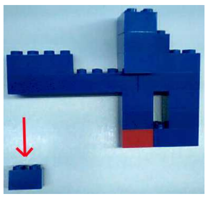


Figure 8 – Step 7.

The final state should be the one shown at Figure 9 to ensure the correctness of the disassembly task.

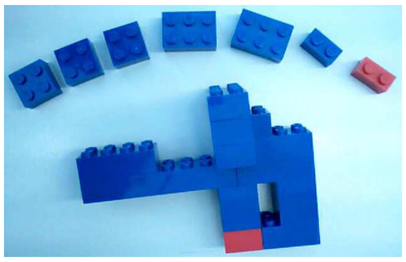


Figure 9 – Lego Elephant after the disassembly process.

Appendix E

Generated Publications

Books

Sánchez, J. R., Álvarez, H., and Borro, D. *Gft: Gpu fast triangulation of 3d points* (ISBN: 3-642-15909-5), volume 6374 of *Lecture Notes in Computer Science, Computer Vision and Graphics*, pp. 235–242. Springer-Verlag Berlin Heidelberg. 2010.

Journals

Puerto, M., Gil, J., Álvarez, H., and Sánchez, E. "Influence of user grasping position on haptic rendering". *IEEE/ASME Transactions on Mechatronics*, N. 99, pp. 1–9. 2011.

Álvarez, H., Leizea, I., and Borro, D. "A new marker design for a robust marker tracking system against occlusions". *Submitted to Computer Animation and Virtual Worlds*, 2011.

Álvarez, H. and Borro, D. "Junction assisted 3d pose retrieval of untextured 3d models in monocular images". *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

Conferences

Basogain, X., Olabe, M., Etxebarri, A., Izkara, J. L., Garrido, R., and Álvarez, H. "Towards the augmented reality in wearable personal assistants". In *II Jornadas sobre Realidad Virtual y Entornos Virtuales (JOREVIR'08)*. Albacete, Spain. June, 2008.

Álvarez, H. and Borro, D. "Cálculo de la pose de la cámara ante oclusiones de un marcador". In *Proceedings of the XVIII Conferencia Española de Computación Gráfica (CEIG'08)*, pp. 123–132. Barcelona, Spain. September, 2008.

Álvarez, H. and Borro, D. "A novel approach to achieve robustness against marker occlusion". In *International Conference on Computer Vision Theory and Applications (VISAPP'09)*, pp. 478–483. Lisboa, Portugal. February, 2009.

Sánchez, J. R., Álvarez, H., and Borro, D. "Towards real time 3d tracking and reconstruction on a gpu using monte carlo simulations". In *International Symposium on Mixed and Augmented Reality (ISMAR'10)*, pp. 185–192. Seoul, Korea. October, 2010.

Álvarez, H., Aguinaga, I., and Borro, D. "Providing guidance for maintenance operations using automatic markerless augmented reality system". In *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR'11)*, pp. 181–190. Basel, Switzerland. October, 2011.

Posters

Barandiarán, J., Álvarez, H., and Borro, D. "Edge-based markerless 3d tracking of rigid objects". In *International Conference on Artificial Reality and Telexistence (ICAT'07)*, pp. 282–283. Esbjerg, Denmark. November, 2007.

Sánchez, J. R., Álvarez, H., and Borro, D. "Gpu optimizer: A 3d reconstruction on the gpu using monte carlo simulations". In *Proceedings of the 5th International Conference on Computer Vision Theory and Applications (VISAPP'10)*, pp. 443–446. Angers, France. May, 2010.

References

Aguinaga, I. *Automatic analysis of the precedence relationships and disassembly routes for selective disassembly planning in mechanical systems using virtual mock-ups.* PhD thesis, Escuela Superior de Ingenieros, Universidad de Navarra. 2007.

Aguinaga, I., Borro, D., and Matey, L. “Path planning techniques for the simulation of disassembly tasks”. *Assembly Automation*, Vol. 27, N. 3, pp. 207–214. 2007.

Aguinaga, I., Borro, D., and Matey, L. “Parallel rrt-based path planning for selective disassembly planning”. *The International Journal of Advanced Manufacturing Technology*, Vol. 36, pp. 1221–1233. 2008.

Álvarez, H., Aguinaga, I., and Borro, D. “Providing guidance for maintenance operations using automatic markerless augmented reality system”. In *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR’11)*, pp. 181–190. Basel, Switzerland. October, 2011.

Álvarez, H. and Borro, D. “Cálculo de la pose de la cámara ante oclusiones de un marcador”. In *Proceedings of the XVIII Conferencia Española de Computación Gráfica (CEIG’08)*, pp. 123–132. Barcelona, Spain. September, 2008.

Álvarez, H. and Borro, D. “A novel approach to achieve robustness against marker occlusion”. In *International Conference on Computer Vision Theory and Applications (VISAPP’09)*, pp. 478–483. Lisboa, Portugal. February, 2009.

Álvarez, H. and Borro, D. “Junction assisted 3d pose retrieval of untextured

3d models in monocular images". *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

Álvarez, H., Leizea, I., and Borro, D. "A new marker design for a robust marker tracking system against occlusions". *Submitted to Computer Animation and Virtual Worlds*, 2011.

Armstrong, M. and Zisserman, A. "Robust object tracking". In *Asian Conference on Computer Vision (ACCV'95)*, volume I, pp. 58–61. Singapore. December, 1995.

Arulampalam, M. S., Maskell, S., and Gordon, N. "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking". *IEEE Transactions on Signal Processing*, Vol. 50, pp. 174–188. 2002.

Azuma, R. "A survey of augmented reality". *Media*, Vol. 6, N. 4, pp. 355–385. 1997.

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. "Recent advances in augmented reality". *IEEE Computer Graphics and Applications*, Vol. 21, N. 6, pp. 34–47. 2001.

Baird, K. and Barfield, W. "Evaluating the effectiveness of augmented reality displays for a manual assembly task". *Virtual Reality*, Vol. 4, pp. 250–259. 1999.

Barandiarán, J., Álvarez, H., and Borro, D. "Edge-based markerless 3d tracking of rigid objects". In *International Conference on Artificial Reality and Telexistence (ICAT'07)*, pp. 282–283. Esbjerg, Denmark. November, 2007.

Basogain, X., Olabe, M., Etxebarri, A., Izkara, J. L., Garrido, R., and Álvarez, H. "Towards the augmented reality in wearable personal assistants". In *II Jornadas sobre Realidad Virtual y Entornos Virtuales (JOREVIR'08).* Albacete, Spain. June, 2008.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. "Speeded-up robust features (surf)". *Computer Vision and Image Understanding*, Vol. 110, pp. 346–359. 2008.

Billinghurst, M., Hakkarainen, M., and Woodward, C. "Augmented assembly using a mobile phone". In *Proceedings of the 7th International*

Conference on Mobile and Ubiquitous Multimedia (MUM'08), pp. 84–87. Umea, Sweden. December, 2008.

Bleser, G., Pastarmov, Y., and Stricker, D. "Real-time 3d camera tracking for industrial augmented reality applications". In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'05)*, pp. 47–54. Plzen-Bory, Czech Republic. January, 2005.

Blum, T., Heining, S. M., Kutter, O., and Navab, N. "Advanced training methods using an augmented reality ultrasound simulator". In *8th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*, pp. 177–178. Orlando, USA. October, 2009.

Bouguet, J.-Y. "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm". 2000.

Bradski, G. and Kaehler, A. *Learning OpenCV: Computer vision with the OpenCV library* (ISBN: 0596516134). O'Reilly Media, 1st edition. 2008.

Brown, D. C. "Decentering distortion of lenses". Vol. 32, N. 3, pp. 444–462. 1966.

Brown, M., Burschka, D., and Hager, G. "Advances in computational stereo". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, N. 8, pp. 993–1008. 2003.

Canada, C., Mcdonald, C., and Roth, G. "Replacing a mouse with hand gesture in a plane-based augmented reality system". In *Proceedings of Vision Interface (VI'03)*. Halifax, Canada. June, 2003.

Cawood, S. and Fiala, M. *Augmented reality a practical guide*. Pragmatic Bookshelf. 2008.

Cazorla, M. and Escolano, F. "Two bayesian methods for junction classification". *IEEE Transactions on Image Processing*, Vol. 12, N. 3, pp. 317–327. 2003.

Chum, O. and Matas, J. "Matching with prosac - progressive sample consensus". volume 1, pp. 220–226. Los Alamitos, California, USA. June, 2005.

Costa, M. S. and Shapiro, L. G. “3d object recognition and pose with relational indexing”. *Computer Vision and Image Understanding*, Vol. 79, pp. 364–407. 2000.

Davison, A. “Real-time simultaneous localisation and mapping with a single camera”. In *Proceedings of 9th IEEE International Conference on Computer Vision (ICCV’03)*, pp. 1403 –1410 vol.2. Nice, France. October, 2003.

De Crescenzio, F., Frantini, M., Persiani, F., Di Stefano, L., Azzari, P., and Salti, S. “Augmented reality for aircraft maintenance and operations support”. *IEEE Computer Graphics and Applications*, Vol. 31, N. 1, pp. 96–101. 2011.

de Ipiña, D. L., ca, P. R. S. M., and Hopper, A. “Trip: A low-cost vision-based location system for ubiquitous computing”. *Personal and Ubiquitous Computing*, Vol. 6, pp. 206–219. 2002.

Deutscher, J., Blake, A., and Reid, I. “Articulated body motion capture by annealed particle filtering”. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR’00)*, volume 2, pp. 126 –133. Hilton Head, SC, USA. June, 2000.

Drost, B., Ulrich, M., Navab, N., and Ilic, S. “Model globally, match locally: Efficient and robust 3d object recognition”. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR’10)*, pp. 998–1005. San Francisco, USA. June, 2010.

Drummond, T. and Cipolla, R. “Real-time visual tracking of complex structures”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, pp. 932–946. 2002.

Farkhatdinov, I. and Ryu, J.-H. “Development of educational system for automotive engineering based on augmented reality”. In *International Conference on Engineering Education and Research (ICEER’09)*. Seoul, South Korea. August, 2009.

Faugeras, O., Luong, Q. T., and Papadopoulo, T. *The geometry of multiple images: the laws that govern the formation of multiple images of a scene.* MIT Press. 2001.

Ferrari, V., Fevrier, L., Jurie, F., and Schmid, C. "Groups of adjacent contour segments for object detection". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, N. 1, pp. 36–51. 2008.

Fiala, M. "Artag, a fiducial marker system using digital techniques". pp. 590–596. San Diego, CA, USA. June, 2005.

Fiala, M. "Comparing artag and artoolkit plus fiducial marker systems". In *IEEE International Workshop on Haptic Audio Visual Environments and their Applications (HAVE'05).* Ottawa, Ontario, Canada. October, 2005.

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

Flaquer, J., Olaizola, J., and Olaizola, J. *Curso de álgebra lineal.* Eunsa. 2004.

Franken, T., Dellepiane, M., Ganovelli, F., Cignoni, P., Montani, C., and Scopigno, R. "Minimizing user intervention in registering 2d images to 3d models". *The Visual Computer*, Vol. 21, N. 8-10, pp. 619–628. 2005.

Gall, J., Potthoff, J., Schnörr, C., Rosenhahn, B., and Seidel, H.-P. "Interacting and annealing particle filters: Mathematics and a recipe for applications". *Journal of Mathematical Imaging and Vision*, Vol. 28, N. 1, pp. 1–18. 2007.

Gall, J., Rosenhahn, B., and Seidel, H. P. "Robust pose estimation with 3d textured models". In *Pacific-Rim Symposium on Image and Video Technology (PSIVT'06)*, pp. 84–95. Hsinchu, Taiwan. December, 2006.

Gauglitz, S., Höllerer, T., and Turk, M. "Evaluation of interest point detectors and feature descriptors for visual tracking". *International Journal of Computer Vision*, Vol. 94, pp. 335–360. 2011.

Hakkarainen, M., Woodward, C., and Billinghurst, M. "Augmented assembly using a mobile phone". In *7th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR'08)*, pp. 167–168. Cambridge, UK. September, 2008.

Harris, C. “Tracking with rigid objects”. *MIT Press*, 1992.

Hartley, R. I. “Euclidean reconstruction from uncalibrated views”. In *Proceedings of the Second Joint European - US Workshop on Applications of Invariance in Computer Vision*, pp. 237–256. Ponta Delgada, Azores, Portugal. October, 1994.

Hartley, R. I. and Zisserman, A. *Multiple view geometry in computer vision.* Cambridge University Press, ISBN: 0521540518, second edition. 2004.

Heisele, B. and Rocha, C. “Local shape features for object recognition”. In *19th International Conference on Pattern Recognition (ICPR’08)*, pp. 1–4. Tampa, Florida, USA. December, 2008.

Hemayed, E. “A survey of camera self-calibration”. In *Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS’03)*, pp. 351 – 357. Miami, FL, USA. July, 2003.

Henderson, S. J. and Feiner, S. “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 (ISMAR’09)*, pp. 135–144. Orlando, USA. October, 2009.

Hinterstoisser, S., Benhimane, S., and Navab, N. “N3m: Natural 3d markers for real-time object detection and pose estimation”. In *IEEE 11th International Conference on Computer Vision (ICCV’07)*, pp. 1–7. Rio de Janeiro, Brazil. October, 2007.

Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., and Navab, N. “Dominant orientation templates for real-time detection of texture-less objects”. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR’10)*, pp. 2257–2264. San Francisco, USA. June, 2010.

Holzer, S., Hinterstoisser, S., Ilic, S., and Navab, N. “Distance transform templates for object detection and pose estimation”. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR’09)*, pp. 1177–1184. Miami, FL, USA. June, 2009.

Homem de Mello, L. and Sanderson, A. “Representations of mechanical assembly sequences”. *IEEE Transactions on Robotics and Automation*, Vol. 7, N. 2, pp. 211–227. 1991.

Izkara, J., Mediavilla, A., Rodriguez-Maribona, I., and Armijo, A. "Information technologies and cultural heritage: Innovative tools for the support in the participative management of historical centres". In *8th European Conference on Research for Protection, Conservation and Enhancement of Cultural Heritage (CHRESP'08).* Ljubljana, Slovenia. November, 2008.

Jiao, X. and Heath, M. T. "Feature detection for surface meshes". In *Proceedings of 8th International Conference on Numerical Grid Generation in Computational Field Simulations*, pp. 705–714. Waikiki Beach, Hawaii. June, 2002.

Juan, C., Llop, E., Abad, F., and Lluch, J. "Learning words using augmented reality". In *IEEE 10th International Conference on Advanced Learning Technologies (ICALT'10)*, pp. 422–426. Sousse, Tunisia. July, 2010.

Kato, H. and Billinghurst, M. "Marker tracking and hmd calibration for a video-based augmented reality conferencing system". In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pp. 85–94. Washington, DC, USA. October, 1999.

Klein, G. and Murray, D. "Full-3d edge tracking with a particle filter". In *Proceeding of the British Machine Vision Conference (BMVC'06)*, volume 3, pp. 1119–1128. Edinburgh. September, 2006.

Klein, G. and Murray, D. "Parallel tracking and mapping for small ar workspaces". In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, pp. 1–10. Nara, Japan. November, 2007.

Koller, D., Danilidis, K., and Nagel, H.-H. "Model-based object tracking in monocular image sequences of road traffic scenes". *International Journal of Computer Vision*, Vol. 10, pp. 257–281. 1993.

Kotake, D., Satoh, K., Uchiyama, S., and Yamamoto, H. "A fast initialization method for edge-based registration using an inclination constraint". In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, pp. 239–248. Orlando, USA. November, 2007.

Laganiere, R. and Elias, R. “The detection of junction features in images”. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP’04)*, volume 3, pp. 573–576. Montreal, Quebec, Canada. May, 2004.

Lamdan, Y. and Wolfson, H. “Geometric hashing: A general and efficient model-based recognition scheme”. In *Second International Conference on Computer Vision (CV’88)*, pp. 238–249. Tampa, USA. December, 1988.

Latombe, J.-C. “Motion planning: A journey of robots, molecules, digital actors, and other artifacts”. *International Journal of Robotics Research*, Vol. 18, N. 11, pp. 1119–1128. 1999.

LaViola, J. J. “Double exponential smoothing: an alternative to kalman filter-based predictive tracking”. In *Proceedings of the Workshop on Virtual Environments (EGVE’03)*, pp. 199–206. Zurich, Switzerland. May, 2003.

Lee, G., Billinghurst, M., and Kim, G. “Occlusion based interaction methods for tangible augmented reality environments”. In *Proceedings of ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry (VRCAI’04)*, pp. 419–426. Nanyang Technological University, Singapore. June, 2004.

Lepetit, V. and Fua, P. “Monocular model-based 3d tracking of rigid objects: a survey”. *Foundations and trends in computer graphics and vision*, Vol. 1, pp. 1–89. 2005.

Lepetit, V., Moreno-Noguer, F., and Fua, P. “Epnp: An accurate o(n) solution to the pnp problem”. *International Journal of Computer Vision*, Vol. 81, pp. 155–166. 2009.

Liverani, A., Amati, G., and Caligiana, G. “A cad-augmented reality integrated environment for assembly sequence check and interactive validation”. *Concurrent Engineering*, Vol. 12, N. 1, pp. 67–77. 2004.

Longuet-Higgins, H. C. “A computer algorithm for reconstructing a scene from two projections”. *Nature*, Vol. 293, pp. 133–135. 1981.

Lowe, D. “Robust model-based motion tracking through the integration of search and estimation”. *International Journal of Computer Vision*, Vol. 8, pp. 113–122. 1992.

Lowe, D. "Local feature view clustering for 3d object recognition". In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)*, volume 1, pp. 682–688. Kauai, HI, USA. December, 2001.

Lowe, D. "Distinctive image features from scale-invariant keypoints". *International Journal on Computer Vision*, Vol. 60, pp. 91–110. 2004.

Lucas, B. D. and Kanade, T. "An iterative image registration technique with an application to stereo vision". In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI'81)*, pp. 674–679. Vancouver, BC, Canada. August, 1981.

Maad, S. *Augmented reality.* InTech. January 2010.

Madsen, K., Nielsen, H. B., and Tingleff, O. "Methods for non-linear least squares problems". 1999.

Malik, S., Roth, G., and Mcdonald, C. "Robust 2d tracking for real-time augmented reality". In *Proceedings of International Conference on Vision Interface (VI'02)*, pp. 399–406. Calgary, Canada. May, 2002.

Marian, R. M., Luong, L. H., and Abhary, K. "Assembly sequence planning and optimisation using genetic algorithms part i. automatic generation of feasible assembly sequences". *Applied Soft Computing*, Vol. 2/3F, pp. 223–253. 2003.

Marimon, D., Maret, Y., Abdeljaoued, Y., and Ebrahimi, T. "Particle filter based camera tracker fusing marker and feature point based cues". In *Proc. of the IS&T/SPIE Electronic Imaging Conf. on Visual Communications and Image Processing.* San Jose, CA, USA. February, 2007.

Mikolajczyk, K. and Schmid, C. "Scale & affine invariant interest point detectors". *International Journal of Computer Vision*, Vol. 60, pp. 63–86. 2004.

Mikolajczyk, K. and Schmid, C. "A performance evaluation of local descriptors". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, pp. 1615–1630. 2005.

Milgram, P., Takemura, H., Utsumi, A., and Kishino, F. "Augmented reality: A class of displays on the Reality-Virtuality continuum". In *Proceedings of the SPIE Conference on Telemanipulator and Telepresence Technologies*, volume 2351, pp. 282–292. Boston, Massachusetts, USA. November, 1995.

Mohring, M., Lessig, C., and Bimber, O. "Video see-through ar on consumer cell-phones". In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pp. 252–253. Arlington, VA, USA. November, 2004.

Opelt, A., Pinz, A., and Zisserman, A. "A boundary-fragment-model for object detection". In *European Conference on Computer Vision (ECCV'06)*, pp. 575–588. Graz, Austria. May, 2006.

Otero, M. and Dolado, J. *Medición para la gestión en la ingeniería del software* (ISBN: 84-7897-403-2), chapter 3, pp. 51–72. RA-MA. 2000.

Parida, L., Geiger, D., and Hummel, R. "Junctions: detection, classification, and reconstruction". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, N. 7, pp. 687–698. 1998.

Platonov, J., Heibel, H., Meier, P., and Grollmann, B. "A mobile markerless ar system for maintenance and repair". In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'06)*, pp. 105–108. Santa Barbara, USA. October, 2006.

Platonov, J. and Langer, M. "Automatic contour model creation out of polygonal cad models for markerless augmented reality". In *Proceedings of 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, pp. 1–4. Nara, Japan. November, 2007.

Puerto, M., Gil, J., Álvarez, H., and Sánchez, E. "Influence of user grasping position on haptic rendering". *IEEE/ASME Transactions on Mechatronics*, N. 99, pp. 1–9. 2011.

Pupilli, M. and Calway, A. "Real-time camera tracking using a particle filter". In *In Proceedings of the British Machine Vision Conference (BMVC'05)*, pp. 519–528. Oxford, U.K. September, 2005.

Quintana, A., Quirós, R. J., Remolar, I., and Camahort, E. "Un sistema de realidad aumentada basado en el campo de luz". In *XX Congreso Español de Informática Gráfica (CEIG'10)*, pp. 97–104. Valencia, Spain. September, 2010.

Raghavan, V., Molineros, J., and Sharma, R. "Interactive evaluation of assembly sequences using augmented reality". *IEEE Transactions on Robotics and Automation*, Vol. 15, N. 3, pp. 435–449. 1999.

Reitmayr, G., Eade, E., and Drummond, T. W. "Semi-automatic annotations in unknown environments". In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, pp. 1–4. Nara, Japan. November, 2007.

Rolland, J. P., Baillot, Y., and Goon, A. A. *A survey of tracking technology for virtual environments*, pp. 67–112. Ed. Barfield and Caudell, Mahwah, USA. 2001.

Rosten, E. and Drummond, T. "Fusing points and lines for high performance tracking". In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05)*, pp. 1508–1515. Beijing, China. October, 2005.

Rosten, E. and Drummond, T. "Machine learning for high-speed corner detection". In *European Conference on Computer Vision (ECCV'06)*, volume 1, pp. 430–443. Graz, Austria. May, 2006.

Rothganger, F., Lazebnik, S., Schmid, C., and Ponce, J. "3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial contraints". *International Journal of Computer Vision*, Vol. 66, N. 3, pp. 231–259. 2006.

Russ, J. C. *The image processing handbook (3rd ed.)*. CRC Press, Inc., Boca Raton, FL, USA. 1999.

Ryoo, M. S., Grauman, K., and Aggarwal, J. K. "A task-driven intelligent workspace system to provide guidance feedback". *Computer Vision and Image Understanding*, Vol. 114, pp. 520–534. 2010.

Salih, Y. and Malik, A. S. "Comparison of stochastic filtering methods for 3d tracking". *Pattern Recognition*, Vol. 44, pp. 2711–2737. 2011.

Sánchez, J. R. *A stochastic parallel method for real time monocular slam applied to augmented reality.* PhD thesis, Escuela Superior de Ingenieros, Universidad de Navarra. 2010.

Sánchez, J. R., Álvarez, H., and Borro, D. *Gft: Gpu fast triangulation of 3d points* (ISBN: 3-642-15909-5), volume 6374 of *Lecture Notes in Computer Science, Computer Vision and Graphics*, pp. 235–242. Springer-Verlag Berlin Heidelberg. 2010.

Sánchez, J. R., Álvarez, H., and Borro, D. "Gpu optimizer: A 3d reconstruction on the gpu using monte carlo simulations". In *Proceedings of the 5th International Conference on Computer Vision Theory and Applications (VISAPP'10)*, pp. 443–446. Angers, France. May, 2010.

Sánchez, J. R., Álvarez, H., and Borro, D. "Towards real time 3d tracking and reconstruction on a gpu using monte carlo simulations". In *International Symposium on Mixed and Augmented Reality (ISMAR'10)*, pp. 185–192. Seoul, Korea. October, 2010.

Schmalstieg, D. and Wagner, D. "Experiences with handheld augmented reality". In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, pp. 1–13. Nara, Japan. November, 2007.

Schoenfelder, R. and Schmalstieg, D. "Augmented reality for industrial building acceptance". In *IEEE Virtual Reality Conference (VR '08)*, pp. 83–90. Reno, Nevada, USA. March, 2008.

Schweighofer, G. and Pinz, A. "Robust pose estimation from a planar target". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, N. 12, pp. 2024–2030. 2006.

Sehgal, A. "3D object recognition using bayesian geometric hashing and pose clustering". *Pattern Recognition*, Vol. 36, N. 3, pp. 765–780. 2003.

Selinger, A. and Nelson, R. C. "A perceptual grouping hierarchy for appearance-based 3d object recognition". *Computer Vision and Image Understanding*, Vol. 76, pp. 83–92. 1999.

Shahrokni, A., Vacchetti, L., Lepetit, V., and Fua, P. "Polyhedral object detection and pose estimation for augmented reality applications". In

Proceedings of the Computer Animation (CA'02), pp. 65–69. Geneva, Switzerland. June, 2002.

Simon, G. and Berger, M.-O. "Reconstructing while registering: a novel approach for markerless augmented reality". In *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'02)*, pp. 285–293. Darmstadt, Germany. October, 2002.

Sinha, S. N., Frahm, J.-M., Pollefeys, M., and Genc, Y. "Gpu-based video feature tracking and matching". In *Workshop on Edge Computing Using New Commodity Architectures (EDGE'06)*. Chapel Hill, USA. May, 2006.

Stark, M., Goesele, M., and Schiele, B. "Back to the future: Learning shape models from 3d cad data". In *Proceedings of the British Machine Vision Conference (BMVC'10)*, pp. 106.1–106.11. Aberystwyth, UK. August, 2010.

Steger, C. "Occlusion, clutter, and illumination invariant object recognition". In *International Archives of Photogrammetry and Remote Sensing (IAPRS'02)*, volume XXXIV, part 3A, pp. 345–350. 2002.

Sutherland, I. E. "A head-mounted three dimensional display". In *Proceedings of Fall Joint Computer Conference (FJCC'68)*, volume 33, pp. 757–764. San Francisco, California, USA. December, 1968.

Tang, A., Owen, C., Biocca, F., and Mou, W. "Comparative effectiveness of augmented reality in object assembly". In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI'03)*, pp. 73–80. Ft. Lauderdale, Florida, USA. April, 2003.

Tateno, K. "A nested marker for augmented reality". In *IEEE Virtual Reality Conference (VR'07)*, pp. 259–262. Charlotte, North Carolina, USA. March, 2007.

Teichrieb, V., Lima, M., Lourenc, E., Bueno, S., Kelner, J., and Santos, I. H. F. "A survey of online monocular markerless augmented reality". *International Journal of Modeling and Simulation for the Petroleum Industry*, Vol. 1, N. 1, pp. 1–7. 2007.

Tomasi, C. and Kanade, T. "Detection and tracking of point features". Technical Report CMU-CS-91-132, Carnegie Mellon University. 1991.

Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. “Bundle adjustment - a modern synthesis”. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice (ICCV’00)*, pp. 298–372. Corfu, Greece. September, 2000.

Tsai, R. “A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses”. *IEEE Journal of Robotics and Automation*, Vol. 3, N. 4, pp. 323–344. 1987.

Ulrich, M., Steger, C., and Baumgartner, A. “Real-time object recognition using a modified generalized hough transform”. *Pattern Recognition*, Vol. 36, N. 11, pp. 2557–2570. 2003.

Ulrich, M., Wiedemann, C., and Steger, C. “Cad-based recognition of 3d objects in monocular images”. In *International Conference on Robotics and Automation (ICRA’09)*, pp. 1191–1198. Kobe, Japan. May, 2009.

Vacchetti, L., Lepetit, V., and Fua, P. “Stable real-time 3d tracking using online and offline information”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, pp. 1385–1391. 2004.

von Bank, C., Gavrila, D., and Wöhler, C. “A visual quality inspection system based on a hierarchical 3d pose estimation algorithm”. In *Proceedings of the 25th DAGM symposium on Pattern Recognition*, volume 2781, pp. 179–186. Magdeburg, Germany. September, 2003.

Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., and Schmalstieg, D. “Pose tracking from natural features on mobile phones”. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR’08)*, pp. 125–134. Cambridge, UK. September, 2008.

Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., and Schmalstieg, D. “Real-time detection and tracking for augmented reality on mobile phones”. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 16, N. 3, pp. 355 –368. 2010.

Wagner, D. and Schmalstieg, D. “Artoolkitplus for pose tracking on mobile devices”. In *Proceedings of 12th Computer Vision Winter Workshop (VCWW’07)*, pp. 139–146. St. Lambrecht, Austria. February, 2007.

Wang, B., Bai, X., Wang, X., Liu, W., and Tu, Z. "Object recognition using junctions". In *Proceedings of the 11th European conference on Computer vision (ECCV'10)*, pp. 15–28. Heraklion, Crete, Greece. September, 2010.

Weidenhausen, J., Knoepfle, C., and Stricker, D. "Lessons learned on the way to industrial augmented reality applications, a retrospective on arvika". *Computers and Graphics*, Vol. 27, N. 6, pp. 887–891. 2003.

Wiedemann, C., Ulrich, M., and Steger, C. "Recognition and tracking of 3d objects". In *Proceedings of the 30th DAGM symposium on Pattern Recognition*, pp. 132–141. Munich, Germany. June, 2008.

Wilson, R. H. *On geometric assembly planning.* Phd-thesis, Stanford University. 1992.

Wuest, H., Wientapper, F., and Stricker, D. "Adaptable model-based tracking using analysis-by-synthesis techniques". In *Proceedings of the 12th international conference on Computer analysis of images and patterns (CAIP'07)*, pp. 20–27. Vienna, Austria. August, 2007.

Xu, K., Chia, K. W., and Cheok, A. D. "Real-time camera tracking for marker-less and unprepared augmented reality environments". *Image Vision Computing*, Vol. 26, N. 5, pp. 673–689. 2008.

Yamakawa, S. and Shimada, K. "Polygon crawling: Feature-edge extraction from a general polygonal surface for mesh generation". *Engineering With Computers*, Vol. 26, pp. 257–274. 2005.

Yuan, M. L., Ong, S. K., and Nee, A. Y. C. "Augmented reality for assembly guidance using a virtual interactive tool". *International Journal of Production Research*, Vol. 46, N. 7, pp. 1745–1767. 2008.

Zauner, J., Haller, M., Brandl, A., and Hartman, W. "Authoring of a mixed reality assembly instructor for hierarchical structures". In *Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'03)*, pp. 237–246. Tokyo, Japan. October, 2003.

Zhang, X., Fronz, S., and Navab, N. "Visual marker detection and decoding in ar systems: A comparative study". In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality (ISMAR'02)*, pp. 97–106. Darmstadt, Germany. September, 2002.

Zhang, Z. “A flexible new technique for camera calibration”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, pp. 1330–1334. 2000.